UQ-LOB: UNCERTAINTY-AWARE LIMIT ORDER BOOK MID-PRICE FORECASTING

Anonymous authors
Paper under double-blind review

ABSTRACT

Forecasting short-horizon mid-price movements from limit order book (LOB) data is central to algorithmic trading, yet most deep LOB forecasters are point predictors: they output a direction or a displacement, but never indicate which of their forecasts can be trusted. We introduce *UQ-LOB*, a lightweight, encoder-agnostic uncertainty quantification module that attaches to any pretrained LOB encoder and, in the spirit of attentive neural processes, conditions each forecast on a context set of recently completed windows whose outcomes are already realised. The *UQ-regression* variant outputs a calibrated Gaussian over the future tick displacement, while the *UQ-classification* variant outputs a categorical distribution over down/up/stationary. Both expose a scalar confidence (predicted signal-to-noise ratio or class probability) that supports selective prediction. On 5.2 billion LOB events across seven cryptocurrency assets and horizons of 5, 10 and 15 seconds, UQ-regression attains near-nominal 68% interval coverage, and restricting to the most confident 10% of predictions raises directional macro F1 by 0.11–0.15 for UQ-regression and 0.05–0.11 for UQ-classification, at every horizon. On large, economically meaningful moves, the tightest confidence tier reaches a directional F1 of 0.88 (down) and 0.83 (up) at the 5-second horizon.

1 INTRODUCTION

The limit order book (LOB) is the mechanism through which price discovery occurs in most electronic markets (Gould et al., 2013). It aggregates resting buy (bid) and sell (ask) orders by price level; the best bid and best ask define the spread, and their midpoint, the mid-price, is the standard instantaneous valuation of the asset (Glosten & Milgrom, 1985; Kolm et al., 2023). An incoming order that crosses the spread executes immediately against resting liquidity, whereas one that does not rests in the book at its own price level (Briola et al., 2025). Because thousands of such events arrive every second, the LOB carries microstructural signal about the next few seconds of price formation that predictive models seek to exploit (Sirignano & Cont, 2019; Zhang et al., 2019a; Lucchese et al., 2024).

Deep LOB forecasters are overwhelmingly *point predictors*: they emit a direction or a displacement, but not how much that output should be trusted. For a trading decision this omission is costly. Predictability varies sharply across market regimes and even across consecutive windows, and every action incurs fees and slippage, so the practically relevant question is rarely "which way will the price move?" but "is *this* forecast reliable enough to act on?", the setting of selective prediction (Chow, 1970; El-Yaniv & Wiener, 2010; Geifman & El-Yaniv, 2017). Answering it requires input-dependent (aleatoric) uncertainty that is *calibrated*: an interval claimed to cover 68% of outcomes should do so empirically.

Figure 1 illustrates the output we target. Instead of a single value, the model returns a distribution $\mathcal{N}(\mu, \sigma^2)$ over the mid-price displacement at $t + h$, and the ratio $|\mu|/\sigma$ provides a scalar confidence that can be thresholded. We obtain this with *UQ-LOB*, a module attached to the hidden representation of a pretrained LOB encoder. Inspired by conditional and attentive neural processes (Garnelo et al., 2018a; Kim et al., 2019), UQ-LOB conditions each prediction on a *context set* of recently completed windows whose realised outcomes are already known at time t, so that the forecast can adapt to the prevailing regime without any gradient update.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

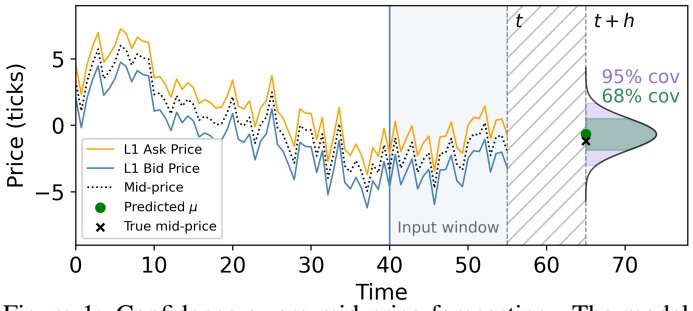


Figure 1: Confidence-aware mid-price forecasting. The model sees only the input window ending at t (shaded) and returns a Gaussian over the displacement at $t + h$; its 68% and 95% intervals give a thresholdable measure of how much the forecast should be trusted.

To the best of our knowledge, no prior work equips mid-price forecasting with calibrated, input-dependent uncertainty of this kind. We instantiate UQ-LOB as a *UQ-regression* variant, which outputs a Gaussian over the tick displacement, and a *UQ-classification* variant, which outputs a categorical distribution over the three directional classes (down, up, stationary), and we compare the two under matched confidence gating. Our contributions are:

- **An encoder-agnostic, in-context UQ head.** An attentive head that conditions on the realised outcomes of recent windows and produces a calibrated Gaussian $\mathcal{N}(\mu, \sigma^2)$ over tick displacement, with confidence given by its signal-to-noise ratio $|\mu|/\sigma$. Its objective combines a heteroscedastic likelihood with magnitude weighting, a directional margin, and a differentiable magnitude-conditional calibration penalty that turns the post-hoc calibration metrics of Levi et al. (2022) into a training signal.
- **A classification counterpart on the same trunk**, trained on the three-class label with confidence given by its maximum class probability, enabling a controlled comparison between continuous and discrete targets at identical parameter count.
- **An evaluation protocol for selective LOB prediction** based on *directional macro F1*, which removes the trivial effect of the stationary class under gating, a magnitude-restricted analysis on large moves, and prior-matched random baselines. On 5.2 billion events across seven assets and three horizons, confidence gating improves directional reliability at every horizon, and the regression head matches the dedicated classifier at the top confidence decile while additionally providing a magnitude and a calibrated interval.

2 Related Work

LOB forecasting. A large share of deep learning work on LOB data targets next-message or next-event prediction (Nagy et al., 2023), an autoregressive formulation in which errors compound as the effective horizon grows. The FI-2010 benchmark (Ntakaris et al., 2018) standardised a second line of work that predicts the direction of the mid-price over a horizon of k future events as a three-class label (Zhang et al., 2019a; Xiao et al., 2025). Architectures for this task have progressed from CNNs (Tsantekidis et al., 2017) and LSTMs (Tsantekidis et al., 2020) to bilinear and attention-based models (Tran et al., 2018; Wallbridge, 2020), pretrained Transformers (Xiao et al., 2025), and BERT-style encoders trained on Level-2/Level-3 event streams (Linna et al., 2025). All of these, including work that treats the displacement as a regression target (Yang et al., 2025), are point predictors with no accompanying measure of predictive confidence.

Uncertainty in LOB forecasting. Quantile regression (Zhang et al., 2019b) jointly models several return quantiles and avoids quantile crossing, but targets ask- and bid-side quantiles separately and fixes the quantile levels in advance rather than yielding a single predictive distribution. BDLOB (Zhang et al., 2018) applies dropout variational inference (Gal & Ghahramani, 2016) to a LOB CNN and shows that the resulting posterior predictive uncertainty can be used for position sizing and for avoiding unnecessary trades; Magris et al. (2023) obtain predictive distributions over class probabilities from a Bayesian bilinear network. Both derive uncertainty from a distribution over network *weights* (epistemic uncertainty) and require multiple stochastic passes at inference, whereas our head predicts input-dependent (aleatoric) uncertainty in a single forward pass and conditions it on recently realised outcomes.

Calibration, selective prediction, and neural processes. Heteroscedastic Gaussian regression (Nix & Weigend, 1994; Kendall & Gal, 2017) provides aleatoric uncertainty but is prone to *loss attenua-*

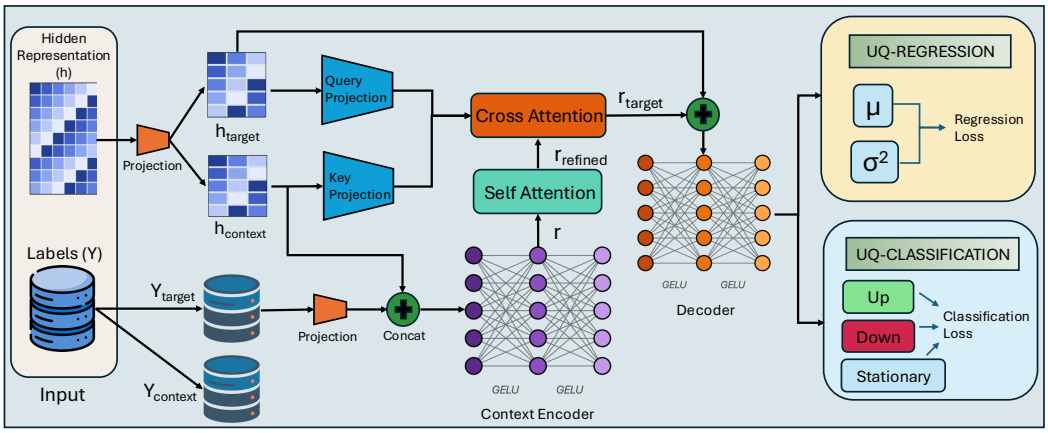


Figure 2: The UQ-LOB. Context and target representations from a pretrained encoder are projected into a shared space; each context representation is fused with its (scaled) realised label, refined by self-attention, and read out by the target through cross-attention. The decoder outputs either a Gaussian (μ, σ^2) over tick displacement (UQ-regression) or three class logits (UQ-classification).

tion (Seitzer et al., 2022; Stirn et al., 2023); post-hoc recalibration (Guo et al., 2017; Kuleshov et al., 2018), deep ensembles (Lakshminarayanan et al., 2017), and conformal methods (Angelopoulos & Bates, 2023) improve calibration but do not by themselves make it input-dependent, and predictive uncertainty is known to degrade under distribution shift (Ovadia et al., 2019), which is endemic to financial data. Selective prediction with a reject option (Chow, 1970; El-Yaniv & Wiener, 2010; Geifman & El-Yaniv, 2017) and the maximum softmax probability as a confidence score (Hendrycks & Gimpel, 2016) form the evaluation lens we adopt. Finally, neural processes (Garnelo et al., 2018b) and their conditional and attentive variants (Garnelo et al., 2018a; Kim et al., 2019) learn predictive distributions conditioned on an observed context set; we adapt this in-context formulation to the LOB setting, where the context is the set of most recently completed windows whose labels are already realised.

