Accurate Multivariate Stock Movement Prediction via Data-Axis Transformer with Multi-Level Contexts

Jaemin Yoo[1], **Yejun Soun**[12], **Yong-chan Park**[1], and **U Kang**[12]

[1] Seoul National University

[2] DeepTrade, Inc.

KDD 2021

Outline

- **Introduction**
- Proposed Method
- Experiments
- Conclusion

Stock Market Prediction

- To predict the future values of stock prices
  - The most popular task in the financial domain
- The problem is challenging but rewarding
  - **Challenging.** Stock prices have no clear patterns and make random movements
  - **Rewarding.** Even the increase of 1% of prediction accuracy results in enormous profit

Problem Definition

- **Stock movement prediction**
  - model the problem as binary classification
  - To predict a *movement* rather than exact price
- **Definition**
  - **Given** the historical prices $\{p_t\}_{t \leq T}$ of a stock
    - assume daily prices; p_t is the price at day t
    - T is the current index for prediction
  - **Predict** the rise ($p_{T+1} > p_T$) or fall ($p_{T+1} \leq p_T$)

Research Motivation

- Stocks are strongly correlated with each other
- Previous models can be categorized as
  - **Univariate models**
    - They treat each stock independently from the others
  - **Multivariate models with fixed correlations**
    - They take the correlations as a predefined input
    - The correlations cannot reflect the dynamic property

How can we learn dynamic correlations between stocks with no prior knowledge?

Outline

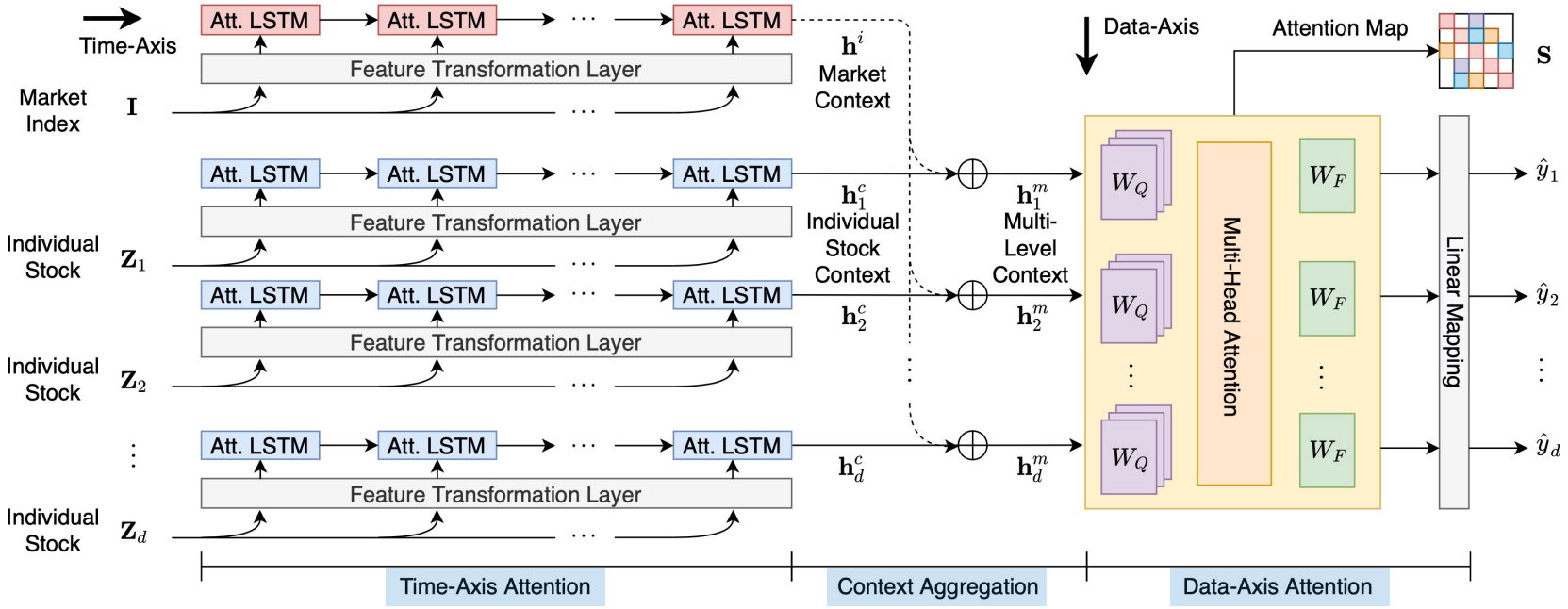
- Introduction
- **Proposed Method**
- Experiments
- Conclusion

Overview (1)

- propose **DTML** for stock price prediction
  - *Data-axis Transformer with Multi-Level contexts*
- **Idea 1.** Time-axis attention
  - To summarize the historical prices of each stock
- **Idea 2.** Context aggregation
  - To combine individual contexts with a global trend
- **Idea 3.** Data-axis attention
  - To learn the stock correlations by a transformer

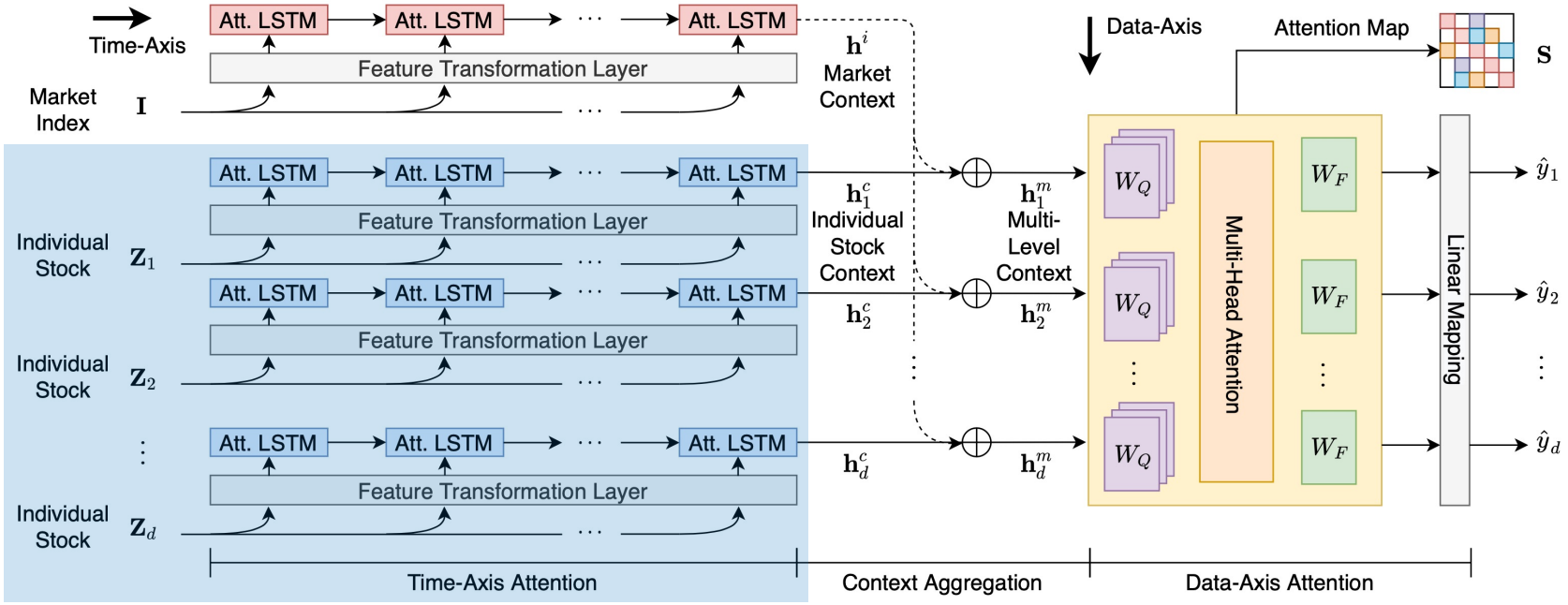
Overview (2)

