ETH zürich

Machine Learning for Finance and Complex Systems

Bitcoin volatility forecasting

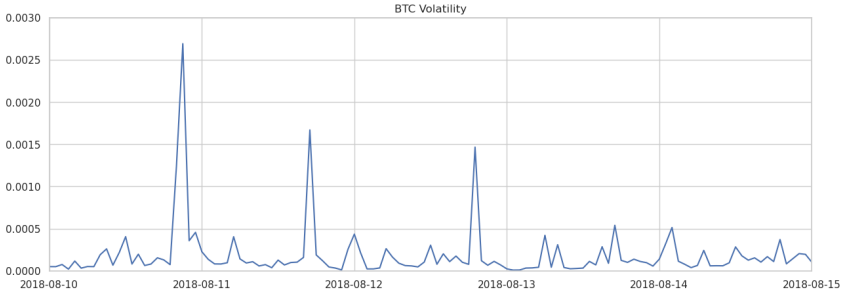
Konrad Ochedzan, Viacheslav Buchkov, Miroslav Zivanovic, Marcus Imris

19 May 2025

Outline

Outline

Volatility Dynamics: intuition and stylized facts



- Clusters of low and high volatility.
- Shocks - reaction to a sudden change in sentiment.
- Decays after shocks

Data Preparation

- **Orderbook Features**:
  1. *Bid-Ask Spread* - signal of market-makers and other agents about microstructure volatility
  2. *Ask & Bid Depths* - how much an agent with large volume can buy/sell
  3. *Depth Difference* - change in skew of buyers versus sellers
  4. *Ask & Bid Volumes* - size of orders from buyers and sellers
  5. *Volume Difference* - overall imbalance of buyers vs sellers
  6. *Weighted Spread* - how much one can shift the market with large volume order to buy versus to sell
  7. *Ask & Bid Slopes* - how much of existing volume one may shift to execute a large order

Dataset

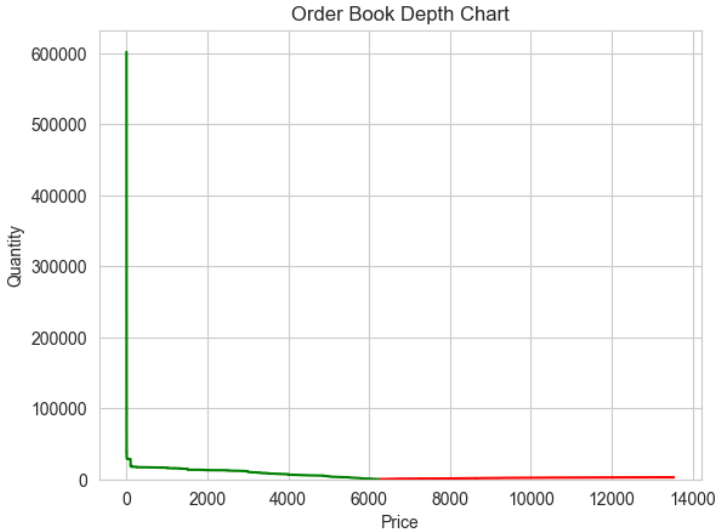
Data: Orderbook + Trades: 2018-06-04 23:00:00 - 2018-09-30 21:00:00 (2579 datapoints, 22 points on average in rolling for testing)
Target: Hourly quadratic variation of the returns on trades within this hour:

$$y = \sum_{t=1}^{T} r_s^2$$

under $s \in [t, T)$ for T - 1 hour and s enumerating the trades.

- **Features**:
  - Compute orderbook features for each snapshot
  - Resample orderbook features for every 30s
  - Add 5min returns in history (for σ-LSTM) to avoid sparsity
  - Append the data up to 30 seconds **before the QV accrual starts**
  - End up with 12 returns + 120 OB features for each target point

Orderbook Initial



Orderbook Cleaned

Outline

Idea behind mixture models

Goal: Improve short-term volatility predictions by dynamically combining insights from historical volatility and order-book data.

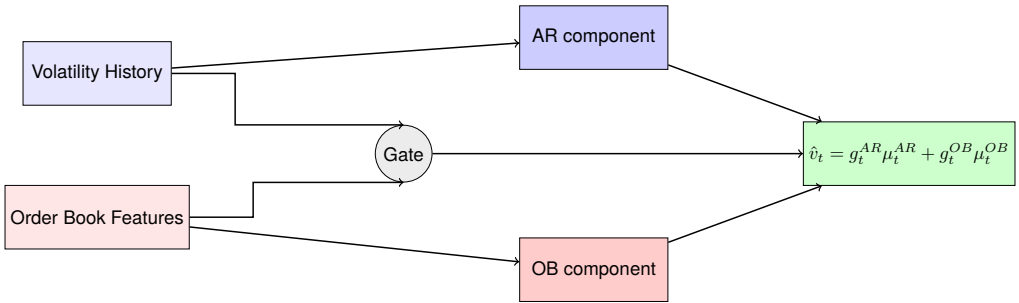
Motivation:

- Volatility exhibits auto-regressive patterns (e.g., clustering).
- Order-book captures fine-grained market sentiment and trading intention.
- Their relative influence can shift over time.

Modeling Insight:

- Use two specialized models:
  - **Auto-regressive component:** captures temporal trends in volatility.
  - **Order-book component:** models market microstructure dynamics.
- A **gate function** learns to weigh each component dynamically.