3 UQ-LOB: The Uncertainty Quantification Head

Point predictions of mid-price displacement carry no indication of their own reliability, which is particularly problematic in LOB markets, where volatility, liquidity and directional momentum shift rapidly across regimes. UQ-LOB addresses this by producing, for a target window with hidden representation $\mathbf{h}^t$, a full predictive distribution conditioned on a context set $\mathcal{C}$ of (representation, realised label) pairs from recently completed windows (Figure 2). The head operates on the encoder's hidden representation rather than on raw market features, so any pretrained LOB encoder can serve as its starting point (Section 4.3).

3.1 Problem Setup and Labels

The event stream of each asset is segmented into non-overlapping windows of $L = 512$ events (Section 4.2). Let t denote the timestamp of the last event of a window, i.e. the model's present moment. To reflect execution latency, the start price p_{start} is the mid-price at the first event after a small fixed delay d past t, and the end price p_{end} is the mean mid-price over the $k = 10$ events immediately preceding $t_{\text{start}} + h$, which damps single-tick noise without washing out genuine displacement at the tick rates of our data. The regression target is the raw tick displacement $y_h(t) = p_{\text{end}} - p_{\text{start}}$. The classification label thresholds the same displacement at $\pm\delta_h(t)$, a volatility-scaled cutoff derived from the training-split log-return distribution of each asset and horizon and converted to ticks at the current price level (Appendix A.7). All labels are computed independently per asset and horizon.

3.2 In-Context Conditioning

The head follows conditional neural processes (CNPs) (Garnelo et al., 2018a), which cast regression as an in-context learning problem: rather than learning a fixed input–output map, a CNP summarises an observed context set $\mathcal{C} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{C}$ into a representation $\mathbf{r}(\mathcal{C})$ and predicts $p(y^* \mid \mathbf{x}^*, \mathcal{C}) = p(y^* \mid \mathbf{x}^*, \mathbf{r}(\mathcal{C}))$. The original CNP aggregates context points by a mean, which weighs every point equally; attentive neural processes (Kim et al., 2019) let the query attend selectively to the most relevant points, and we adopt this form.

In our setting the inputs are windows. Let $\mathbf{X}_i \in \mathbb{R}^{L\times D}$ be a raw window of L events with D features, encoded by a pretrained encoder into $\mathbf{h}_i = f_\theta(\mathbf{X}_i) \in \mathbb{R}^{d_h}$. A training instance consists of $N = C + 1$ windows: C context pairs $\{(\mathbf{h}_i^c, y_i^c)\}_{i=1}^{C}$ and a single withheld target $(\mathbf{h}^t, y^t)$. The context set is drawn from the windows that most recently precede the target, subject to a *causal constraint*: if a context window ends at t_i^c, its label is realised at $e_i^c = t_i^c + d + h$, and we require

$$e_i^c \;\le\; t \qquad \forall\, i \in \{1, \dots, C\}, \tag{1}$$

so that every context label is known at the moment the target prediction is made. This is strictly stronger than merely requiring the context and target input windows not to overlap: it also excludes context windows whose label horizon runs past t, which would leak future price information into the prediction. The same construction is used at training, validation and test time, so the test-time predictor only ever conditions on information available at t (Appendix A.5).

3.3 Head Architecture

Figure 2 shows the head. Context and target representations are projected into a shared space by a common linear map W_p: $\tilde{\mathbf{h}}_i^c = W_p \mathbf{h}_i^c$ and $\tilde{\mathbf{h}}^t = W_p \mathbf{h}^t$. Context labels are divided by a fixed, asset- and horizon-specific scale y_{ref} (Appendix B.4) and projected through a $\tanh$-bounded linear layer, $\mathbf{y}_i^{\text{proj}} = \tanh\big(W_y\, y_i^c / y_{\text{ref}}\big)$. Each projected representation is concatenated with its projected label and passed through the *context encoder*, a three-layer GELU MLP,

$$\mathbf{r}_i = \text{MLP}_{\text{enc}}\Big(\Big[\tilde{\mathbf{h}}_i^c\,;\, \mathbf{y}_i^{\text{proj}}\Big]\Big), \qquad i = 1, \dots, C. \tag{2}$$

Self-attention over $\{\mathbf{r}_i\}_{i=1}^{C}$ then lets each context point re-weigh itself by inter-context relevance, yielding a refined set $\{\mathbf{r}_i^{\text{ref}}\}_{i=1}^{C}$. The target queries this refined context by cross-attention: the projected target embedding is mapped to a *query* and the projected context embeddings to *keys* through learned maps W_q, W_k, while the refined representations serve directly as *values*, $\mathbf{q} = W_q \tilde{\mathbf{h}}^t$, $\mathbf{k}_i = W_k \tilde{\mathbf{h}}_i^c$, $\mathbf{v}_i = \mathbf{r}_i^{\text{ref}}$, giving

$$\mathbf{r}^t = \text{softmax}\left(\frac{\mathbf{q}\,\mathbf{K}^\top}{\sqrt{d_r}}\right)\mathbf{V}, \tag{3}$$

where $\mathbf{K}, \mathbf{V}$ stack the keys and values row-wise. Separating keys from values lets the target decide *which* context windows are relevant from their representation alone, while retrieving the label-informed summary of those windows: recent, similarly structured LOB states receive higher weight than dissimilar ones.

The attended summary is concatenated with the target's own projected embedding and passed to a *decoder*, a five-layer GELU MLP, which thus sees both the inferred market context and the target's own features. The UQ-regression decoder outputs

$$[\mu,\, \nu] = \text{MLP}_{\text{dec}}\Big(\Big[\mathbf{r}^t\,;\, \tilde{\mathbf{h}}^t\Big]\Big), \qquad \sigma^2 = y_{\text{ref}}^2 \cdot \text{softplus}(\nu) + \epsilon, \tag{4}$$

giving $p(y^t \mid \mathbf{h}^t, \mathcal{C}) = \mathcal{N}(\mu, \sigma^2)$ with ϵ a small constant for numerical stability. The UQ-classification decoder outputs three logits $[\ell_{\text{down}}, \ell_{\text{up}}, \ell_{\text{stat}}] = \text{MLP}_{\text{dec}}^{\text{cls}}([\mathbf{r}^t; \tilde{\mathbf{h}}^t])$. The two variants share the encoder, projection and attention trunk and differ only in the decoder (Appendix A.2).

3.4 Training Objectives

UQ-regression. The regression head minimises a weighted sum of four terms, each computed on target windows only:

$$\mathcal{L}_{\text{reg}} = \lambda_1 \mathcal{L}_{\text{NLL}} + \lambda_2 \mathcal{L}_{\text{WMAE}} + \lambda_3 \mathcal{L}_{\text{dir}} + \lambda_4 \mathcal{L}_{\text{calib}}. \tag{5}$$