- This is the overall structure of DTML
  - Three modules correspond to the three main ideas



Time-Axis Attention (1)

- **Module 1. Time-axis attention**
  - Summarizes the historical prices as a single vector



Time-Axis Attention (2)

- **Given** feature vectors $\{\mathbf{z}_{ut}\}_{t \le T}$ of stock u
  - $\mathbf{z}_{ut}$ is made from the prices of stock u until day t
- **Feature transformation**
  - We transform each feature vector as follows:

$$\tilde{\mathbf{z}}_{ut} = \tanh(\mathbf{W}_s \mathbf{z}_{ut} + \mathbf{b}_s),$$

  - $\mathbf{W}_s$ and $\mathbf{b}_s$ are learnable weight and bias, resp.

Time-Axis Attention (3)

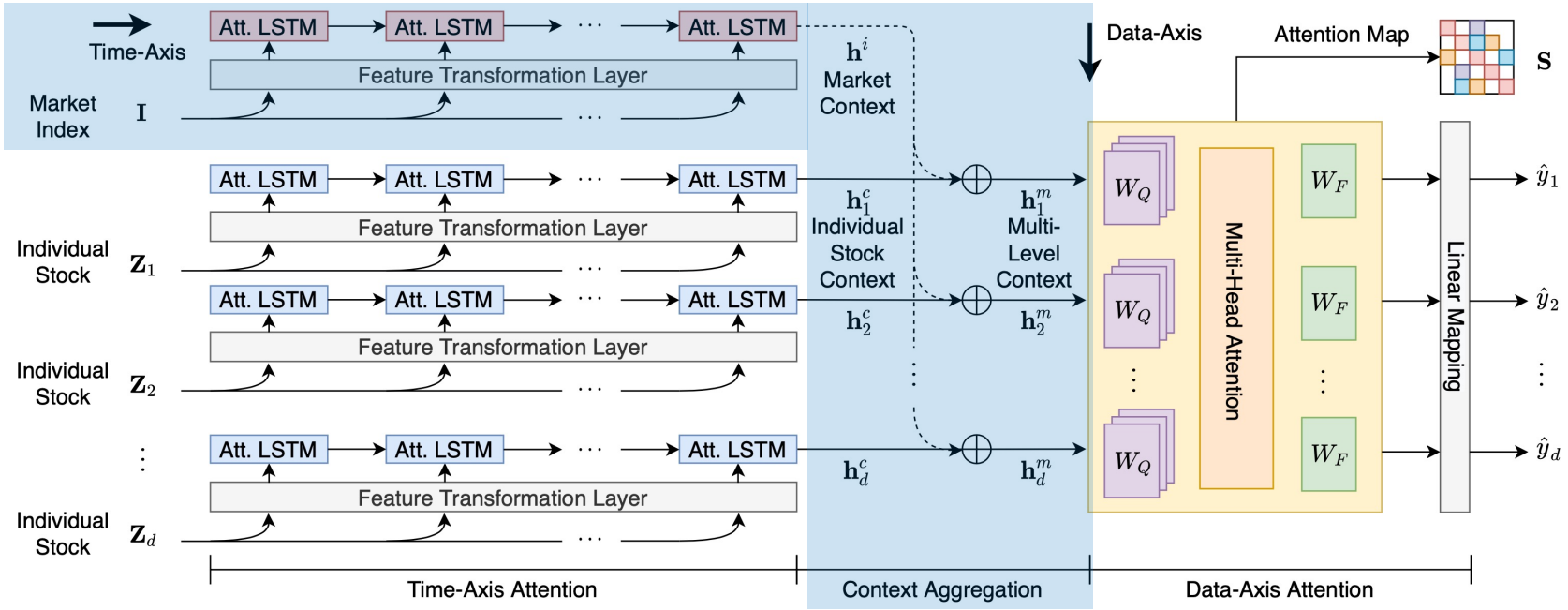
- **Attention LSTM**
  - We run LSTM to generate state vectors $\{\mathbf{h}_{ut}\}_t$
  - We then compute an attention score α_i such that

$$\alpha_i = \frac{\exp(\mathbf{h}_i^\top \mathbf{h}_T)}{\sum_{j=1}^{T} \exp(\mathbf{h}_j^\top \mathbf{h}_T)}.$$

  - The state vectors are combined as $\tilde{\mathbf{h}}^c = \sum_i \alpha_i \mathbf{h}_{ui}$

Context Aggregation (1)

- **Module 2. Context aggregation**
  - Combines individual and global context vectors



Context Aggregation (2)

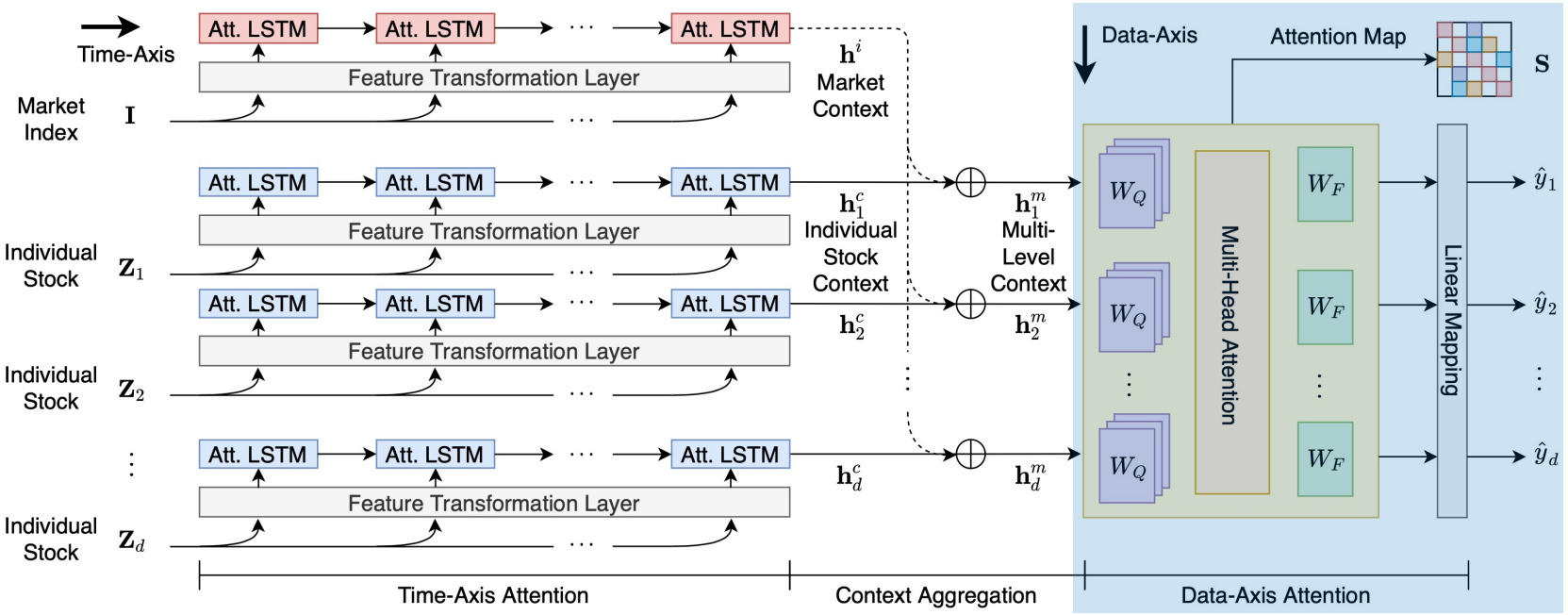
- **Prepare** a global market context $\mathbf{h}^i$
  - Consider a market index i as an individual stock
    - Such as NDX100 or DJI in the US stock markets
  - Apply the time-axis attention to i and make $\mathbf{h}^i$
- **Make** a multi-level context for each stock u:

$$\mathbf{h}_u^m = \mathbf{h}_u^c + \beta \mathbf{h}^i,$$

  - β is a parameter for balancing the two contexts

Data-Axis Attention (1)