Idea behind mixture models



Mixture models - theory

Goal: Estimate short-term volatility as a probabilistic mixture of two components:

- Auto-regressive (AR) component: uses past volatility
- Order-book (OB) component: uses order flow features

Key Elements:

- Gating function determines weights $g_t = (g_t^{AR}, g_t^{OB})$
- Each component predicts a distribution: D_t^{AR}, D_t^{OB}

Gates:

$$g_t^{\mathrm{AR}} = \frac{\exp\left(\frac{\alpha^\top \mathbf{v}_{t-p:t-1}}{\tau}\right)}{\exp\left(\frac{\alpha^\top \mathbf{v}_{t-p:t-1}}{\tau}\right) + \exp\left(\frac{s(A_g, B_g, X_t, b_g)}{\tau}\right)}$$

Mixture Prediction:

$$p(v_t | v_{t-p:t-1}, X_t) = g_t^{AR} \cdot f_t^{AR}(v_t) + g_t^{OB} \cdot f_t^{OB}(v_t)$$

$$\mathbb{E}[v_t | v_{t-p:t-1}, X_t] = g_t^{AR} \mu_t^{AR} + g_t^{OB} \mu_t^{OB}$$

Types of mixtures

Implemented Variants of Temporal Mixture Models:

Models from original paper (Baseline):

- **TM-N** *(Normal distribution for both components)*
- **TM-H** *(Normal + hinge penalty for negative means)*
- **TM-LN** *(Log-Normal distribution for both components)*

Additional models (Paper extension):

- **TM-IG** *(Inverse-Gaussian for both components)*
- **TM-W** *(Weibull distribution for both components)*
- **TM-HN-W** *(Normal AR + Weibull OB)*
- **TM-HN-IG** *(Normal AR + Inverse-Gaussian OB)*

TM-N: Normal Component Model

What is TM-N?

- Assumes **normal distributions** for both AR and OB components.
- Final prediction is a **mixture of means** weighted by a learned gate.

Key formulas:

- $\mu_t^{AR} = \phi^\top v_{t-p:t-1}$ *(AR mean)*
- $\sigma_t^{AR} = \exp\left(\frac{1}{2}\gamma^\top v_{t-p:t-1}\right)$ (we estimate log-sigma due to additivity)
- $\mu_t^{OB} = s(A_\mu, B_\mu, X_t, b_\mu)$ *(bilinear mean)*
- $\sigma_t^{OB} = \exp\left(\frac{1}{2}s(A_\sigma, B_\sigma, X_t, b_\sigma)\right)$

Loss function:

$$\mathcal{O}_{\text{TM-N}} = -\mathcal{L}_{\text{TM-N}} + \lambda_2\|\theta\|_2^2, \text{ where } \mathcal{L}_{\text{TM-N}} \text{ is a log-likelihood}$$

Remarks:

- **+** Simple and interpretable baseline
- **–** Risk of negative predictions without penalty

TM-H: Hinge-Normal Component Model

Hinge modification:

$$\mathcal{O}_{\text{hinge}} = \mathcal{O}_{\text{TM-N}} + \beta \cdot [\delta - \mu_t]_+$$

Remarks:

- + Penalizes negative mean predictions from AR component
- + Simple, effective fix without changing the distribution
- – Still lacks natural support for strictly positive values

TM-LN: Log-Normal Component Model (1/2)

What is TM-LN?

- Assumes that both AR and OB components follow **log-normal distributions**.
- Naturally ensures that predicted volatility values are **non-negative**.
- Prediction is made in log-space and exponentiated back to original scale.

Key formulas:

- Let $\ell_t = \log v_t$
- $\ell_t^{AR} \sim \mathcal{N}(\mu_t^{AR}, (\sigma_t^{AR})^2)$
- $\ell_t^{OB} \sim \mathcal{N}(\mu_t^{OB}, (\sigma_t^{OB})^2)$
- AR component parameters:
  - $\mu_t^{AR} = \phi^\top v_{t-p:t-1}$
  - $\sigma_t^{AR} = \exp\left(\frac{1}{2}\gamma^\top v_{t-p:t-1}\right)$
- OB component parameters (bilinear):
  - $\mu_t^{OB} = s(A_\mu, B_\mu, X_t, b_\mu)$
  - $\sigma_t^{OB} = \exp\left(\frac{1}{2}s(A_\sigma, B_\sigma, X_t, b_\sigma)\right)$
- Final prediction: $\hat{v}_t = g_t^{AR} \cdot \exp(\mu_t^{AR} + \frac{(\sigma_t^{AR})^2}{2}) + g_t^{OB} \cdot \exp(\mu_t^{OB} + \frac{(\sigma_t^{OB})^2}{2})$

TM-LN: Log-Normal Component Model (2/2)

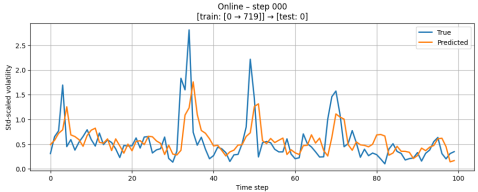
Loss function:

$$\mathcal{O}_{\text{TM-LN}} = -\mathcal{L}_{\text{TM-LN}} + \lambda_2 \|\theta\|_2^2$$

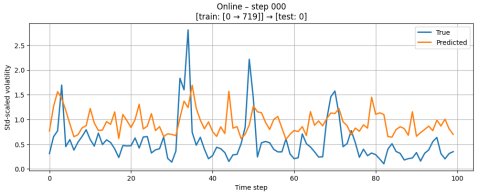
Remarks:

- + Guarantees positive predictions without penalties
- + Better aligned with volatility's distributional properties
- + Good in predicting heavy-tailed behaviours
- – Difficulties predicting low volatility periods due to density approaching 0 in $(0, \varepsilon)$
- – Slightly more complex likelihood computation

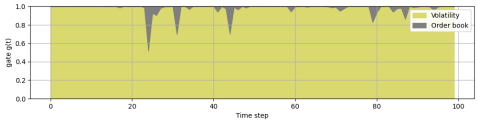
What do we get? (1/2)