(i) Heteroscedastic NLL (Nix & Weigend, 1994; Kendall & Gal, 2017), $\mathcal{L}_{\mathrm{NLL}} = \mathbb{E}\big[\frac{1}{2}\log(2\pi\sigma^2) + (y-\mu)^2/(2\sigma^2)\big]$. Letting σ depend on the input allows the network to temper the residual of hard examples by inflating σ (*loss attenuation*). Seitzer et al. (2022) show this can happen before μ has converged: since $\partial\mathcal{L}_{\mathrm{NLL}}/\partial\mu \propto 1/\sigma^2$, an inflated σ starves the gradient reaching μ and biases μ toward small magnitudes. *(ii) Magnitude-weighted MAE*, which counteracts this bias by up-weighting large displacements,

$$\mathcal{L}_{\mathrm{WMAE}} = \mathbb{E}\left[w(y)\,\frac{|\mu - y|}{y_{\mathrm{ref}}}\right], \qquad w(y) = \min\left(1 + \left(\frac{|y|}{y_{\mathrm{ref}}}\right)^{p},\; w_{\max}\right), \tag{6}$$

with the clamp $w_{\max}$ bounding the influence of extreme outliers ($p = 4$, $w_{\max} = 40$ throughout). *(iii) Directional loss.* Neither of the above targets the discrete up/down/stationary boundary δ_i used at evaluation time. Writing $\tilde{\mu}_i = \mu_i / y_{\mathrm{ref}}$, we add

$$\mathcal{L}_{\mathrm{dir}} = \mathbb{E}\Big[\mathbb{1}[|y_i| \geq \delta_i]\,\big(\tilde{\mu}_i - \mathrm{sign}(y_i)\,m\big)^2 \;+\; \mathbb{1}[|y_i| < \delta_i]\,\big|\tilde{\mu}_i - \mathrm{sign}(y_i)\,m\big|\Big], \tag{7}$$

where δ_i is the asset-, horizon- and window-specific threshold of Appendix A.7.3 and $m = 1$ is a fixed margin. Directional examples receive a quadratic pull toward a confidently signed target $\pm m$; stationary examples only a linear one, so their (weaker, bounded-gradient) sign signal is retained without a strong pull in magnitude. This class-conditional quadratic/linear switching is structurally related to the reverse-Huber (berHu) loss (Zwald & Lambert-Lacroix, 2012; Laina et al., 2016), which switches on residual magnitude rather than on the true class. *(iv) Magnitude-conditional calibration loss.* The three terms above shape μ; none directly corrects σ. We therefore add

$$\mathcal{L}_{\mathrm{calib}} = \frac{1}{B}\sum_{b=1}^{B}\left(\mathbb{E}_{i\in\mathcal{B}_b}\left[\frac{|\mu_i - y_i|}{\sigma_i}\right] - \sqrt{\tfrac{2}{\pi}}\right)^2, \tag{8}$$

where $\{\mathcal{B}_b\}_{b=1}^{B}$ partitions the batch into B quantile bins of $|\mu|$ and $\sqrt{2/\pi} = \mathbb{E}|Z|$, $Z \sim \mathcal{N}(0,1)$, is the expected standardised residual of a calibrated Gaussian. Unlike remedies that reweight the NLL globally (Seitzer et al., 2022) or decouple the mean and variance branches architecturally (Stirn et al., 2023), $\mathcal{L}_{\mathrm{calib}}$ turns the post-hoc calibration metrics of Levi et al. (2022) into a differentiable penalty, binning on predicted magnitude so that calibration is enforced separately for small and large predicted moves. All loss hyperparameters are listed in Appendix A.4.

UQ-classification. The classification head minimises a class-weighted cross-entropy over the three-class label, $\mathcal{L}_{\mathrm{cls}} = -\sum_{c=0}^{2} w_c\,\mathbb{1}[y_{\mathrm{cls}} = c]\log p_c$, with $w_c \propto 1/f_c$ the inverse training-split frequency of class c, normalised so that $\sum_c w_c = 3$, which counteracts the class imbalance documented in Appendix B.

3.5 Confidence Scores and Selective-Prediction Protocol

Confidence. For UQ-regression we use the predicted signal-to-noise ratio $\mathrm{SNR} = |\mu|/\sigma$: a large predicted displacement paired with a small predicted uncertainty is both worth acting on and sharply estimated. For UQ-classification we use the softmax confidence $\max_c p_c$ (Hendrycks & Gimpel, 2016). Gating at percentile q retains the $(100-q)\%$ most confident predictions of each asset–horizon test set and evaluates only on those; $q = 0$ is the full test set.

Directional macro F1. To compare the heads on a common footing, the regression mean μ is mapped to a three-class label with the same rule as the ground truth, using a threshold multiplier calibrated on a held-out split so that the predicted stationary proportion matches the true one (Appendix A.7.3). Under gating, the stationary class behaves degenerately: confident predictions are large in magnitude and therefore predominantly directional, so stationary F1 collapses while down/up F1 rise (Table 8), and three-class macro F1 conflates the two effects. We therefore report *directional macro F1*, the mean of the down and up F1 scores, and provide a prior-matched reference: a classifier predicting at random with the true class priors attains an expected F1 of p_c for class c, so its directional macro F1 is $(1 - p_{\mathrm{stat}})/2$. Because gating changes the class composition of the retained subset, this reference is only valid for the full test set and rises under gating; the large-move analysis of Section 5.3 fixes it at 0.5 by construction.

Calibration and magnitude-weighted fit. We report the empirical coverage of the nominal 68% and 95% intervals, $\mathrm{cov}_{68} = \Pr[|y-\mu| \leq \sigma]$ and $\mathrm{cov}_{95} = \Pr[|y-\mu| \leq 1.96\sigma]$, and the negative

log predictive density (NLPD). Point fit is summarised by a magnitude-weighted R^2,

$$R^2_w = 1 - \frac{\sum_i w_i (y_i - \mu_i)^2}{\sum_i w_i (y_i - \bar{y}_w)^2}, \qquad \bar{y}_w = \frac{\sum_i w_i y_i}{\sum_i w_i}, \tag{9}$$

with $w_i = w(y_i)$ from Equation 6. Large moves are those that cross the stationary threshold and, after fees, are the only ones a trader can profit from; WR^2 therefore measures fit on the moves that matter rather than on the small fluctuations that dominate standard R^2.

4 Experimental Setup

4.1 Dataset

We use Level-2 and Level-3 order book data collected from the Kraken cryptocurrency exchange through its public market data API for seven USD-quoted assets: Bitcoin (BTC), Ethereum (ETH), Solana (SOL), Litecoin (LTC), Dogecoin (DOGE), Sui (SUI) and Bittensor (TAO). The data span November 2025 to February 2026 and contain approximately 5.2 billion events over 600 asset-days, with event rates from 39 to 156 events/s and mean inter-event times from 6.4 to 26.0 ms (Table 6, Appendix B). For every asset, data up to 18 January 2026 are used for training, 19–31 January for validation and 1–15 February for testing (about 68%, 15% and 17% of asset-days). The validation period is split into two equal halves for model selection and for calibrating the threshold multiplier of Appendix A.7.3; the test period is strictly out-of-sample and chronologically after both. Because no data-sharing agreement is in place with the exchange, the raw data cannot be redistributed; we release code, all label thresholds and scale values, and the dataset statistics needed to reconstruct the pipeline on comparable data (Reproducibility Statement).

4.2 Input Representation

Following Linna et al. (2025), each event is encoded as a discrete token from a vocabulary of $|\mathcal{V}| = 439$ entries built from five categorical attributes (event type, side, discretised volume, discretised price distance from the opposing best quote, and a simultaneity flag); events deeper than $L_{\max} = 10$ levels are discarded. Each event also carries seven continuous features. Three follow Linna et al. (2025): log inter-event time, log tick distance from the opposing best quote and log relative volume. Four capture directional order-flow pressure at the top of the book. Building on the order flow imbalance of Cont et al. (2014), the depth-normalised flow imbalance

$$\mathrm{DNFI}_t = \frac{\Delta Q^b_{1,t} - \Delta Q^a_{1,t}}{Q^b_{1,t} + Q^a_{1,t}} \in [-1, +1], \qquad \mathrm{QI}_t = \frac{Q^b_{1,t} - Q^a_{1,t}}{Q^b_{1,t} + Q^a_{1,t}} \in [-1, +1], \tag{10}$$

tracks the net change in best-level queue sizes $Q^b_{1,t}, Q^a_{1,t}$ relative to resting depth (Δ denotes the change from the previous event); two cumulative sums of DNFI over the preceding 50 and 200 events capture short- and medium-term flow momentum; and the queue imbalance QI_t complements the flow signal with a static snapshot of resting-liquidity asymmetry. The stream is segmented into non-overlapping windows of $L = 512$ events, each forming one model input (Appendix A.6).

4.3 Encoder, Training and Model Size

As encoder we use D-TABL, a deeper variant of the temporal attention-augmented bilinear network of Tran et al. (2018) adapted to our 512-event windows (Appendix A.1). It is first pretrained on the three-class label alone, independently for each of the 21 asset–horizon pairs. UQ-LOB is then attached to the pretrained encoder and the whole model is fine-tuned end-to-end with Equation 5 (UQ-regression) or the weighted cross-entropy (UQ-classification). Each training instance consists of $N = 16$ windows, $C = 15$ context windows and one withheld target, and each step processes a batch of 16 such instances. The encoder has 0.27M parameters; the shared projection and context/attention trunk adds 0.79M, and each decoder 0.17M, so both variants total 1.23M parameters (Table 4). Training all 42 models requires about 210 GPU-hours on a single NVIDIA GH200; full details and hyperparameters are given in Appendices A.3 and A.4.

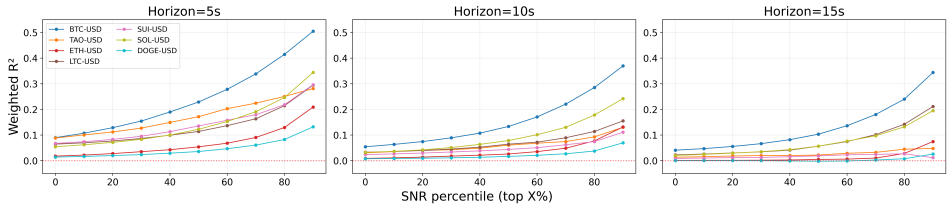


Figure 4: Magnitude-weighted R^2 (Equation 9) of the UQ-regression head as a function of the SNR percentile retained, per asset and horizon.

