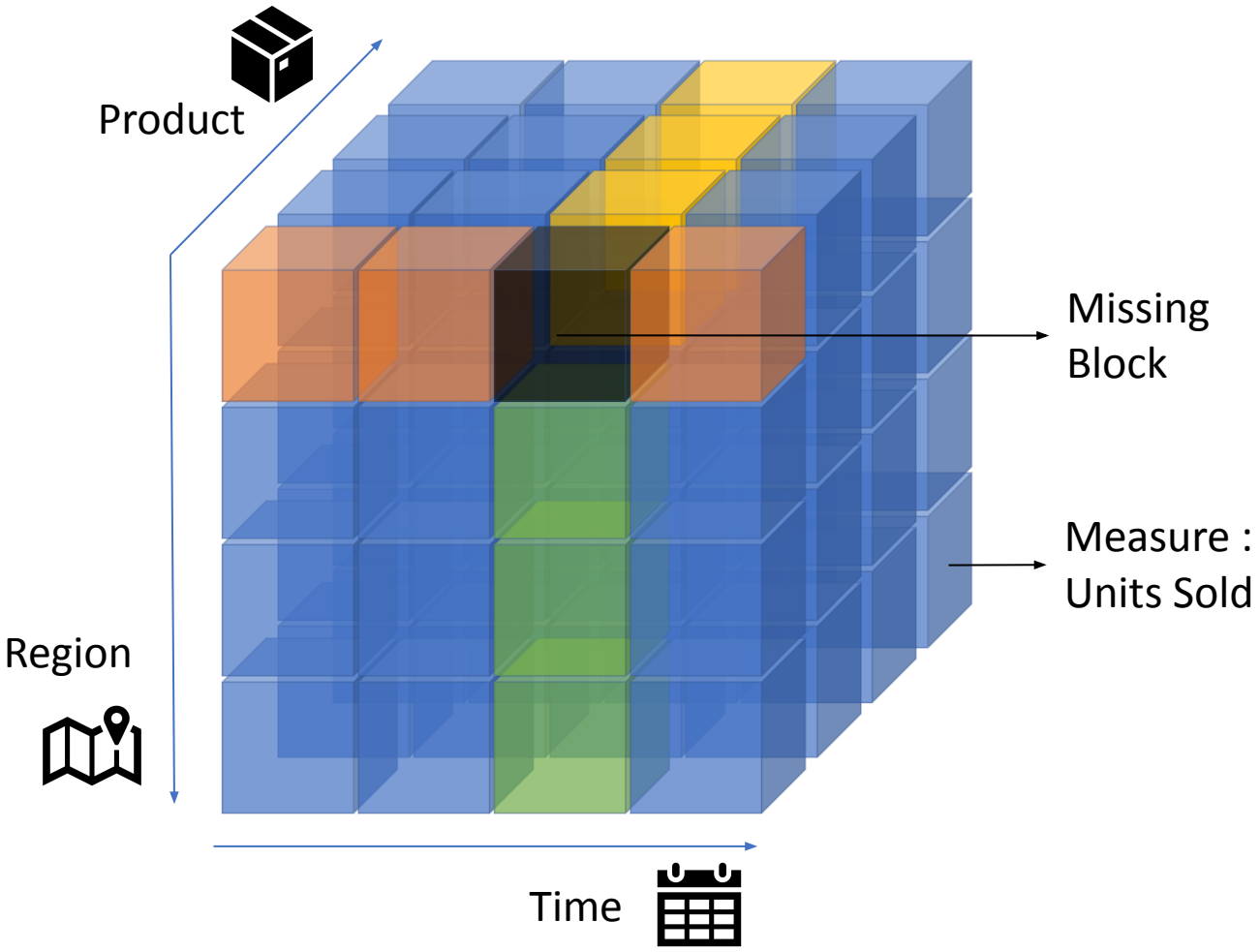


DeepMVI

Problem Statement

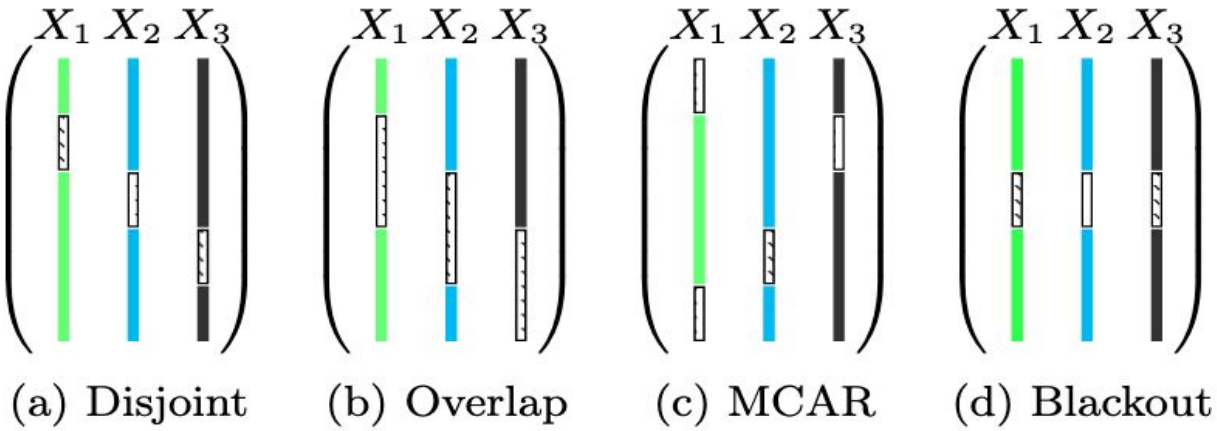
- **Problem considered** : Missing Value Imputation
- **Input** : Incomplete Multidimensional (MD) data
  - Say : Sales data (number of units of a products sold), with 3 dimensions
  - For each time (hour/day), product and store we have units sold
- **Output** : Imputed MD data
- **Optimize** : Minimise MAE loss between Ground Truth MD and Imputed MD

Challenges – Robustness in Wild

- Black-box Deep learning solutions rarely work out
- MRNN: Bidirectional RNNs
  - *“The results of our evaluation show that NN-based recovery techniques are not suitable for the data we use here. For instance, MRNN [.] incur a high error (average RMSE higher than 1), while it takes orders of magnitude more time than the slowest algorithm from our benchmark”* ***(2020)***

Challenges – Robustness in Wild

- Work across **arbitrary** missing scenarios : a sample illustrated
- X1, X2, X3 : different time series