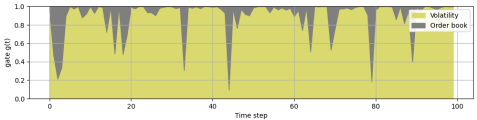
Hinge predictions



Log-normal predictions

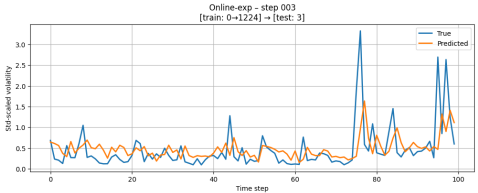


Hinge gate values

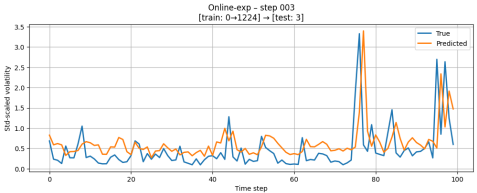


Log-normal gate values

What do we get? (2/2)



Hinge predictions



Log-normal predictions

What more could we do?

- Stay on the positive support of the distribution
- Model heavy tails and non zero density in the neighborhood of 0
- Capture more complex behaviors of volatility
- Ensure the model has enough flexibility to model non standard distributions

TM-W: Weibull Component Model (1/2)

What is TM-W?

- Assumes both AR and OB components follow **Weibull distributions**.
- Captures asymmetric and heavy-tailed behavior in volatility.

Key formulas:

- Weibull density: $f(v;k,\lambda) = \frac{k}{\lambda}\left(\frac{v}{\lambda}\right)^{k-1}\exp\left[-\left(\frac{v}{\lambda}\right)^{k}\right]$
- AR parameters:
  - $k_t^{AR} = \text{softplus}(W_k^{AR} \cdot \text{ReLU}(W_0^{AR} v_{t-p:t-1} + b_0) + b_k)$
  - $\lambda_t^{AR} = \text{softplus}(W_\lambda^{AR} \cdot \text{ReLU}(W_0^{AR} v_{t-p:t-1} + b_0) + b_\lambda)$
- OB parameters:
  - $k_t^{OB} = \text{softplus}(W_k^{OB} \cdot \text{ReLU}(W_0^{OB} \hat{X}_t + b_0) + b_k)$
  - $\lambda_t^{OB} = \text{softplus}(W_\lambda^{OB} \cdot \text{ReLU}(W_0^{OB} \hat{X}_t + b_0) + b_\lambda)$
- Final prediction: $\hat{v}_t = g_t^{AR} \cdot \lambda_t^{AR} \cdot \Gamma\left(1 + \frac{1}{k_t^{AR}}\right) + g_t^{OB} \cdot \lambda_t^{OB} \cdot \Gamma\left(1 + \frac{1}{k_t^{OB}}\right)$

TM-W: Weibull Component Model (2/2)

Loss function:

$$\mathcal{O}_{\text{TM-W}} = -\mathcal{L}_{\text{TM-W}} + \lambda_2 \|\theta\|_2^2$$

Remarks:

- + Flexible for modeling skewed and heavy-tailed distributions
- + All parameters constrained to be positive
- + Dynamics of distribution fit volatility dynamics
- – Harder to interpret and calibrate than Normal-based models
- – Numerically unstable during training

TM-IG: Inverse-Gaussian Component Model (1/2)

What is TM-IG?

- Assumes both AR and OB components follow **Inverse-Gaussian distributions**.
- Designed to model positively skewed, heavy-tailed volatility behavior.

Key formulas:

- IG density: $f(v;\mu,\lambda) = \sqrt{\frac{\lambda}{2\pi v^3}} \exp\left(-\frac{\lambda(v-\mu)^2}{2\mu^2 v}\right)$
- AR parameters:
  - $\mu_t^{AR} = \phi^\top v_{t-p:t-1}$ *(AR mean)*
  - $\lambda_t^{AR} = \text{softplus}(W_\lambda^{AR} \cdot \text{ReLU}(W_0 v_{t-p:t-1} + b_0) + b_\lambda)$
- OB parameters:
  - $\mu_t^{OB} = s(A_\mu, B_\mu, X_t, b_\mu)$ *(bilinear mean)*
  - $\lambda_t^{OB} = \text{softplus}(W_\lambda^{OB} \cdot \text{ReLU}(W_0 \hat{X}_t) + b_\lambda)$
- Final prediction: $\hat{v}_t = g_t^{AR} \cdot \mu_t^{AR} + g_t^{OB} \cdot \mu_t^{OB}$

TM-IG: Inverse-Gaussian Component Model (2/2)

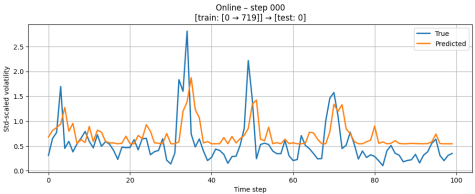
Loss function:

$$\mathcal{O}_{\text{TM-IG}} = -\mathcal{L}_{\text{TM-IG}} + \lambda_2 \|\theta\|_2^2$$

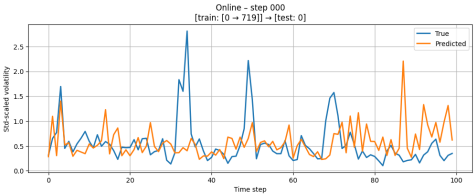
Remarks:

- + Good fit for volatility spikes and heavy-tails
- + Parameters are naturally non-negative
- + Can adapt to local variance asymmetry
- – More numerically sensitive than Gaussian/log-normal
- – More complex loss gradients during training

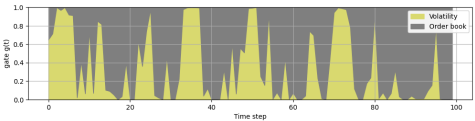
Results and issues



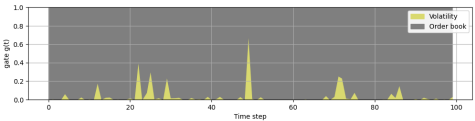
Inverse-Gaussian predictions



Weibull predictions



IG gate values



Weibull gate values

Mixed Models: TM-HN-W and TM-HN-IG (1/2)

What are TM-HN-W and TM-HN-IG?