- **Module 3. Data-axis attention**
  - Computes dynamic correlations between stocks



Data-Axis Attention (2)

- **Given**
  - Matrix $\mathbf{H} \in \mathbb{R}^{n \times d}$ that stacks multi-level contexts
    - n is the number of stocks, and d is the context size
- **Make** attention components $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$ as

$$\mathbf{Q} = \mathbf{H}\mathbf{W}_{\mathrm{q}} \qquad \mathbf{K} = \mathbf{H}\mathbf{W}_{\mathrm{k}} \qquad \mathbf{V} = \mathbf{H}\mathbf{W}_{\mathrm{v}}.$$

- **Aggregate** the context vectors as follows:

$$\tilde{\mathbf{H}} = \mathbf{S}\mathbf{V} \quad \text{where} \quad \mathbf{S} = \mathrm{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^{\top}}{\sqrt{h}}\right).$$

Data-Axis Attention (3)

- We transform the aggregated contexts $\tilde{\mathbf{H}}$ with
  - **Nonlinear transformation**

$$\mathbf{H}_\mathrm{p} = \tanh(\mathbf{H} + \tilde{\mathbf{H}} + \mathrm{MLP}(\mathbf{H} + \tilde{\mathbf{H}})),$$

  - **Final prediction layer**

$$\hat{\mathbf{y}} = \sigma(\mathbf{H}_\mathrm{p}\mathbf{W}_\mathrm{p} + \mathbf{b}_\mathrm{p}).$$

    - σ is the logistic sigmoid function for an output

Summary

- DTML is a combination of three modules
  - **Time-axis attention** for the prices of each stock
  - **Context aggregation** with a global market trend
  - **Data-axis attention** by a transformer encoder
- DTML is trained by a gradient-based way
  - To minimize the cross-entropy loss $\mathcal{L}$ for training
  - Apply the L2 regularizer only to the last predictor
    - **Why?** To restrict the output space while not affecting the main functionality for making stock correlations

Outline

- Introduction
- Proposed Method
- **Experiments**
- Conclusion

Datasets

- We use six datasets in four different countries
  - Two are public datasets used in previous work
  - Four are private datasets collected in this work

Dataset	Country	Stocks	Days	From	To
ACL18[1]	US	87	504	2014-01-01	2015-12-31
KDD17[1]	US	50	2,518	2007-01-01	2016-12-31
NDX100	US	95	1,259	2013-01-01	2017-12-31
CSI300	China	219	1,119	2015-06-01	2019-12-31
NI225	Japan	51	856	2016-07-01	2019-12-31
FTSE100	UK	24	1,134	2014-01-01	2018-06-30

Feature Engineering

- We use the same features as in previous work
  - Each feature $\mathbf{z}_{ut}$ summarizes the price movement of stock u until day t by simple operations

Features	Calculation
z_{open}	$z_{\text{open}} = \text{open}_t / \text{close}_t - 1$
z_{high}	$z_{\text{high}} = \text{high}_t / \text{close}_t - 1$
z_{low}	$z_{\text{low}} = \text{low}_t / \text{close}_t - 1$
z_{close}	$z_{\text{close}} = \text{close}_t / \text{close}_{t-1} - 1$
$z_{\text{adj\_close}}$	$z_{\text{adj\_close}} = \text{adj\_close}_t / \text{adj\_close}_{t-1} - 1$
z_{d5}, z_{d10} z_{d15}, z_{d20} z_{d25}, z_{d30}	e.g., $z_{dk} = \frac{\sum_{i=0}^{k} \text{adj\_close}_{t-i}}{k \cdot \text{adj\_close}_t} - 1$

Evaluation

- We split each dataset into train/valid/test sets
  - The split is done by the chronological order
  - The splitting dates are the same as in prev. work
- We use two evaluation metrics
  - Simple accuracy (ACC)
    - The number of correct predictions over all predictions
  - The Matthews correlation coefficient (MCC)
    - Measures the accuracy in a more balanced manner

Questions

- We answer the questions by experiments:
  - **Q1 (Accuracy).** Does DTML outperform previous models in terms of classification accuracy?
  - **Q2 (Profit).** Does DTML make the profit on actual investment simulation in our datasets?
  - **Q3 (Correlations).** Does DTML make reasonable correlations between stocks?
  - **Q4 (Ablation study).** Does each module of DTML help improving the classification accuracy?

Q1. Prediction Accuracy

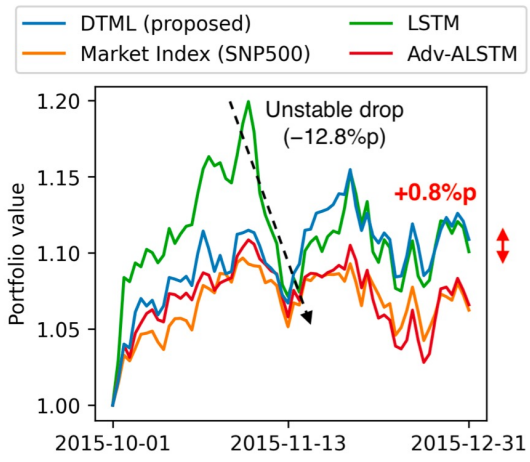
- DTML produces the highest ACC and MCC
  - The improvement is more significant with MCC

Model	ACL18 (US)		KDD17 (US)		NDX100 (US)	
	ACC	MCC	ACC	MCC	ACC	MCC
LSTM [24]	0.4987 ± 0.0127	0.0337 ± 0.0398	0.5118 ± 0.0066	0.0187 ± 0.0110	0.5263 ± 0.0003	0.0037 ± 0.0049
ALSTM [31]	0.4919 ± 0.0142	0.0142 ± 0.0275	0.5166 ± 0.0041	0.0316 ± 0.0119	0.5260 ± 0.0007	0.0028 ± 0.0084
StockNet [31]	0.5285 ± 0.0020	0.0187 ± 0.0011	0.5193 ± 0.0001	0.0335 ± 0.0050	0.5392 ± 0.0016	0.0253 ± 0.0102
Adv-ALSTM [9]	0.5380 ± 0.0177	0.0830 ± 0.0353	0.5169 ± 0.0058	0.0333 ± 0.0137	0.5404 ± 0.0003	0.0046 ± 0.0090
DTML (proposed)	**0.5744 ± 0.0194**	**0.1910 ± 0.0315**	**0.5353 ± 0.0075**	**0.0733 ± 0.0195**	**0.5406 ± 0.0037**	**0.0310 ± 0.0193**