Table 1: Directional macro F1 (mean of down and up F1; stationary excluded) as a function of the confidence percentile retained, for UQ-regression (SNR-gated) and UQ-classification (softmax-gated), pooled over the seven assets. Row 0 is the full test set. A prior-matched random classifier attains $(1 - p_{\text{stat}})/2$ on the full test set: 0.337 at 5 s ($p_{\text{stat}} = 0.326$), 0.350 at 10 s ($p_{\text{stat}} = 0.300$) and 0.362 at 15 s ($p_{\text{stat}} = 0.276$); because gating changes the class mix of the retained subset, this full-set reference does not apply to the gated rows (Section 5.2).

H	**5s**						**10s**						**15s**					
	UQ-Regression			UQ-Classification			UQ-Regression			UQ-Classification			UQ-Regression			UQ-Classification		
Conf.	**F1(↓)**	**F1(↑)**	Mean	**F1(↓)**	**F1(↑)**	Mean	**F1(↓)**	**F1(↑)**	Mean	**F1(↓)**	**F1(↑)**	Mean	**F1(↓)**	**F1(↑)**	Mean	**F1(↓)**	**F1(↑)**	Mean
0	0.45	0.39	0.42	0.48	0.42	0.45	0.43	0.39	0.41	0.46	0.43	0.45	0.44	0.36	0.40	0.47	0.42	0.45
10	0.47	0.41	0.44	0.49	0.43	0.46	0.45	0.41	0.43	0.47	0.44	0.45	0.46	0.38	0.42	0.48	0.43	0.46
20	0.49	0.43	0.46	0.50	0.43	0.47	0.47	0.43	0.45	0.48	0.44	0.46	0.48	0.40	0.44	0.49	0.43	0.46
30	0.51	0.45	0.48	0.51	0.43	0.47	0.49	0.45	0.47	0.48	0.45	0.47	0.50	0.41	0.46	0.50	0.43	0.47
40	0.52	0.47	0.49	0.52	0.44	0.48	0.50	0.46	0.48	0.49	0.45	0.47	0.51	0.42	0.47	0.51	0.44	0.47
50	0.53	0.48	0.50	0.53	0.45	0.49	0.51	0.47	0.49	0.50	0.45	0.48	0.52	0.42	0.47	0.52	0.44	0.48
60	0.54	0.49	0.51	0.55	0.46	0.50	0.52	0.48	0.50	0.51	0.46	0.48	0.53	0.43	0.48	0.53	0.44	0.48
70	0.55	0.50	0.52	0.56	0.47	0.51	0.53	0.49	0.51	0.52	0.47	0.49	0.54	0.43	0.48	0.54	0.45	0.49
80	0.57	0.52	0.54	0.58	0.48	0.53	0.54	0.50	0.52	0.53	0.47	0.50	0.55	0.44	0.49	0.55	0.45	0.50
90	0.60	0.54	0.57	0.61	0.51	0.56	0.57	0.53	0.55	0.55	0.46	0.50	0.57	0.46	0.51	0.57	0.47	0.52

5 Results

5.1 Calibration of the Regression Head

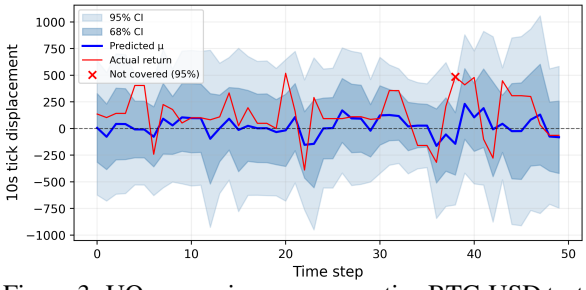


Figure 3: UQ-regression on consecutive BTC-USD test windows at the 10 s horizon: predicted μ with 68% and 95% intervals against the realised displacement; crosses mark realisations outside the 95% interval.

Figure 3 gives a qualitative view of the predictive distribution, and Table 3 quantifies it across horizons. Empirical 68% coverage is 68.1–68.2%, essentially nominal, and it is stable across horizons. The 95% interval, however, covers only 90.4–90.7% of outcomes: the residual distribution is heavier tailed than a Gaussian, and Equation 8 constrains only the first absolute moment of the standardised residual, so it enforces the one-sigma level but not the tails. We return to this in Section 7.

5.2 Confidence Gating

UQ-regression. Figure 4 plots WR^2 against the SNR percentile retained. For every asset and horizon, WR^2 rises monotonically as the retained subset is restricted to higher-SNR predictions, most markedly for BTC-USD (from roughly 9% on the full 5 s test set to over 50% in the top decile). Three-class accuracy under SNR gating follows the same pattern (Figure 10, Appendix C; BTC-USD rises from about 45% to 57% at 5 s). Table 1 reports directional macro F1: between the full test set and the most confident 10% of predictions it rises from 0.42 to 0.57 at 5 s, from 0.41 to 0.55 at 10 s and from 0.40 to 0.51 at 15 s, and Figure 5 (top) shows that the increase is monotone over the whole percentile range for every asset, with DOGE at 15 s the only mild exception at the highest tiers. Two caveats apply. Both heads score lower on the up class than on the down class at

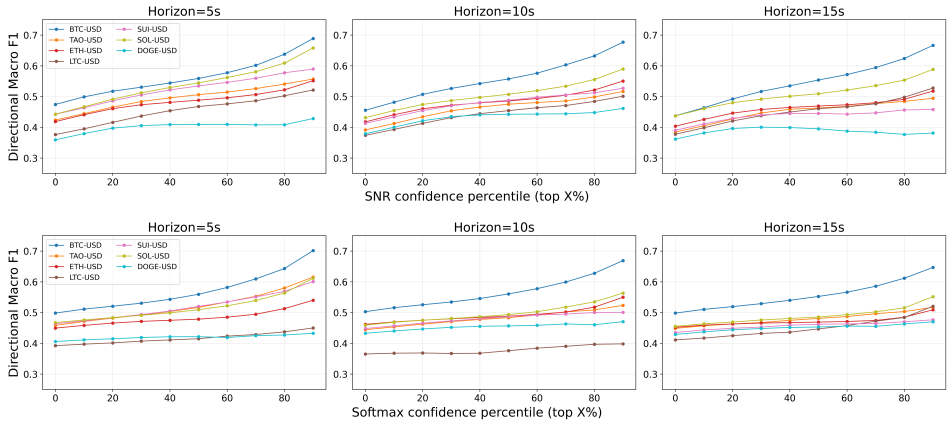


Figure 5: Directional macro F1 as a function of the confidence percentile retained, per asset and horizon. Top: UQ-regression gated by SNR. Bottom: UQ-classification gated by softmax confidence.

every tier; as the asymmetry is shared, it reflects the test period rather than either head. And since the prior-matched reference rises with the gate as the retained subset becomes more directional, the top-decile values should not be read against the full-set reference of 0.34–0.36; the magnitude-restricted analysis below is the cleaner test.

UQ-classification. Under softmax gating the same directional macro F1 rises from 0.45 to 0.56 at 5 s, 0.45 to 0.50 at 10 s and 0.45 to 0.52 at 15 s (Table 1), monotonically at every horizon (Figure 5, bottom). Three-class macro F1 and accuracy show the same behaviour (Figures 9 and 10, Appendix C): for BTC-USD, macro F1 climbs from 0.47 on the full 5 s test set to nearly 0.68 in the top decile, and accuracy from about 48% to 68%. Softmax confidence is therefore a reliable ranking signal for this head as well.

5.3 Large Moves

Table 2 restricts evaluation to large true moves, defined per asset and horizon as $|y| > q_{67}$, the upper tercile of the true displacement magnitude. By construction these examples are almost never stationary and the down/up split is close to balanced, so a prior-matched random classifier scores about 0.5 on either class regardless of the gate, which makes this the cleanest test of whether confidence identifies correct directional calls. Per-class F1 rises steadily with SNR at every horizon: at 5 s, F1(down) increases from 0.69 in the top 50% to 0.88 in the top 1%, and F1(up) from 0.63 to 0.83; the same pattern holds at 10 s and 15 s, with a smaller gain on the up class at 15 s. This corroborates the gating result of Table 1 under an independent, magnitude-based restriction, and it is exactly the regime, large moves at short horizons, in which a reliable signal is most valuable after transaction costs.

Table 2: Per-class directional F1 of UQ-regression on large true moves ($|y| > q_{67}$) at increasing SNR tiers.

	Top 50%		Top 30%		Top 10%		Top 5%		Top 1%	
H	**F1(↓)**	**F1(↑)**	**F1(↓)**	**F1(↑)**	**F1(↓)**	**F1(↑)**	**F1(↓)**	**F1(↑)**	**F1(↓)**	**F1(↑)**
5s	0.69	0.63	0.73	0.67	0.81	0.75	0.84	0.79	0.88	0.83
10s	0.65	0.59	0.68	0.63	0.75	0.70	0.78	0.73	0.85	0.78
15s	0.65	0.52	0.68	0.53	0.74	0.59	0.76	0.61	0.81	0.68

Table 3: Calibration of UQ-regression (full test set, pooled over assets). Calib. error: mean over assets of $|\mathrm{cov}_{68} - 0.68|$.