- Blackout's imputation strategy very different from MCAR's
  - i.e. Model and it's training should be amiable to missing scenarios



Some common missing scenarios encountered in real world data

Challenges – Robustness in Wild

- Work across varied datasets
  - Different correlations exist in real world dataset
- Periodic Time Series Datasets
- Non-Periodic Series with Temporal Patterns
- MD data with strong correlations across time series correlation
- Need to extract and combine both **Within Time series** and **Across Time Series relations**

Challenges – General Purpose Utility

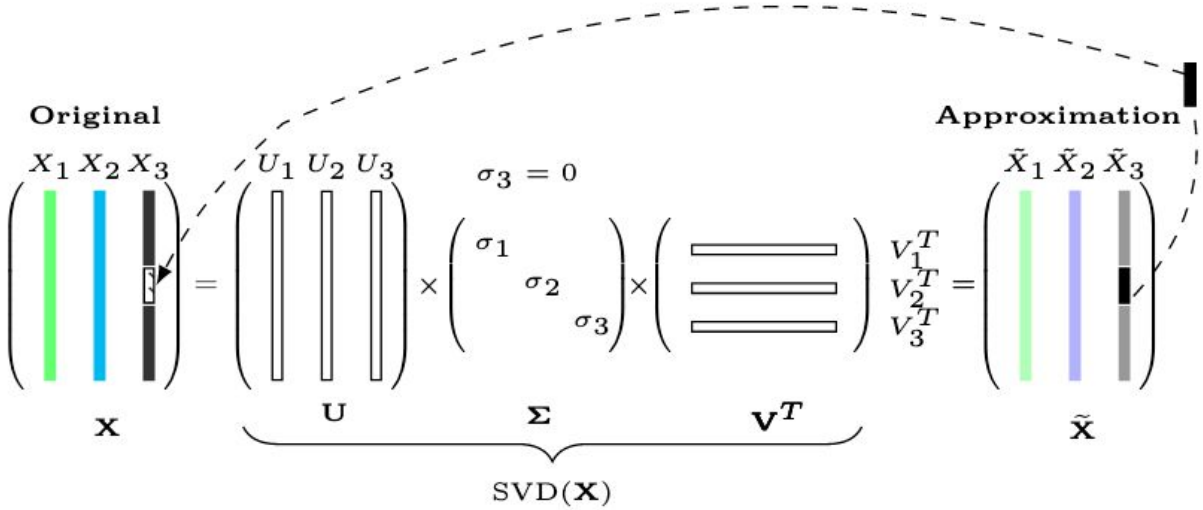
- Deep Learning
  - Over-parameterized
  - Dataset specific tuning
  - Training Samples in order of millions
  - Training time in order of hours/days
  - GPU intensive compute
- Conventional methods (e.g. SVD imputation) are
  - Parameter free (almost)
  - Can be used as Blackbox
  - No training. Just inference
  - Running time in minutes
  - GPU use minimal
- Pose problem for Deep Learning methods to be widely adopted