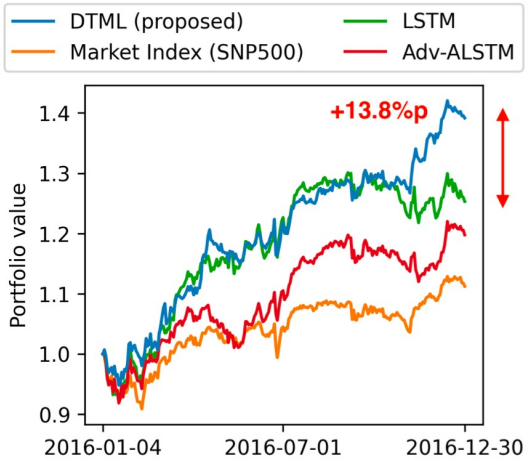
Model	CSI300 (China)		NI225 (Japan)		FTSE100 (UK)	
	ACC	MCC	ACC	MCC	ACC	MCC
LSTM [24]	0.5367 ± 0.0038	0.0722 ± 0.0050	0.5079 ± 0.0079	0.0148 ± 0.0162	0.5096 ± 0.0065	0.0187 ± 0.0129
ALSTM [31]	0.5315 ± 0.0036	0.0625 ± 0.0076	0.5060 ± 0.0066	0.0125 ± 0.0139	0.5106 ± 0.0038	0.0231 ± 0.0077
StockNet [31]	0.5254 ± 0.0029	0.0445 ± 0.0117	0.5015 ± 0.0054	0.0050 ± 0.0118	0.5036 ± 0.0095	0.0134 ± 0.0135
Adv-ALSTM [9]	0.5337 ± 0.0050	0.0668 ± 0.0084	0.5160 ± 0.0103	0.0340 ± 0.0201	0.5066 ± 0.0067	0.0155 ± 0.0140
DTML (proposed)	**0.5442 ± 0.0035**	**0.0826 ± 0.0074**	**0.5276 ± 0.0103**	**0.0626 ± 0.0230**	**0.5208 ± 0.0121**	**0.0502 ± 0.0214**

Q2. Investment Simulation

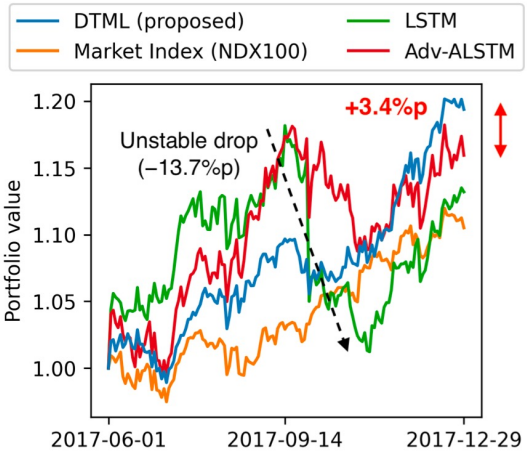
- DTML makes larger profit than the baselines
  - DTML does not suffer from unstable drops
  - The trend is the same as in the other datasets



(a) ACL18 (US)



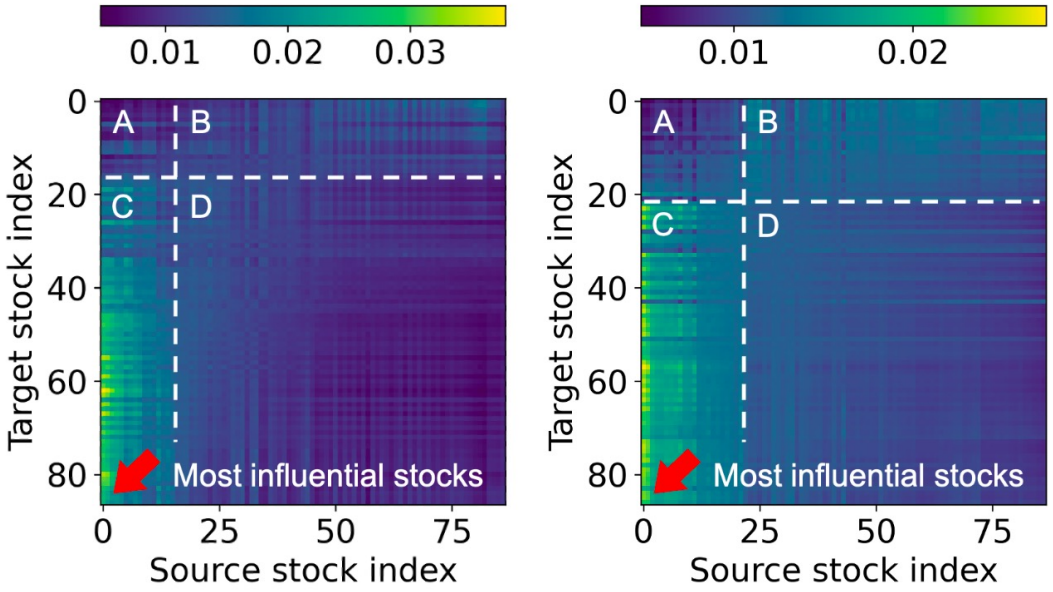
(b) KDD17 (US)



(c) NDX100 (US)

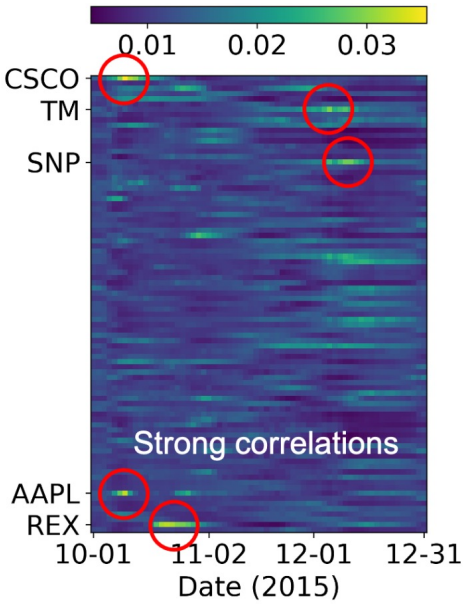
Q3. Data-Axis Attention (1)

- Attention scores indicate stock importances
  - We sort the stocks by their attention scores
  - Region C contains the most influential stocks

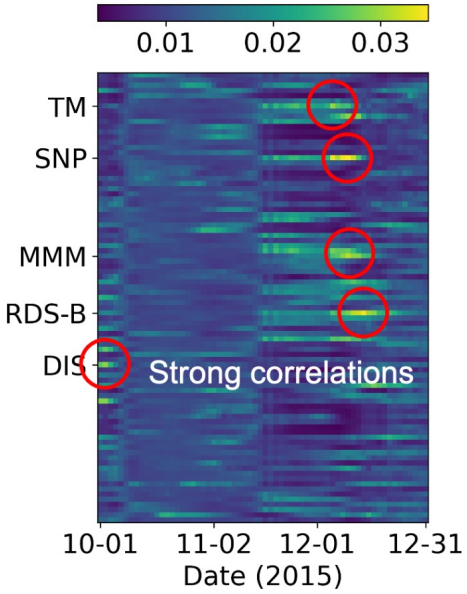


Q3. Data-Axis Attention (2)

- We study the cases of Amazon and Google
  - They show smooth changes in scores with others