Metric	**5s**	**10s**	**15s**
Cov. 68 (nominal 0.68)	0.682	0.682	0.681
Cov. 95 (nominal 0.95)	0.904	0.904	0.907
Calib. error	0.048	0.048	0.045
NLPD	5.278	5.586	5.778

5.4 Regression versus Classification Head

The two heads share every parameter except the decoder, so Table 1 is a controlled comparison of the two targets. On the full test set the dedicated classifier is stronger by 0.03–0.05 directional macro F1 (0.45 versus 0.40–0.42), as one would expect from a head trained directly on the evaluation label. Gating closes this gap: at the top decile the regression head matches or exceeds the classifier (0.57 versus 0.56 at 5 s, 0.55 versus 0.50 at 10 s, 0.51 versus 0.52 at 15 s), and the gain from gating is larger for SNR than for softmax confidence at every horizon (+0.11 to +0.15 versus +0.05 to +0.11). Since SNR combines predicted magnitude with predicted precision, its top tier is enriched for large, well-estimated moves, exactly the ones on which the regression head is strongest (Section 5.3). The regression head thus offers, at identical cost, a calibrated magnitude and interval in addition to a direction, and loses nothing in directional reliability on the subset a trader would act on.

6 Conclusion

We introduced UQ-LOB, an encoder-agnostic module that turns a pretrained LOB encoder into a calibrated probabilistic forecaster by conditioning on the realised outcomes of recently completed windows. Across seven cryptocurrency assets and three horizons, its regression variant attains near-nominal 68% coverage, and gating by its own signal-to-noise ratio raises directional macro F1 by 0.11–0.15 at the top confidence decile (0.05–0.11 for the classification variant under softmax gating), reaching F1 of 0.88 (down) and 0.83 (up) on large 5-second moves. Although not trained on the discrete label, the regression head matches the dedicated classifier at the top decile while also supplying a magnitude and a calibrated interval: a single head can serve both continuous and directional decisions together with a grounded measure of when to act.

7 Limitations and Future Work

Tail calibration. The 95% interval under-covers (90.4–90.7%), because the Gaussian likelihood and the first-moment calibration penalty do not capture the heavy tails of tick displacements; a Student-t likelihood or post-hoc conformal recalibration of the intervals is a direct extension. **Baselines and variance.** Our comparison is between the two UQ heads on a shared trunk; a comparison against weight-space uncertainty (MC dropout, deep ensembles) and against gating the base encoder's own softmax, together with multi-seed error bars and ablations of the four loss terms and of the context set, would sharpen the attribution of the gains to in-context conditioning. **Distribution shift.** The training and test windows differ markedly in regime for BTC-USD (stationary share 70% versus 27% at 5 s, Appendix B.3); that gating still works there is encouraging, but a longer test period spanning several regimes is needed to establish robustness. **Economic evaluation.** We evaluate directional reliability, not profitability; a trading simulation with realistic execution and fees is required before any claim about returns. **Scope.** Models are trained separately per asset and horizon on a single venue; a single model trained jointly across horizons, and wider input windows for longer horizons, are natural next steps.

Ethics Statement

This work uses only public market data collected through the exchange's public API; it contains no personal data. The models forecast short-horizon price movements and could inform trading decisions; we make no claim of profitability, report all limitations above, and note that acting on such forecasts carries financial risk. We see no direct negative societal impact beyond those common to quantitative trading research.

Reproducibility Statement

We release the complete source code for the D-TABL encoder, both UQ heads, and the training and evaluation pipelines, including the context construction of Appendix A.5. The raw order book data cannot be redistributed (Section 4.1); to support reproduction on comparable Level-2/Level-3 data from any venue we release all label thresholds and scale values (Table 7), the dataset statistics (Table 6), and the full specification of the labelling procedure (Appendix A.7), input repre-

sentation (Section 4.2 and Appendix A.6), training strategy (Appendix A.3) and hyperparameters (Appendix A.4).

AI USE STATEMENT

No generative AI tools were used for any task requiring disclosure under the ICLR 2027 AI Policy for Authors: the encoder and UQ head architectures, the training objectives, the experimental design, the experiments and the interpretation of all reported results are the authors' own work. Under the recommended-disclosure categories, generative AI tools were used to edit the manuscript for clarity and readability, to help reorganise sections, and to assist in structuring the architectural diagrams. All AI-assisted text and figures were reviewed and finalised by the authors, who take full responsibility for the final content, including all text, claims and artifacts.

REFERENCES

Anastasios N Angelopoulos and Stephen Bates. Conformal prediction: A gentle introduction. *Foundations and Trends in Machine Learning*, 16(4):494–591, 2023.

Antonio Briola, Silvia Bartolucci, and Tomaso Aste. Deep limit order book forecasting: a microstructural guide. *Quantitative Finance*, 25(7):1101–1131, 2025.

C. K. Chow. On optimum recognition error and reject tradeoff. *IEEE Transactions on Information Theory*, 16(1):41–46, 1970.

Rama Cont, Arseniy Kukanov, and Sasha Stoikov. The price impact of order book events. *Journal of Financial Econometrics*, 12(1):47–88, 2014.

Ran El-Yaniv and Yair Wiener. On the foundations of noise-free selective classification. *Journal of Machine Learning Research*, 11:1605–1641, 2010.

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pp. 1050–1059. PMLR, 2016.

Marta Garnelo, Dan Rosenbaum, Christopher J Maddison, Tiago Ramalho, David Saxton, Murray Shanahan, Yee Whye Teh, Danilo J Rezende, and S M Ali Eslami. Conditional neural processes. In *Proceedings of ICML*, pp. 1704–1713, 2018a.

Marta Garnelo, Jonathan Schwarz, Dan Rosenbaum, Fabio Viola, Danilo Jimenez Rezende, SM Eslami, and Yee Whye Teh. Neural processes. *arXiv preprint arXiv:1807.01622*, 2018b.

Yonatan Geifman and Ran El-Yaniv. Selective classification for deep neural networks. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Lawrence R Glosten and Paul R Milgrom. Bid, ask and transaction prices in a specialist market with heterogeneously informed traders. *Journal of Financial Economics*, 14(1):71–100, 1985.

Martin D Gould, Mason A Porter, Stacy Williams, Mark McDonald, Daniel J Fenn, and Sam D Howison. Limit order books. *Quantitative Finance*, 13(11):1709–1742, 2013.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *Proceedings of ICML*, pp. 1321–1330, 2017.

Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *arXiv preprint arXiv:1610.02136*, 2016.

Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? *Advances in neural information processing systems*, 30, 2017.

Hyunjik Kim, Andriy Mnih, Jonathan Schwarz, Marta Garnelo, Ali Eslami, Dan Rosenbaum, Oriol Vinyals, and Yee Whye Teh. Attentive neural processes. In *Proceedings of ICLR*, 2019.

Petter N Kolm, Jeremy Turiel, and Nicholas Westray. Deep order flow imbalance: Extracting alpha at multiple horizons from the limit order book. *Mathematical Finance*, 33(4):1044–1081, 2023.

Volodymyr Kuleshov, Nathan Fenner, and Stefano Ermon. Accurate uncertainties for deep learning using calibrated regression. In *Proceedings of ICML*, pp. 2796–2804, 2018.

Iro Laina, Christian Rupprecht, Vasileios Belagiannis, Federico Tombari, and Nassir Navab. Deeper depth prediction with fully convolutional residual networks. In *2016 Fourth international conference on 3D vision (3DV)*, pp. 239–248. IEEE, 2016.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Dan Levi, Liran Gispan, Niv Giladi, and Ethan Fetaya. Evaluating and calibrating uncertainty prediction in regression tasks. *Sensors*, 22(15):5540, 2022.

Eljas Linna, Kestutis Baltakys, Alexandros Iosifidis, and Juho Kanniainen. Lobert: Generative ai foundation model for limit order book messages, 2025. URL https://arxiv.org/abs/2511.12563.

Lorenzo Lucchese, Mikko S Pakkanen, and Almut ED Veraart. The short-term predictability of returns in order book markets: A deep learning perspective. *International Journal of Forecasting*, 40(4):1587–1621, 2024.

Martin Magris, Mostafa Shabani, and Alexandros Iosifidis. Bayesian bilinear neural network for predicting the mid-price dynamics in limit-order book markets. *Journal of Forecasting*, 42(6): 1407–1428, 2023.

Peer Nagy, Sascha Frey, Silvia Sapora, Kang Li, Anisoara Calinescu, Stefan Zohren, and Jakob Foerster. Generative ai for end-to-end limit order book modelling: A token-level autoregressive generative model of message flow using a deep state space network. In *Proceedings of the fourth ACM international conference on AI in finance*, pp. 91–99, 2023.

David A Nix and Andreas S Weigend. Estimating the mean and variance of the target probability distribution. In *Proceedings of 1994 ieee international conference on neural networks (ICNN'94)*, volume 1, pp. 55–60. IEEE, 1994.

Adamantios Ntakaris, Martin Magris, Juho Kanniainen, Moncef Gabbouj, and Alexandros Iosifidis. Benchmark dataset for mid-price forecasting of limit order book data with machine learning methods. *Journal of Forecasting*, 37(8):852–866, 2018.

Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, David Sculley, Sebastian Nowozin, Joshua V Dillon, Balaji Lakshminarayanan, and Jasper Snoek. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Aiusha Sangadiev, Rodrigo Rivera-Castro, Kirill Stepanov, Andrey Poddubny, Kirill Bubenchikov, Nikita Bekezin, Polina Pilyugina, and Evgeny Burnaev. Deepfolio: Convolutional neural networks for portfolios with limit order book data. *arXiv preprint arXiv:2008.12152*, 2020.

Maximilian Seitzer, Arash Tavakoli, Dimitrije Antic, and Georg Martius. On the pitfalls of heteroscedastic uncertainty estimation with probabilistic neural networks. *arXiv preprint arXiv:2203.09168*, 2022.

Justin Sirignano and Rama Cont. Universal features of price formation in financial markets: perspectives from deep learning. *Quantitative Finance*, 19(9):1449–1459, 2019.

Andrew Stirn, Harm Wessels, Megan Schertzer, Laura Pereira, Neville Sanjana, and David Knowles. Faithful heteroscedastic regression with neural networks. In *International conference on artificial intelligence and statistics*, pp. 5593–5613. PMLR, 2023.

Dat Thanh Tran, Alexandros Iosifidis, Juho Kanniainen, and Moncef Gabbouj. Temporal attention-augmented bilinear network for financial time-series data analysis. *IEEE transactions on neural networks and learning systems*, 30(5):1407–1418, 2018.

Avraam Tsantekidis, Nikolaos Passalis, Anastasios Tefas, Juho Kanniainen, Moncef Gabbouj, and Alexandros Iosifidis. Forecasting stock prices from the limit order book using convolutional neural networks. In *2017 IEEE 19th conference on business informatics (CBI)*, volume 1, pp. 7–12. IEEE, 2017.

Avraam Tsantekidis, Nikolaos Passalis, Anastasios Tefas, Juho Kanniainen, Moncef Gabbouj, and Alexandros Iosifidis. Using deep learning for price prediction by exploiting stationary limit order book features. *Applied Soft Computing*, 93:106401, 2020.

James Wallbridge. Transformers for limit order books. *arXiv preprint arXiv:2003.00130*, 2020.

Yonghong Xiao, Carmine Ventre, Yuanbo Wang, Hexuan Li, Yuhao Huan, and Biao Liu. LiT: Limit order book transformer. *Frontiers in Artificial Intelligence*, 8, 2025.

Mingyu Yang, Woo Chang Kim, Gukyuk Chung, and Chanyeong Kim. Type-based representation of limit order book events for predicting mid-price change. *Available at SSRN 5269651*, 2025.

Zihao Zhang, Stefan Zohren, and Stephen Roberts. Bdlob: Bayesian deep convolutional neural networks for limit order books. *arXiv preprint arXiv:1811.10041*, 2018.

Zihao Zhang, Stefan Zohren, and Stephen Roberts. DeepLOB: Deep convolutional neural networks for limit order books. *IEEE Transactions on Signal Processing*, 67(11):3001–3012, 2019a.

Zihao Zhang, Stefan Zohren, and Stephen Roberts. Extending deep learning models for limit order books to quantile regression. *arXiv preprint arXiv:1906.04404*, 2019b.

Laurent Zwald and Sophie Lambert-Lacroix. The berhu penalty and the grouped effect. *arXiv preprint arXiv:1207.6868*, 2012.

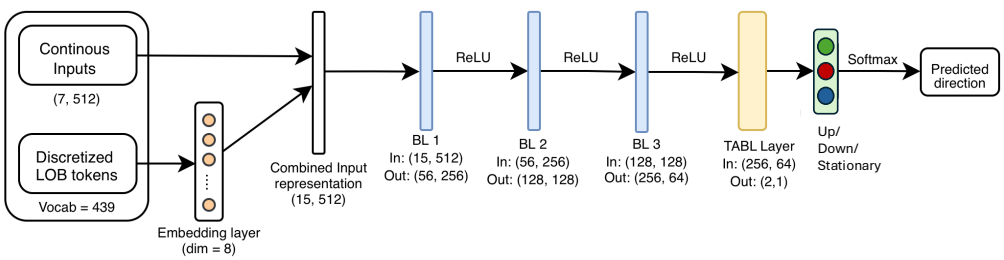


Figure 6: The D-TABL encoder: three bilinear layers followed by a temporal-attention-augmented bilinear (TABL) layer; the token embedding is concatenated with the continuous features at every time step. The UQ head consumes the pre-logit representation.

A Model Specifications

A.1 D-TABL Encoder

UQ-LOB is agnostic to the choice of encoder, whose role is to map raw LOB inputs into a useful representation. Notably, the encoder need not be trained on the exact forecasting task: any classification task predicting the direction of the mid-price is sufficient for pretraining. As a lightweight encoder we adapt the Temporal Attention-Augmented Bilinear Network (TABL) of Tran et al. (2018), originally proposed for mid-price forecasting on FI-2010. The original C(TABL) stacks two bilinear (BL) layers followed by a single attention-augmented TABL layer, operating on a 40×10 input (40 price/volume features, 10 time steps). Our data differ from FI-2010 in feature composition and window length, so we introduce D-TABL with two changes: (i) three BL layers rather than two, and (ii) all layer dimensions re-derived for our 15×512 input (seven continuous features plus an 8-dimensional token embedding, Appendix A.6, over a 512-event window). The topology, BL layers feeding a terminal TABL layer, is unchanged (Figure 6).

Each BL layer applies a bilinear transformation independently along the feature and temporal axes,

$$Y = \phi\big(W_1 X W_2 + B\big), \tag{11}$$

where $X \in \mathbb{R}^{D \times T}$, $W_1 \in \mathbb{R}^{D' \times D}$ projects the feature axis, $W_2 \in \mathbb{R}^{T \times T'}$ projects the temporal axis, $B \in \mathbb{R}^{D' \times T'}$ is a learned bias, and ϕ is a ReLU. The final layer augments Equation 11 with a learned temporal attention mechanism (Tran et al., 2018). The feature-axis projection $\bar{X} = W_1 X$ is computed first; a learned $W \in \mathbb{R}^{T \times T}$ then produces attention scores $E = \bar{X}W$, one per pair of time steps, normalised row-wise by a softmax, $\alpha_{ij} = \exp(e_{ij}) / \sum_k \exp(e_{ik})$, into the attention matrix A. This matrix is applied element-wise to $\bar{X}$, and each time step's representation is replaced by a convex combination of its attention-masked and its original value,

$$\tilde{X} = \lambda\big(\bar{X} \odot A\big) + (1 - \lambda)\,\bar{X}, \qquad Y = \phi\big(\tilde{X} W_2 + B\big), \tag{12}$$

with the learned scalar $\lambda \in [0, 1]$ controlling the weight of the attended representation. The diagonal of W is fixed to $1/T$ while its off-diagonal entries are learned, so every time step has the same fixed self-contribution and the learned entries encode the relative importance of one time step with respect to the others.

A.2 Model Parameter Count

Table 4 reports the parameter count of each component. Both variants share the D-TABL encoder, the feature projection and the context/attention trunk; the only architectural difference is the final decoder, which produces two continuous outputs (μ, ν) for the regression variant and three logits for the classification variant.

A.3 Training Strategy

Training proceeds in two stages. First, the D-TABL encoder is pretrained on the three-class classification task alone, with three output logits and the class-weighted cross-entropy of Section 3.4,

Table 4: Parameter count by architectural component.

Component	Parameters
D-TABL encoder (shared)	268,356
Feature projection (shared)	24,704
CNP context / attention trunk (shared)	766,784
Regression decoder	165,378
Classification decoder	165,731
Total (regression variant)	**1,225,222**
Total (classification variant)	**1,225,575**

and no UQ head attached. A separate encoder is trained for every asset–horizon pair on 512-event windows. Training duration per pair varies with the asset's liquidity, since more active assets yield more windows over the same calendar period (Table 6); each encoder is trained for up to 15 epochs of about 15 minutes each on a single NVIDIA GH200 Grace Hopper Superchip, and the checkpoint with the best validation macro F1 is retained.

Second, each UQ head is initialised from the encoder checkpoint of its asset and horizon and fine-tuned end-to-end together with the encoder (Section 4.3). Both variants are trained for up to 15 epochs of about 20 minutes each on the same hardware, the additional cost reflecting the attention-based head. The UQ-regression checkpoint with the best validation WR^2 and the UQ-classification checkpoint with the best validation macro F1 are retained. Across the 21 asset–horizon pairs, base encoder (DTABL) pretraining requires about 78.75 GPU-hours and each UQ-LOB variant a further 105 GPU-hours, for a total of about 289 GPU-hours.

A.4 Hyperparameters

Table 5 lists all hyperparameters. Values that are shared across assets and horizons were fixed a priori or chosen on the validation split of BTC-USD at 5 s and then held fixed for all other pairs.

Table 5: Hyperparameters used for all asset–horizon pairs.