- **AR:** Normal distribution with hinge penalty to discourage negative means.
- **OB:**
  - TM-HN-W: **Weibull** distribution
  - TM-HN-IG: **Inverse-Gaussian** distribution

Parameter estimation (shared structure):

- **AR (Normal with hinge):**
  - $\mu_t^{AR} = \phi^\top v_{t-p:t-1}$ *(AR mean)*
  - $\sigma_t^{AR} = \text{softplus}(W_\sigma^{AR} \cdot \text{ReLU}(W_0 v_{t-p:t-1} + b_0) + b_\sigma)$
- **OB (Weibull or IG):** Parameters are estimated using simple neural network $\text{softplus}(W_\theta \cdot \text{ReLU}(W_0 \hat{X}_t + b_0) + b_\theta)$

Mixed Models: TM-HN-W and TM-HN-IG (2/2)

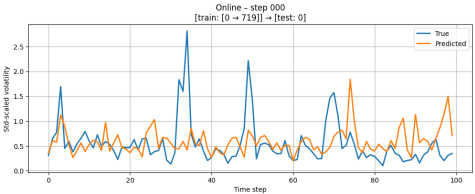
Loss function (shared structure for both models):

$$\mathcal{O}_{\text{Mixed}} = -\mathcal{L}_{\text{mix}} + \lambda_2 \|\theta\|_2^2 + \beta \left[\delta - \mu_t^{AR}\right]_+$$

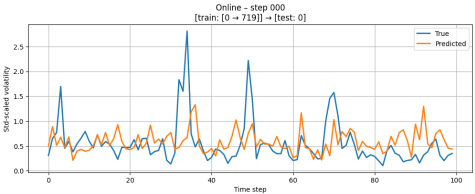
Remarks:

- + AR hinge penalty adds control over negative forecasts
- + OB component is flexible (Weibull or IG) to model volatility spikes
- + IG can model more peaked asymmetry; Weibull better for heavy tails
- – Mixed distributions make optimization more sensitive
- – More tuning required for stable training

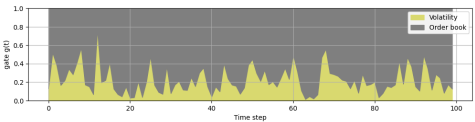
Mixed models results



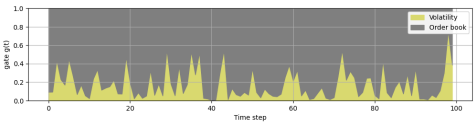
TM-HN-IG predictions



TM-HN-W predictions



Gate values (IG)



Gate values (Weibull)

Results - weekly rolling

Models	**RMSE**		**MAE**	
	Mean	Stddev	Mean	Stddev
TM-N	0.1458	0.1128	0.0869	0.0507
TM-H	0.2039	0.1606	0.1010	0.0561
TM-LN	0.2758	0.3795	0.1176	0.0600
TM-IG	0.2009	0.1508	0.1054	0.0406
TM-W	0.2144	0.1454	0.1002	0.0415
TM-H-W	0.2088	0.1423	0.0966	0.0375
TM-H-IG	0.2165	0.1458	0.0991	0.0367
ARIMAX	0.2279	0.0630	0.0976	0.0182
HARX	0.2254	0.0638	0.0948	0.0191

All values multiplied by 10^{-3}

Results - weekly incremental

Models	RMSE		MAE	
	Mean	Stddev	Mean	Stddev
TM-N	0.1536	0.1085	0.0897	0.0504
TM-H	0.1539	0.1098	0.0899	0.0376
TM-LN	0.1887	0.1451	0.1009	0.0806
TM-IG	0.1640	0.1167	0.0897	0.0319
TM-W	0.1427	0.1031	0.0884	0.0373
TM-H-W	0.1586	0.1088	0.0913	0.0316
TM-H-IG	0.2141	0.1450	0.1011	0.0315
GARCHX	0.2425	0.0536	0.0922	0.0170
HARX	0.2233	0.0630	0.0901	0.0147

All values multiplied by 10^{-3}

Outline

Neural Networks Experiments (1/3)

- **Prior Knowlegde On Volatility**:
  - Volatility is autoregressive and mean-reverting.
  - Order-book captures information from potential future trades that will shift volatility level.
  - However, market exhibits different regimes throughout the dataset.
    $\Rightarrow$ Challenge to use sequential learning throughout the whole dataset.

Neural Networks Experiments (2/3)

- **Prior Knowlegde On Volatility**:
  - Volatility is autoregressive and mean-reverting.
  - Order-book captures information from potential future trades that will shift volatility level.
  - However, market exhibits different regimes throughout the dataset.
    $\Rightarrow$ Challenge to use sequential learning throughout the whole dataset.
- **Outline Of Models**:
  - MLP
  - **LSTM**, σ-LSTM
  - Transformer (Encoder)
  - NBeats (Oreshkin et al. (2020))
  - PatchTST (Nie et al. (2023))
    $\Rightarrow$ Rolling regression to predict 1 week / 1 day out-of-sample.