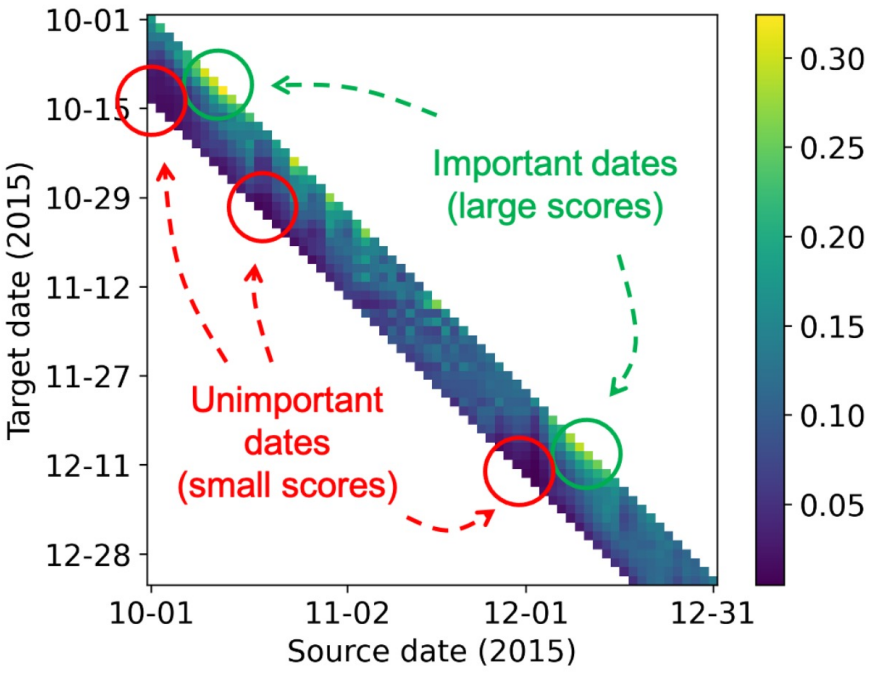
(a) Amazon (AMZN)



(b) Google (GOOG)

Q3. Temporal Attention

- DTML learns also the importances of days
  - Important days have larger influences than others



Q4. Ablation Study

- We perform an ablation study for DTML
  - **TA**, **SA**, and **MC** refer to temporal attention, stock attention, and multi-level contexts, resp.
  - All three modules help improving the accuracy

Model	ACC	MCC
DTML-TA-SA-MC	0.5349 ± 0.0140	0.0828 ± 0.0246
DTML-TA	0.5574 ± 0.0163	0.1387 ± 0.0334
DTML-SA	0.5622 ± 0.0153	0.1453 ± 0.0205
DTML-MC	0.5724 ± 0.0177	0.1856 ± 0.0313
DTML	**0.5744 ± 0.0194**	**0.1910 ± 0.0315**

Outline

- Introduction
- Proposed Method
- Experiments
- **Conclusion**

Conclusion

- We propose DTML for stock price prediction
  - To learn dynamic correlations between stocks
- DTML consists of three modules
  - **Time-axis attention** for the prices of each stock
  - **Context aggregation** with a global market trend
  - **Data-axis attention** by a transformer encoder
- We perform experiments on six datasets
  - DTML achieves the state-of-the-art accuracy
  - DTML makes up to 13.8%p higher actual profit

Attention-Based Autoregression for Accurate and Efficient Time Series Forecasting

Jaemin Yoo and **U Kang**

Computer Science & Engineering

Seoul National University

SDM 2021

Outline

- **Introduction**
- Previous Works
- Proposed Method
- Experiments
- Conclusion

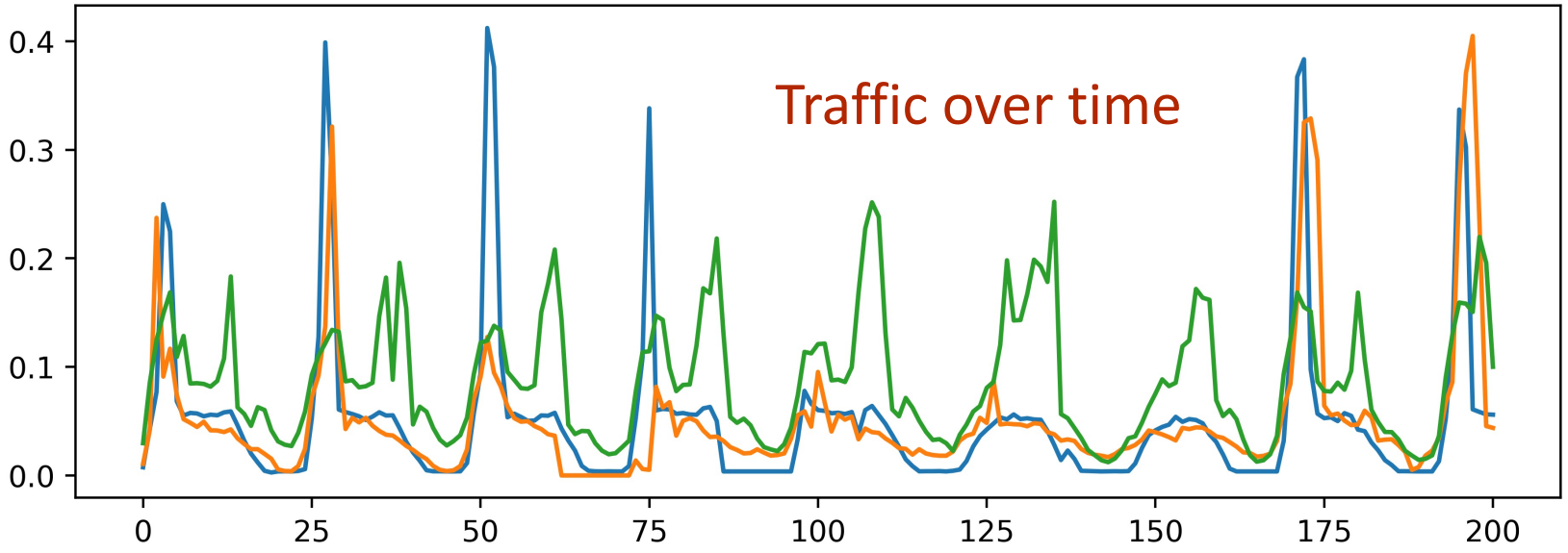
Time Series Forecasting

- Core problem that has numerous applications
  - Stock price prediction
  - Product sales forecasting
  - Weather forecast

https://www.simplilearn.com/tutorials/data-science-tutorial/time-series-forecasting-in-r

Multivariate Time Series

- Most time series data are **multivariate**
  - Such variables have *correlations* to each other
  - Prices of stocks, sales of products, …



Problem Definition

- Multivariate time series forecasting
  - **Given**
    - Multivariate time series $\mathbf{X} \in \mathbb{R}^{d \times w}$
      - d is the number of variables
      - w is the number of recent observations
    - Prediction horizon h
      - Larger h makes the problem more difficult
  - **Predict**
    - The observation $\mathbf{y} \in \mathbb{R}^{d}$ after h time steps