Group	Hyperparameter	Value
Input	Window length L / token vocabulary $\lvert\mathcal{V}\rvert$ / embedding dim.	512 / 439 / 8
	Continuous features / max. depth $L_{\max}$	7 / 10
Labels	Latency d / end-price smoothing k / threshold multiplier α	0.0005s / 10 / 0.25
Head	Context size C / instances per batch	15 / 16
	Projection dim. / attention dim. d_r / attention heads	128 / 256 / 4
	Context encoder / decoder depth (GELU MLP)	3 / 5
Regression	$(\lambda_1, \lambda_2, \lambda_3, \lambda_4)$	(1.0, 0.1, 0.5, 0.5)
	WMAE exponent p / clamp $w_{\max}$	4 / 40
	Directional margin m / calibration bins B / variance floor ϵ	1.0 / 3 / 1×10^{-8}
Optimisation	Optimiser / learning rate (encoder, head) / weight decay	AdamW / (1e-5, 5e-5) / 0.0
	Warmup schedule	LinearLR, 10k steps
	Cosine restart schedule	T_0=15k, T_{mult}=2, $\eta_{\min}=10^{-5}$, γ=0.5
	Max. epochs	15
	Early-stopping metric	WR^2 or macro F1
	Random seed train/test	42 / 999

A.5 Context Construction

Algorithm 1 states the context construction of Section 3.2. For a target window ending at t with horizon h and latency d, the candidate context windows are those whose label horizon has already elapsed, $e_i^c = t_i^c + d + h \leq t$; the C most recent candidates form the context set. The procedure is identical at training, validation and test time, so the test-time predictor never conditions on information unavailable at t.

Algorithm 1 Causal context construction for one target window

Require: ordered windows $\{(\mathbf{X}_j, t_j, y_j)\}$ of one asset; target index τ; horizon h; latency d; context size C

$\mathcal{S} \leftarrow \{\, j < \tau \;:\; t_j + d + h \le t_\tau \,\}$ ▷ windows whose labels are realised by t_τ

$\mathcal{C} \leftarrow$ the C indices in $\mathcal{S}$ with the largest t_j

return context $\{(f_\theta(\mathbf{X}_j), y_j)\}_{j \in \mathcal{C}}$ and target $(f_\theta(\mathbf{X}_\tau), y_\tau)$

A.6 Input Representation

Each input window consists of 8 raw features per event: the 7 continuous features of Section 4.2 (log inter-event time, log tick distance from the opposing best quote, log relative volume, DNFI, its cumulative sums over 50 and 200 events, and QI) and one discrete feature, the event token drawn from a vocabulary of 439 entries. A transformer encoder consumes such tokens directly through an embedding table, but the bilinear structure of D-TABL has no equivalent mechanism and cannot meaningfully operate on a categorical code as if it were a continuous quantity. We therefore embed the token before the bilinear layers: given token id k and an embedding table $E \in \mathbb{R}^{439\times 8}$, we take $e_k = E[k,:] \in \mathbb{R}^8$ and concatenate it with the 7 continuous features at each time step,

$$x_t = \left[\, x_t^{\text{cont}} ;\, e_{k_t} \,\right] \in \mathbb{R}^{15}, \qquad t = 1, \dots, 512, \tag{13}$$

giving a combined input $X \in \mathbb{R}^{15\times 512}$. The embedding table is learned jointly with the rest of the network during encoder pretraining, whose three-class label is defined in Appendix A.7.2.

A.7 Labelling

A.7.1 Regression Labeller

Start price. The model can only condition on information available up to t, and entering a position incurs a small, fixed latency d before it can be realised. We therefore anchor the start price at the first event at or after $t + d$: $t_{\text{start}} = t + d$ and $p_{\text{start}} = p\big(\arg\min_{i:\, t_i \ge t_{\text{start}}} t_i\big)$.

End price. The horizon h is measured from t_{start}: $t_{\text{end}} = t_{\text{start}} + h$. Label statistics are computed for $h \in \{5, 10, 15, 30\}$ s (Table 7); the experiments of the main text use $h \in \{5, 10, 15\}$ s. Using a single tick at t_{end} as the end price is sensitive to single-tick noise, while averaging over the entire horizon, as in DeepLOB (Zhang et al., 2019a) and DeepFolio (Sangadiev et al., 2020), is unsuitable at cryptocurrency tick rates, where a horizon spans thousands of events and block-averaging would wash out genuine displacement. We instead average the $k = 10$ events immediately at or before t_{end},

$$p_{\text{end}} = \frac{1}{k} \sum_{j \in \mathcal{K}} p_j, \qquad \mathcal{K} = \{\text{the } k \text{ events with the largest } t_j \le t_{\text{end}}\}. \tag{14}$$

This never looks past t_{end}; if fewer than k events are available the example is discarded rather than smoothed over a shorter window.

Target. The regression target is the raw tick displacement $y_h(t) = p_{\text{end}} - p_{\text{start}}$.

A.7.2 Classification Labeller

The classification labeller reuses t_{start}, t_{end}, p_{start} and p_{end} exactly as above and thresholds the displacement $y_h(t)$ into $c \in \{\text{down}, \text{up}, \text{stationary}\}$. For each asset and horizon h we compute the standard deviation σ_h^{LR} of the mid-price log-return over h on the training split and set

$$\tau_h^{\text{LR}} = \alpha\, \sigma_h^{\text{LR}}, \tag{15}$$

with $\alpha = 0.25$ throughout. The threshold is converted from log-return space into a tick displacement at the price level of the window, $\delta_h(t) = p_{\text{start}}\big(\exp(\tau_h^{\text{LR}}) - 1\big)$, and

$$c = \begin{cases} 1\ (\text{up}), & y_h(t) \ge \delta_h(t) \\ 0\ (\text{down}), & y_h(t) \le -\delta_h(t) \\ 2\ (\text{stationary}), & \text{otherwise.} \end{cases} \tag{16}$$

Table 6: Dataset characteristics. **Tick**: minimum price increment ($); **Days**: trading days; **Evts (B)**: total events (billions); **Evts/d (M)**: events per day (millions); **Evts/s**: events per second; **IET**: mean inter-event time (ms); **Evts@5s/10s/15s**: expected number of events within a 5/10/15 s horizon (Evts/s $\times$ horizon). Assets are sorted by liquidity (Evts/s).

Sym	Tick	Days	Evts (B)	Evts/d (M)	Evts/s	IET (ms)	Evts@5s	Evts@10s	Evts@15s
BTC	0.10	94	1.254	13.338	156	6.42	780	1560	2340
ETH	0.01	70	0.903	12.899	151	6.61	755	1510	2265
DOGE	1.0e-7	97	1.124	11.587	134	7.46	670	1340	2010
SOL	0.01	78	0.539	6.909	80	12.50	400	800	1200
TAO	0.0001	84	0.577	6.870	80	12.58	400	800	1200
SUI	0.0001	94	0.523	5.565	64	15.52	320	640	960
LTC	0.01	83	0.276	3.326	39	25.97	195	390	585

A.7.3 Comparing Regression and Classification

To evaluate the two heads on a common footing we convert the regression mean μ into a discrete label with the rule of Equation 16, but with a calibrated multiplier k^* on the threshold, $\delta_h^{\mu}(t) = p_{\text{start}}\big(\exp(k^* \tau_h^{\text{LR}}) - 1\big)$. k^* is selected on the held-out calibration half of the validation period by sweeping candidate values and choosing the one whose predicted stationary proportion most closely matches the true one on that split:

$$k^* = \arg\min_k \Big| \Pr[\,|\mu| < \delta_h^{\mu,k}(t)\,] - \Pr[\,|y_h(t)| < \delta_h(t)\,] \Big|. \tag{17}$$

Calibrating k^* against the true class proportions, rather than the classification head's predicted proportions, keeps the comparison independent of any imbalance in the classifier's outputs. The threshold δ_i in Equation 7 is $\delta_h(t)$ of the corresponding window.

B Data and Label Statistics

B.1 Dataset Characteristics

Table 6 summarises the event-level statistics of the dataset of Section 4.1, computed over the train, validation and test splits combined. Tick sizes span six orders of magnitude, from $0.10 for BTC to $\$1\times10^{-7}$ for DOGE, reflecting nominal price levels rather than market activity. Event rate (Evts/s) is used throughout as a proxy for liquidity, and assets are sorted by it from BTC and ETH, the most active, to LTC, the least active. This ordering does not follow total event count or trading days, which depend on the collection duration per asset; event rate normalises for this.

The last three columns translate the event rate into the expected number of events within each horizon. This exposes a substantial disparity in information density at a fixed horizon: at 15 s, a BTC horizon contains about 2,340 events on average against 585 for LTC, a fourfold difference. Since the end-price smoothing window is fixed at $k = 10$ events for every asset, it covers a proportionally larger fraction of the horizon for lower-rate assets such as LTC, SUI and TAO than for BTC, ETH or DOGE.

B.2 Class Balance by Asset and Horizon

Figure 7 reports the three-class proportions for every asset and horizon at $\alpha = 0.25$ across the three splits. Two patterns hold for six of the seven assets (DOGE, SUI, ETH, TAO, LTC, SOL): down and up proportions are closely balanced within each split, reflecting the symmetric threshold, and the stationary proportion falls from 5 s to 15 s (e.g. DOGE: 26%→21%→18% in training) as longer horizons accumulate larger movements that cross the threshold. Train, validation and test proportions of these six assets track each other closely at every horizon, consistent with the modest drift of Section B.3. BTC-USD is the exception: at 5 s the training split assigns roughly 70% of windows to the stationary class but the test split only 27%, the reverse of the usual pattern in which stationary is the minority class. The gap narrows but persists at longer horizons (train stationary 64% at 10 s and 57% at 15 s; test 19% and 15%).