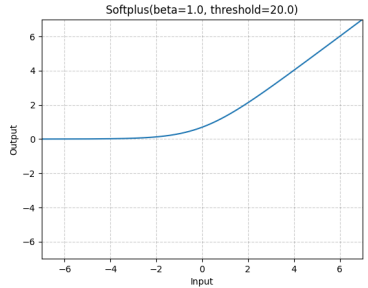
Neural Networks Experiments (3/3)

- **Prior Knowlegde On Volatility**:
  - Volatility is autoregressive and mean-reverting.
  - Order-book captures information from potential future trades that will shift volatility level.
  - However, market exhibits different regimes throughout the dataset.
    $\Rightarrow$ Challenge to use sequential learning throughout the whole dataset.
- **Outline Of Models**:
  - MLP
  - **LSTM**, σ-LSTM
  - Transformer (Encoder)
  - NBeats (Oreshkin et al. (2020))
  - PatchTST (Nie et al. (2023))
    $\Rightarrow$ Rolling regression to predict 1 week / 1 day out-of-sample.
- **Epistemic Uncertainty Modelling** (Our Improvement):
  - Apply Bayesian Learning (Bayes by Backprop / Variational Dropout)
    $\Rightarrow$ Use **uncertainty** of the model estimation to catch structural breaks / balance model point estimates versus baseline.

Softplus Activation

- Need to squash our final layer outputs into $\mathbb{R}^+$
- We use *Softplus* across all models
- It is a primitve of a sigmoid and is a convex function
- However, our objective after NN is not necessarily convex, as *Sofplus* is not a convex conjucate to *MSE* (composition $\implies$ need to have outer function convex and *non-decreasing*)
  $\Rightarrow$ We see improved results, compared to square function in the output.

Figure: Source = Pytorch Library

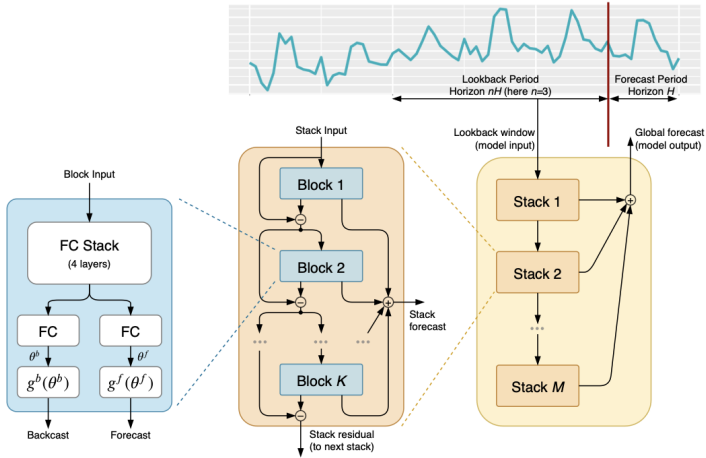


NBeats Model

- The model Oreshkin et al. (2020) aims to solve the vanishing gradient of RNNs by **stacking blocks of MLPs**
- The distinct feature of this model is that at each point it forecasts (future) and **backcasts (past values)**, attempting to reconstruct the time series
- It tries to **repeat the idea of CNNs** - each block is aimed to learn some specific pattern of the time series independently
- Why is it useful? It **allows for pre-specified horizon for prediction instead of recursive updates**
- We train the model on the same set of features as all other NNs

NBeats Architecture

Figure: Source = Oreshkin et al. (2020)

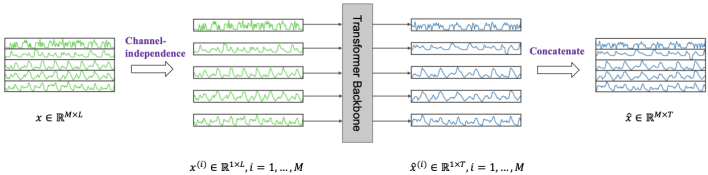


PatchTST Model

- The model Nie et al. (2023) tries to improve the logic of simple Transformer Encoder, treating 'Time Series as sequences of words'
- It **splits the Time Series into chunks** and weighs by attention weights
- Then the resulting patters are combined for the output
- Why is it useful? It generalizes our Transformer architecture

PatchTST Architecture

Figure: Source = Oreshkin et al. (2020)



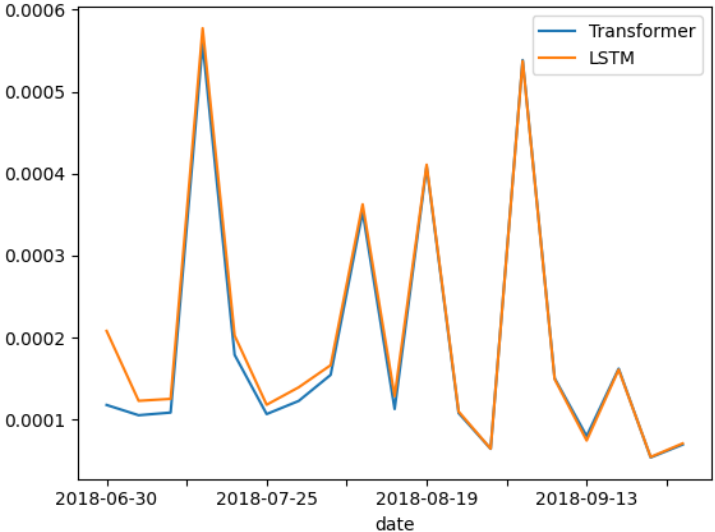
NN Results

Table: RMSE Results (20 epochs).