Outline

- Introduction
- **Previous Works**
- Proposed Method
- Experiments
- Conclusion

Summary

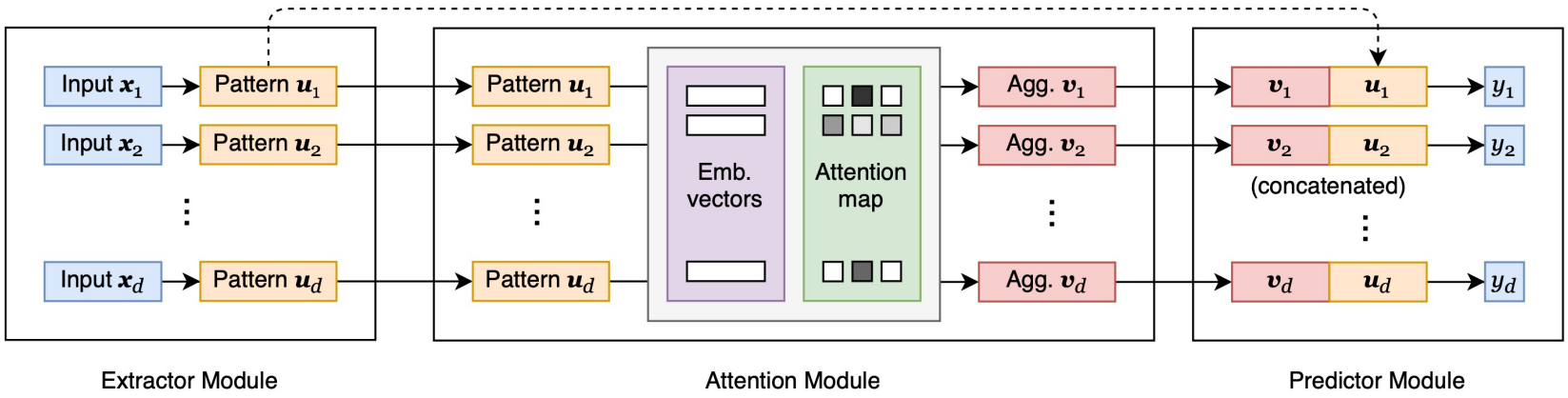
- Existing models are either too simple or have too many parameters
  - AR and TRMF
    - Cannot capture complex patterns in time series
  - VAR, LSTM, GRU, and LSTNet
    - Contain too many parameters and easily overfit
- **Research motivation:**
  - *To correlate variables with minimal parameters*

Outline

- Introduction
- Previous Works
- **Proposed Method**
- Experiments
- Conclusion

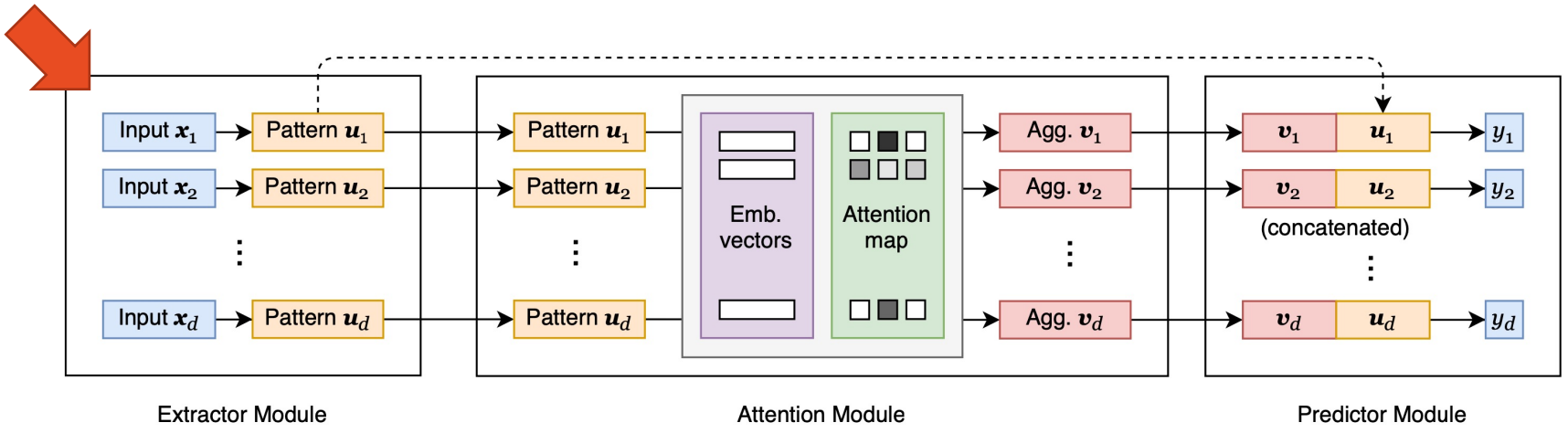
Overview

- **AttnAR** (attention-based autoregression)
  - Our approach for efficient multivariate forecasting
  - End-to-end framework of three separable modules
    - **Extractor**, **attention** and **predictor** modules
    - Module structure is our key idea for high efficiency



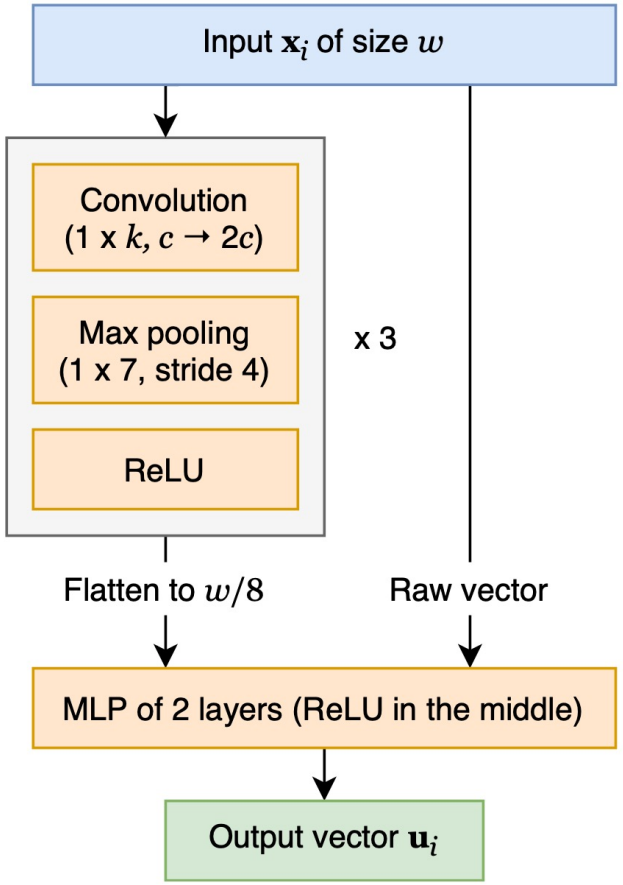
Extractor Module (1)

- **Extractor module** captures univariate patterns
  - Transforms a raw observation $\mathbf{x}_i$ of each variable i into a pattern vector $\mathbf{u}_i$ by a neural network
    - $\mathbf{u}_i$ is fed into both attention and predictor modules
  - *Which network should we use for efficiency?*