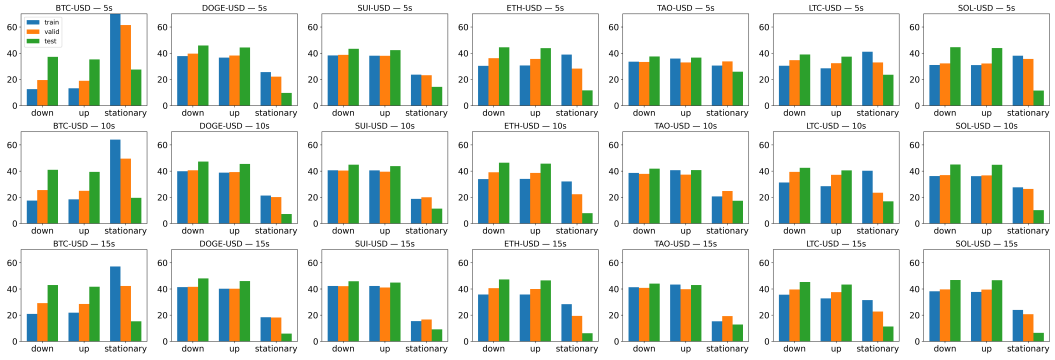


Figure 7: Three-class proportions (train / validation / test) for every asset and horizon at $\alpha = 0.25$.

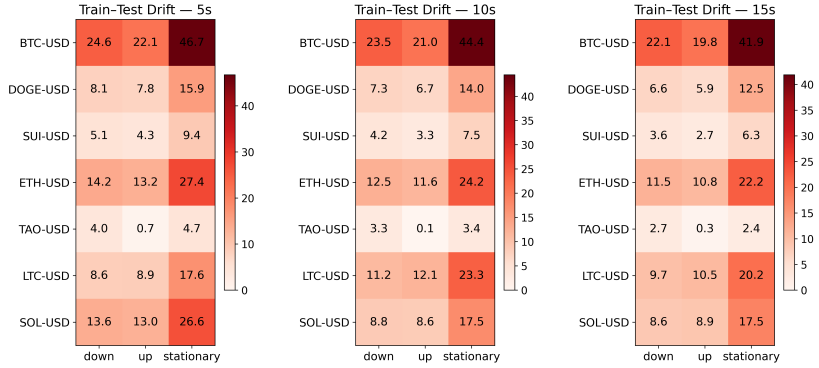


Figure 8: Train–test class proportion drift $D(s, c, h)$ (percentage points) by asset and class, across horizons, $\alpha = 0.25$.

B.3 Class Proportion Drift

Figure 8 reports the absolute difference in true class proportions between the training and test splits, $D(s, c, h) = |P_{\text{train}}(c \mid s, h) - P_{\text{test}}(c \mid s, h)|$, for asset s, class $c \in \{\text{down}, \text{up}, \text{stationary}\}$ and horizon h. Drift varies by asset and shrinks with horizon. TAO-USD and SUI-USD drift least, with stationary-class drift below 10 percentage points at every horizon and 2.4 pp and 6.3 pp respectively at 15 s. ETH-USD and SOL-USD sit in a moderate range, while BTC-USD, the most liquid asset in the dataset, drifts most, reflecting a shift in volatility regime between the training and test windows. This shift is the main source of distribution mismatch in our evaluation, and one motivation for conditioning the head on recently realised outcomes.

B.4 Per-Horizon Threshold and Scale Values

Table 7 reports, per asset and horizon, the log-return volatility σ_h^{LR} and the resulting threshold $\tau_h^{\text{LR}} = \alpha\sigma_h^{\text{LR}}$ used to construct the three-class label (Appendix A.7.2), together with the label scale y_{ref} used to normalise context labels and to scale the predicted variance in the head (Section 3.3). All three quantities are computed independently per asset and horizon on the training split.

C Extended Results

This appendix collects results supporting the gating analysis. Table 8 decomposes three-class macro F1 by class at the 5 s horizon, showing the divergent behaviour of the down, up and stationary classes under SNR gating that motivates directional macro F1. Figure 9 reports three-class macro F1 under softmax gating for the UQ-classification head, and Figure 10 three-class accuracy under SNR and softmax gating for the two heads, the counterparts of the directional macro F1 curves in the main text.

Table 7: Per-asset and per-horizon log-return volatility σ_h^{LR}, label threshold τ_h^{LR}, and label scale y_{ref} (ticks), computed on the training split.

	σ_h^{LR}				τ_h^{LR}				y_{ref}			
Symbol	**5s**	**10s**	**15s**	**30s**	**5s**	**10s**	**15s**	**30s**	**5s**	**10s**	**15s**	**30s**
BTC	1.82e-04	2.63e-04	3.29e-04	4.85e-04	4.50e-05	6.60e-05	8.20e-05	1.21e-04	155	181	205	255
DOGE	3.65e-04	5.12e-04	6.25e-04	8.81e-04	9.10e-05	1.28e-04	1.56e-04	2.20e-04	187	277	352	507
ETH	3.14e-04	4.43e-04	5.42e-04	7.74e-04	7.80e-05	1.11e-04	1.36e-04	1.94e-04	42	56	69	96
LTC	3.70e-04	5.77e-04	6.11e-04	8.94e-04	9.20e-05	1.44e-04	1.53e-04	2.23e-04	2	2	2	3
SOL	3.58e-04	5.10e-04	6.18e-04	8.83e-04	8.90e-05	1.28e-04	1.55e-04	2.21e-04	3	3	4	6
SUI	5.09e-04	7.34e-04	8.79e-04	1.246e-03	1.27e-04	1.83e-04	2.20e-04	3.11e-04	3	5	6	9
TAO	5.01e-04	6.97e-04	8.39e-04	1.176e-03	1.25e-04	1.74e-04	2.10e-04	2.94e-04	628	953	1059	1596

Table 8: Per-class F1 of UQ-regression across SNR tiers, pooled over assets, 5 s horizon. Stationary F1 collapses under gating because confident predictions are large in magnitude, which is why three-class macro F1 is not a suitable gating metric.

Metric	**Top 100%**	**Top 50%**	**Top 30%**	**Top 10%**	**Top 1%**
F1 (Down)	0.447	0.526	0.549	0.598	0.640
F1 (Up)	0.393	0.477	0.499	0.544	0.598
F1 (Stationary)	0.390	0.067	0.012	0.000	0.000
Macro F1	0.410	0.357	0.353	0.381	0.413

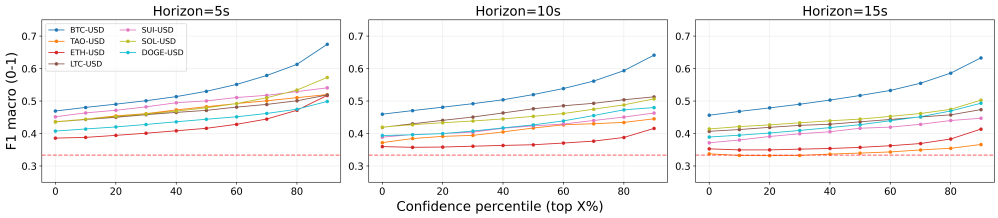


Figure 9: Three-class macro F1 of the UQ-classification head as a function of the softmax-confidence percentile retained, per asset and horizon. The dashed line is the prior-matched random reference.

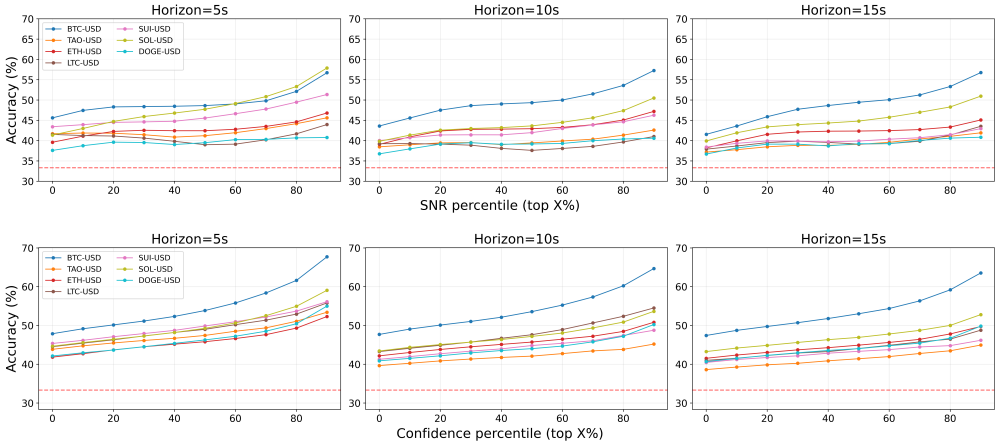


Figure 10: Three-class accuracy as a function of the confidence percentile retained, per asset and horizon. Top: UQ-regression gated by SNR. Bottom: UQ-classification gated by softmax confidence. The dashed line is the chance level.