Model	Expanding Window		Fixed Window	
	Mean	Std	Mean	Std
MLP	0.223	0.154	43900.00	180480.4
LSTM	0.154	0.180	0.154	0.180
Transformer	0.147	0.181	901.5	2535.6%
σ-LSTM returns	26.484	10.737	26.484	10.737
σ-LSTM with features	32.062	10.737	32.062	10.737
NBeats	39.262	34.937	N/A	N/A
PatchTST	77.848	0.7840	N/A	N/A

All values multiplied by 10^{-3}

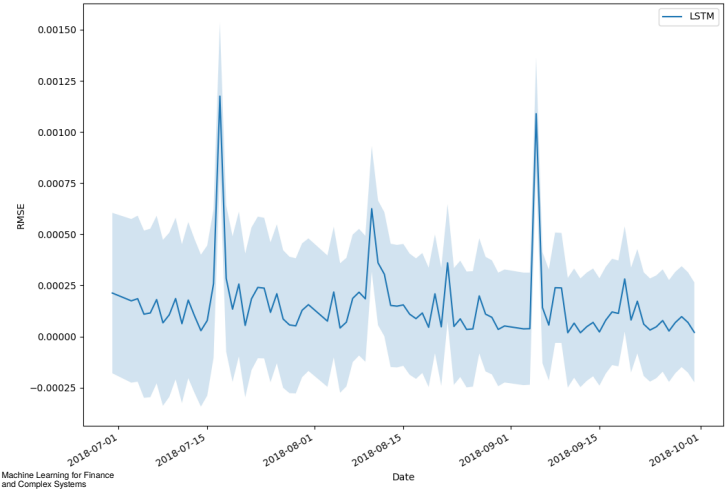
Expanding Window Comparison



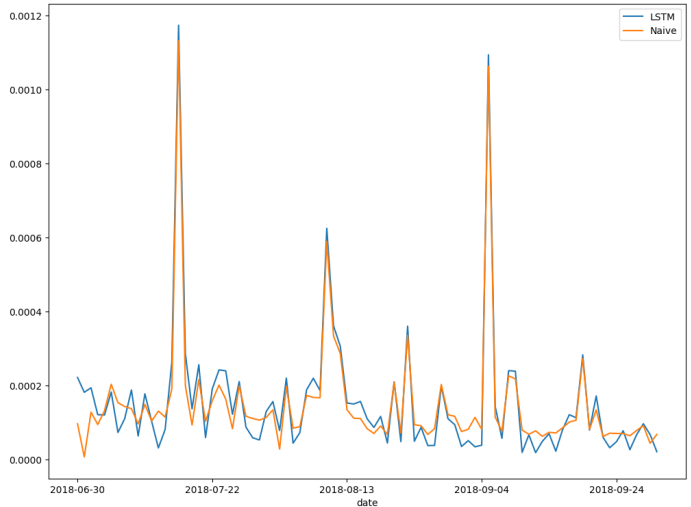
Uncertainty Quantification

- Due to structural breaks in financial data (agents behavior) we need to track, **how relevant our model is for the new data**
- One approach is **treat prediction task not as point prediction, but as distribution**
- We focus on Variational Dropout (but results still hold for Bayes by Backprop)
- The Dropout layers now are **not turned off** for evaluation
- We **approximate epistemic uncertainty** (defined as predictive entropy) by running model inference E times and returning mean and standard deviation - randomness comes from Dropout
  $\Rightarrow$ Obtain uncertainty quantification for each prediction point.

Uncertainty Quantification



Using The Uncertainty



Using The Uncertainty

- We can see that when there is a large change in the volatility, our model captures it well.
- Therefore, part of underperformance may come from **incorrect average level of volatility**, when there are **some structural breaks**
- Our solution - create a dynamic ensemble of LSTM and Naive Predictor (Average Historical Volatility)
- We train LSTM and Naive Predictor (both separately and jointly - i.e. doing the proposed ensemble on train too)
- We combine predictions as following, given uncertainty $u_t = std(\hat{r}_t)$:

$$w_{LSTM} = \frac{u_t - \min u_{\tau:t-1}}{\max u_{\tau:t-1} - \min u_{\tau:t-1}}$$