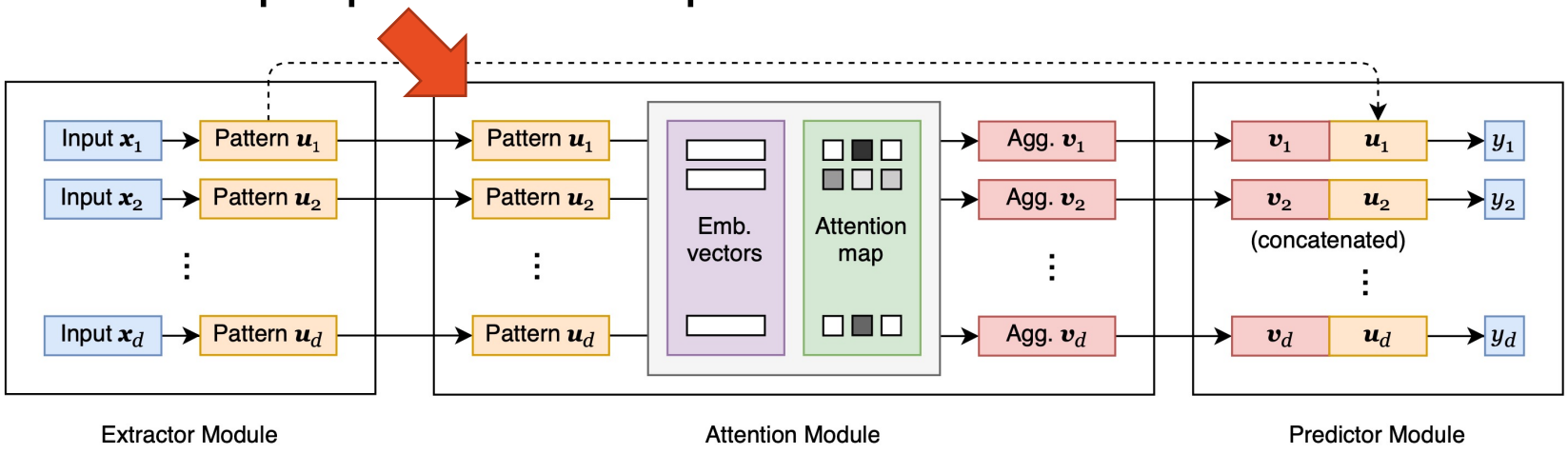
Extractor Module (2)

- **MCE (mixed-convolution extractor)**
  - Our proposed model for efficient pattern extraction
  - **Shallow dense layers**
    - Connect distant time steps
    - Low degree of abstraction
  - **Deep convolution layers**
    - Focus on adjacent time steps
    - High degree of abstraction



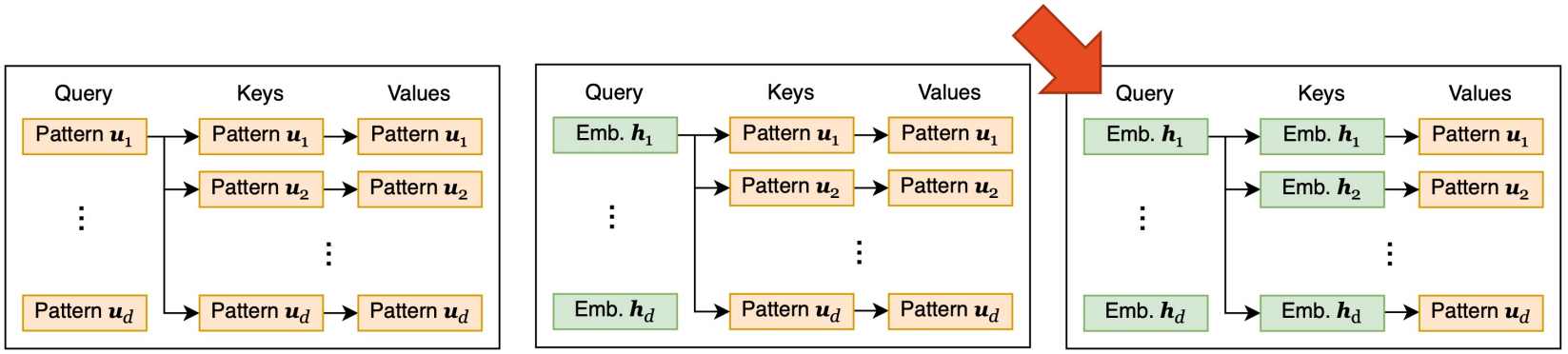
Attention Module (1)

- **Attention module** correlates given variables
  - The main component of our AttnAR
  - Correlates the pattern vectors $\{\mathbf{u}_i\}$ of variables by an attention map $\mathbf{S} \in \mathbb{R}^{d \times d}$ and returns $\{\mathbf{v}_i\}$
  - We propose three options as the attention function



Attention Module (2)

- We use the **time-invariant attention (TIA)** as our attention function
  - Learns a static embedding $\mathbf{h}_i$ for each variable i
  - Generates the attention map from $\mathbf{h}_i$, excluding $\mathbf{u}_i$
- The attention becomes robust and consistent

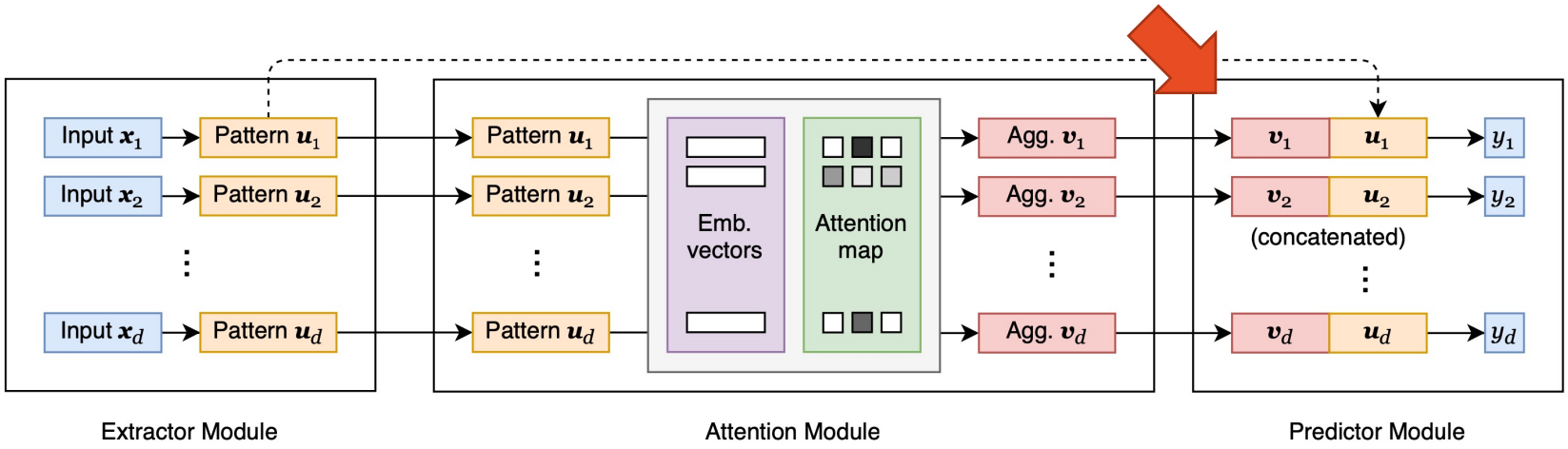


(a) Basic attention (Section 3.3.1). (b) Hybrid attention (Section 3.3.2). (c) Time-invariant attention (S. 3.3.3).