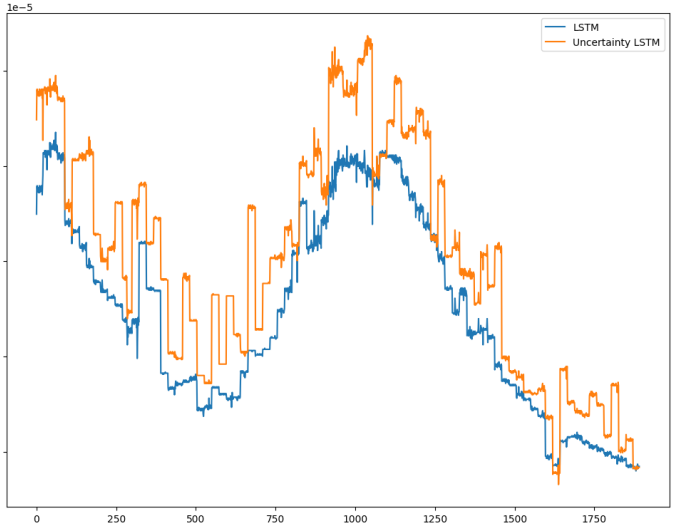
  for τ - start of training. Note that

$$w_{LSTM} \in [0, 1]$$

- We output prediction as convex combination

$$\tilde{r}_t = w_{LSTM} r_t^{LSTM} + (1 - w_{LSTM}) r_t^{Naive}$$

Using The Uncertainty



Using The Uncertainty

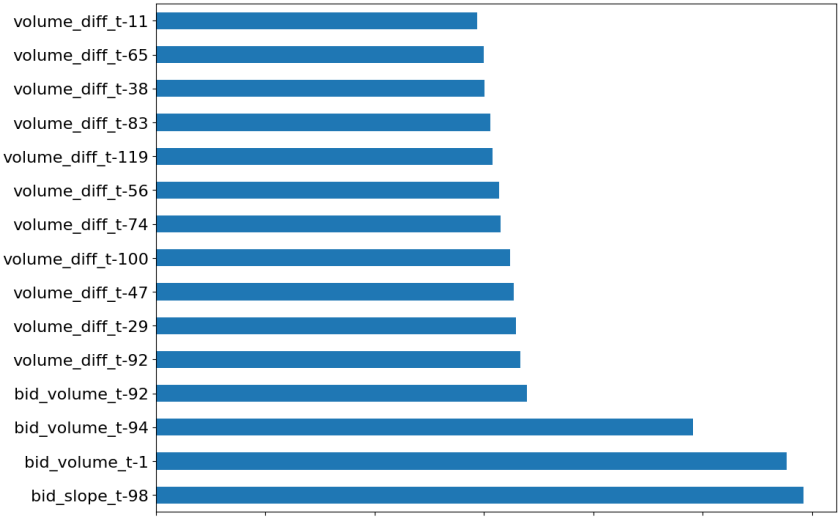
- Another use of uncertainty is based on retraining frequency
- At each point of retraining we have a trade-off - **potentially new patterns in the new data versus more data with 'old' patterns**
- We can make use of the uncertainty in this case too
- At each rolling decision point we look at u_t and compare it versus $\overline{u}_{\tau:t-1} + z\ std(u_{\tau:t-1})$
- If the uncertainty is **above upper bound of this credible interval**, we retrain the model

Uncertainty Results

Table: RMSE Results (20 epochs).

Model	Fixed Window	
	Mean	Std
LSTM	0.1540567	0.1809012
LSTM Trained Ensemble	0.1510132	0.1802456
LSTM Inference Ensemble	0.1511618	0.1801922
LSTM Uncert Retrain	0.1514191	0.1800430

Feature Importances by Feature Ablation



Outline

Results - weekly rolling

Models	**RMSE**		**MAE**	
	Mean	Stddev	Mean	Stddev
TM-H	0.2039	0.1606	0.1010	0.0561
TM-IG	0.2009	0.1508	0.1054	0.0406
TM-W	0.2144	0.1454	0.1002	0.0415
TM-H-W	0.2088	0.1423	0.0966	0.0375
ARIMAX	0.2279	0.0630	0.0976	0.0182
HARX	0.2254	0.0638	0.0948	0.0191
LSTM	0.1541	0.1801	0.0845	0.0812
LSTM Inference Ensemble	**0.1511**	0.1801	**0.0831**	0.0753

All values multiplied by 10^{-3}

Results - weekly expanding

Models	**RMSE**		**MAE**	
	Mean	Stddev	Mean	Stddev
TM-H	0.1539	0.1098	0.0899	0.0376
TM-IG	0.1640	0.1167	0.0897	0.0319
TM-W	**0.1427**	0.1031	0.0884	0.0373
TM-H-W	0.1586	0.1088	0.0913	0.0316
GARCHX	0.2425	0.0536	0.0922	0.0170
HARX	0.2233	0.0630	0.0901	0.0147
Transformer	0.1470	0.1812	**0.0797**	0.0578
LSTM Inference Ensemble	0.1511	0.1801	0.0831	0.0753

All values multiplied by 10^{-3}

Outline

Results - weekly rolling

Models	**RMSE**		**MAE**	
	Mean	Stddev	Mean	Stddev
Naive	0.2729	0.0811	0.0965	0.0168
EWMA	0.2396	0.0671	0.0961	0.0157
ARIMA	0.2247	0.0626	0.0957	0.0115
GARCH*	97.7101	14.39833	37.8156	55.7073
HAR	0.2224	0.0623	0.0923	0.0116
ARIMAX	0.2279	0.0630	0.0976	0.0182
HARX	0.2254	0.0638	0.0948	0.0191
ElasticNet	0.2366	0.0648	0.1058	0.0144
RandomForest	0.2836	0.0761	0.1475	0.0494
XGBoost	0.2371	0.0568	0.1126	0.0086
STR**	0.1847	0.1342	0.0766	0.0338

*All values multiplied by 10^{-3}; * - convergence issues; ** to be double checked*

Results - weekly expanding

Models	**RMSE**		**MAE**	
	Mean	Stddev	Mean	Stddev
Naive	0.2729	0.0811	0.0965	0.0168
EWMA	0.2396	0.0671	0.0961	0.0157
ARIMA	0.2256	0.0630	0.0999	0.0137
GARCH	0.2425	0.0536	0.0922	0.0170
HAR	0.2236	0.0628	0.0947	0.0130
ARIMAX	0.2254	0.0620	0.0924	0.0147
HARX	0.2233	0.0630	0.0901	0.0147
ElasticNet	0.2372	0.0647	0.1117	0.0166
RandomForest	0.2487	0.0541	0.1202	0.0189
XGBoost	0.2389	0.0573	0.1193	0.0079
STR**	0.1841	0.1332	0.0753	0.0341

*All values multiplied by 10^{-3}; ** to be double checked*

Appendix

Rolling

MAE	0	1	2	3	4	5	6	7	8	9
TM-N	.000053	.0001	.000061	.000068	.000141	.000124	.000051	.00019	.000029	.000052
TM-H	.000052	.000092	.000059	.000071	.000135	.000151	.000082	.000231	.000059	.000078
TM-LN	.0001	105610.17	.000081	.000084	.000154	.000254	.000095	7.770351	.000083	.00009
TM-IG	.000062	.000112	.000079	.000091	.000154	.000152	.000091	.000172	.000057	.000084
TM-W	.000071	.000115	.000083	.000068	.000138	.00015	.000076	.000172	.000043	.000086
TM-HW	.000067	.000121	.000085	.000081	.000122	.000134	.000083	.000165	.000043	.000065
TM-HIG	.000071	.000113	.00009	.000092	.000126	.000147	.000065	.000162	.000052	.000073
best	TM-H	TM-H	TM-H	TM-H	TM-N	TM-IG	TM-N	TM-HIG	TM-N	TM-N
RMSE	0	1	2	3	4	5	6	7	8	9
TM-N	.000081	.000165	.000107	.000107	.00024	.000426	.000122	.000052	.000053	.000105
TM-H	.000081	.000155	.000104	.000123	.000236	.000443	.000132	.000542	.00007	.000153
TM-LN	.000096	10707150.55	.000115	.000122	.000242	.001207	.000151	7.849853	.000093	.00018
TM-IG	.000084	.000138	.000112	.000124	.00025	.000429	.000131	.000511	.000077	.000153
TM-W	.00011	.000202	.000138	.000144	.000241	.000431	.000139	.000516	.000073	.00015
TM-HW	.000098	.000189	.00014	.000137	.000226	.000434	.000128	.000498	.000085	.000153
TM-HIG	.000107	.000177	.000157	.000148	.000249	.000437	.000127	.000519	.000087	.000157
best	TM-H	TM-H	TM-H	TM-H	TM-N	TM-IG	TM-N	TM-HW	TM-N	TM-N