Predictor Module

- Lastly, the **predictor module** simply produces the final prediction given the pattern vectors:

$$\hat{y}_i = f_{\text{mlp}}(\mathbf{u}_i \parallel \mathbf{v}_i)$$



Outline

- Introduction
- Previous Works
- Proposed Method
- **Experiments**
- Conclusion

Experimental Setup

- We use four multivariate time series datasets

Dataset	Length	Dim.	Granularity
Traffic	17,544	862	1 hour
Electricity	26,304	321	1 hour
Solar-Energy	52,560	137	10 minutes
Exchange-Rate	7,587	8	1 day

- The prediction horizon h varies in $\{6, 12, 24\}$
- **Evaluation:** Root relative squared error (RSE)
  - RMSE divided by the standard deviation of $\mathbf{Y}$

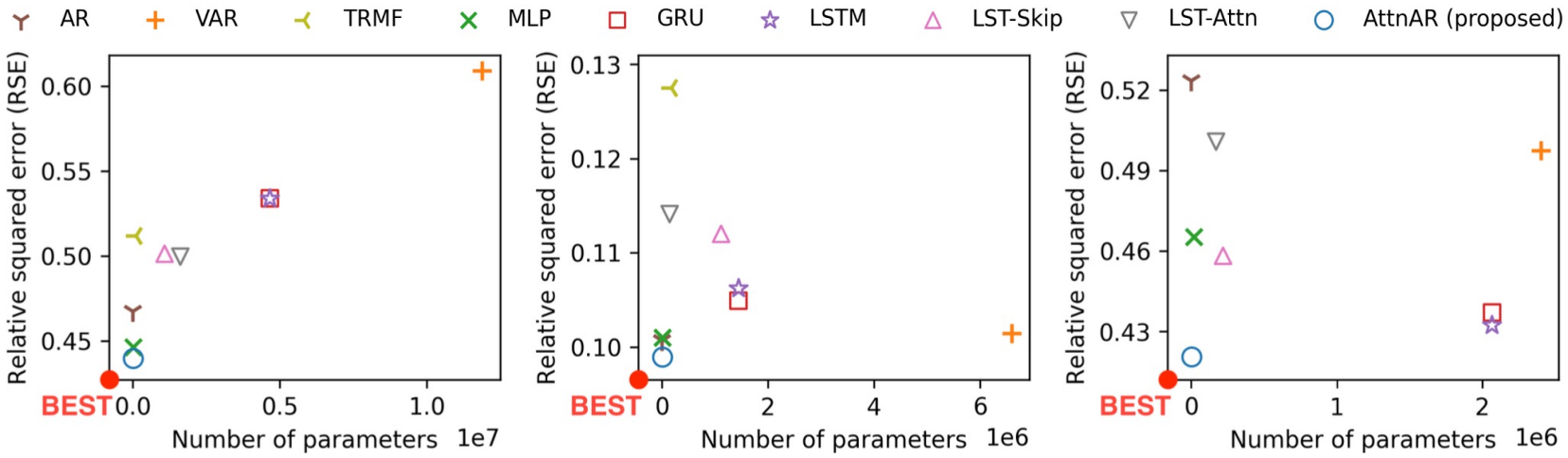
Forecasting Accuracy

- AttnAR makes the most accurate predictions in nine of the twelve cases
  - Exchange-Rate is very noisy, and AR does the best
  - The improvement is significant in Solar-Energy

Method	Traffic			Electricity			Solar-Energy			Exchange-Rate		
	h=6	h=12	h=24	h=6	h=12	h=24	h=6	h=12	h=24	h=6	h=12	h=24
AR	.4647	.4659	.4675	.0930	.0983	.1007	.3120	.4195	.5235	**.0238**	.0329	**.0433**
VAR	.5909	.6008	.6088	.0964	.1010	.1014	.2965	.4112	.4974	.0496	.0652	.0872
TRMF	.4871	.4909	.5120	.1050	.1062	.1275	.6001	.7112	.8434	.0425	.0466	.0542
MLP	.4368	.4436	.4464	**.0871**	.0965	.1010	.2747	.3592	.4652	**.0238**	**.0328**	.0436
GRU	.5158	.5225	.5340	.1088	.0974	.1049	.2485	.3229	.4370	.0322	.0465	.0639
LSTM	.5195	.5268	.5337	.1043	.1008	.1062	.2539	.3328	.4323	.0412	.0503	.0658
LST-Skip	.4811	.4900	.5013	.0993	.0959	.1120	.2537	.3448	.4582	.0279	.0425	.0553
LST-Attn	.4780	.4895	.4996	.0936	.0990	.1141	.2552	.3528	.5007	.0379	.0473	.0590
AttnAR	**.4287**	**.4370**	**.4396**	**.0871**	**.0942**	**.0989**	**.2272**	**.3057**	**.4205**	.0240	.0336	.0448

Parameter-Efficiency (1)

- AttnAR makes the best parameter-efficiency
  - The error often increases with the model size
  - Overfitting is common in multivariate forecasting



(a) Traffic (b) Electricity (c) Solar-Energy

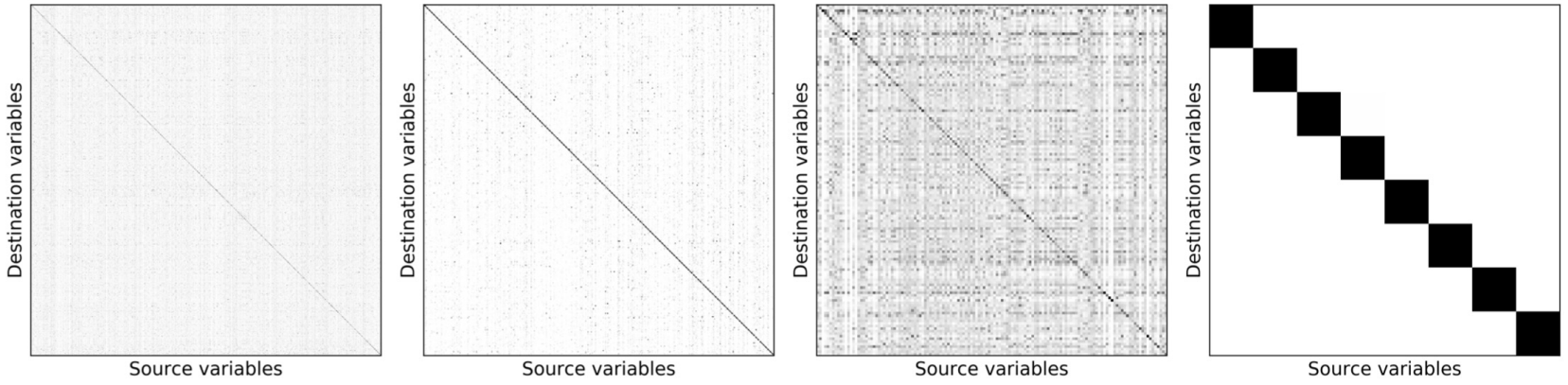
Parameter-Efficiency (2)

- RNN-based models requires many parameters, especially in a dataset with many variables
- AttnAR has up to 42.6× fewer parameters

Method	Traffic	Elec.	Solar	Exchange
GRU	4665.3K	1445.4K	2066.9K	14.5K
LSTM	4665.3K	1445.4K	2066.9K	804.4K
LST-Skip	1086.1K	1114.1K	218.7K	65.4K
LST-Attn	1621.5K	144.1K	170.5K	18.6K
AttnAR	**25.5K**	**9.5K**	**10.7K**	**0.9K**

Attention Map

- AttnAR generates interpretable attention maps
  - Strong correlations in Solar-Energy
  - Weak correlations in Traffic and Electricity
  - No correlations in Exchange-Rate



(a) Traffic (b) Electricity (c) Solar-Energy (d) Exchange-Rate

Outline

- Introduction
- Previous Works
- Proposed Method
- Experiments
- **Conclusion**

Conclusion

- **AttnAR (attention-based autoregression)**
  - Our proposed model for multivariate forecasting
- Main ideas of AttnAR
  - End-to-end learning of three separable modules
  - **MCE** for efficient extraction of univariate patterns
  - **TIA** for consistent and robust attention maps
- Experimental results
  - AttnAR consistently outperforms existing models

Thank you!