Appendix

Incremental

MAE	0	1	2	3	4	5	6	7	8	9
TM-N	.000054	.000098	.000065	.000075	.000139	.000126	.000053	.000196	.000035	.000056
TM-H	.000055	.000105	.000084	.000073	.000127	.000134	.00005	.000184	.000032	.000055
TM-LN	.00009	.000326	.000088	.000082	.000264	.000186	.000074	18936752.439128	.000049	.00006
TM-IG	.000075	.0001	.000074	.000088	.000137	.000142	.00007	.000096	.000044	.000083
TM-W	.000062	.014216	.000083	.000064	.000125	.000144	.000068	.000183	.00005	.000086
TM-HW	.000072	.000114	.000083	.000074	.000118	.000134	.000049	.000165	.000045	.000059
TM-HIG	.000071	.000122	.000091	.000081	.000113	.000148	.00008	.000174	.000053	.000078
best	TM-N	TM-N	TM-H	TM-H	TM-IG	TM-IG	TM-N	TM-HW	TM-W	TM-H
RMSE	0	1	2	3	4	5	6	7	8	9
TM-N	.000081	.000164	.000107	.000106	.000231	.000429	.000124	.00009	.000056	.000148
TM-H	.000083	.000169	.000105	.000109	.000236	.000431	.000124	.000073	.000065	.000144
TM-LN	.000078	.0003	.000108	.000122	.000244	.000518	.000125	193107507.326705	.000067	.000137
TM-IG	.000092	.00017	.000106	.000125	.000261	.000456	.000129	.000087	.000061	.000153
TM-W	.00013	.141258	.000124	.000143	.00025	.000444	.000125	.000084	.000086	.000149
TM-HW	.000095	.000181	.000128	.000122	.000242	.000428	.000122	.000065	.000059	.000145
TM-HIG	.000106	.000188	.000133	.00014	.000231	.000424	.000143	.000523	.000078	.000175
best	TM-N	TM-N	TM-H	TM-H	TM-IG	TM-IG	TM-N	TM-HW	TM-W	TM-H

Appendix

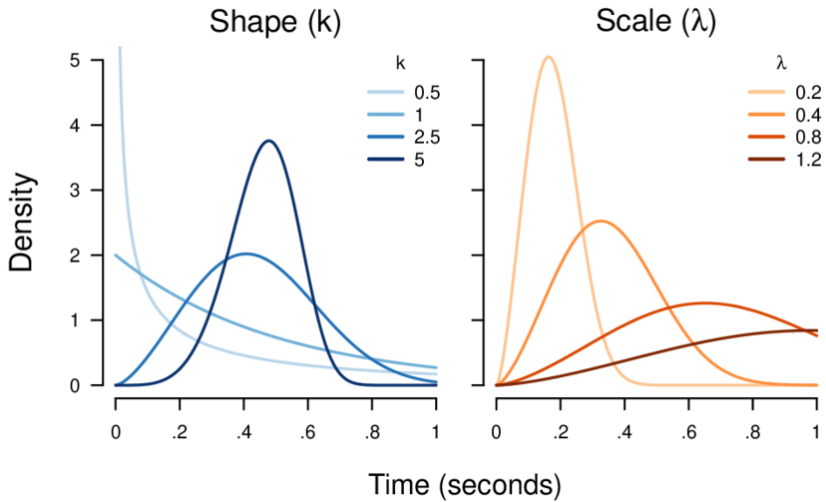
For gating and feature component we define a bilinear score. For $A \in \mathbb{R}^n$, $B \in \mathbb{R}^\ell$, $X \in \mathbb{R}^{n \times \ell}$, define:

$$s(A, B, X, b) = \frac{1}{\sqrt{\ell}} \sum_{i=1}^{n} \sum_{j=1}^{\ell} A_i B_j X_{ij} + b,$$

where b stands for bias. We perform standardization by the number of features' time steps used for the model as it helps to prevent one of the gate weights dominating the other one and leads to more stable solutions.

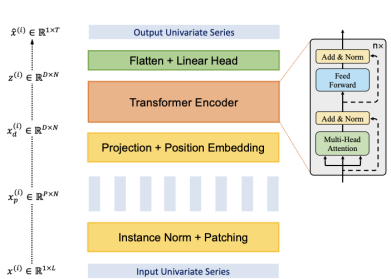
$$\mathcal{L}_{model} = \sum_{t=1}^{T} \log \left[g_t^{AR} \, f_{model}^{AR}(v_t) + g_t^{OB} \, f_{model}^{OB}(v_t) \right]$$

Appendix

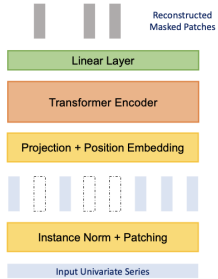

Shape (k)
Scale (λ)
Density
Time (seconds)
k
0.5
1
2.5
5
λ
0.2
0.4
0.8
1.2
0
1
2
3
4
5
0
.2
.4
.6
.8
1

Full PatchTST Architecture

Figure: Source: Oreshkin et al. (2020)



(b) Transformer Backbone (Supervised)



(c) Transformer Backbone (Self-supervised)