Challenges - Scalability

- MD data scales O(n^d) where we have d dimensions each having n distinct values
- Conventional techniques scale linearly with data i.e. O(n^d)

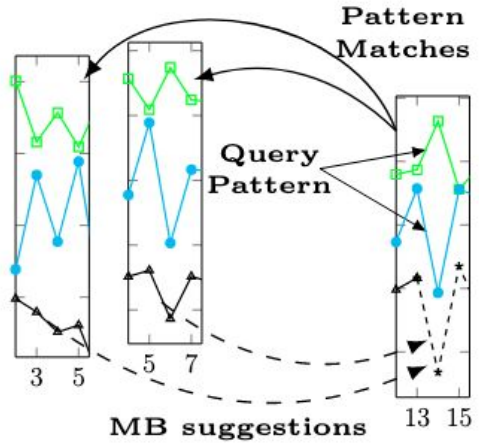
Prior Work – Conventional

- High MAE loss
- Matrix Factorization based and Pattern Matching based



(b)

Matrix Factorisation Based

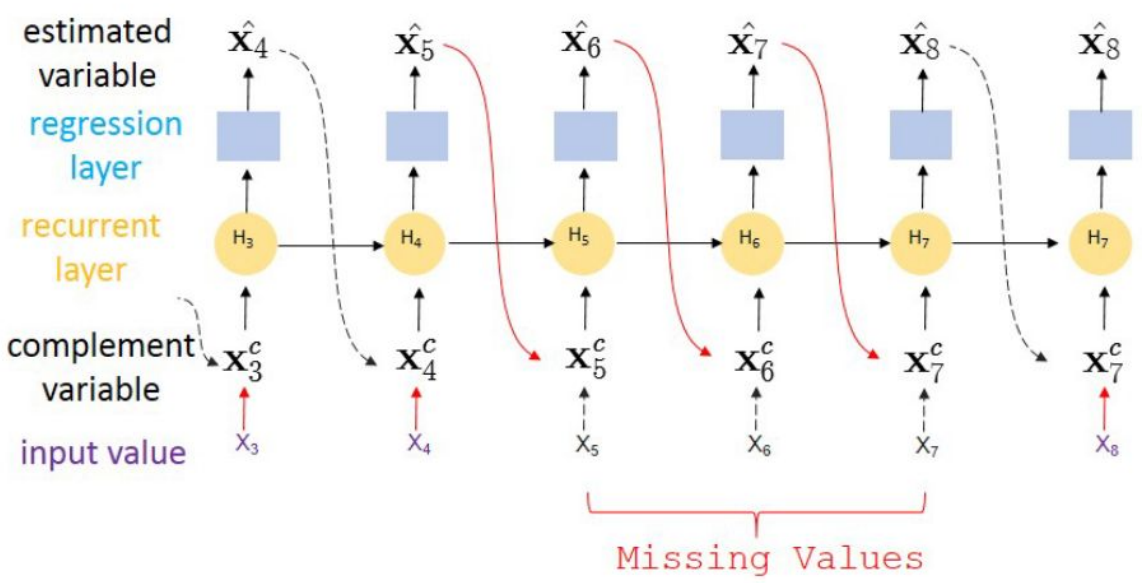


(c)

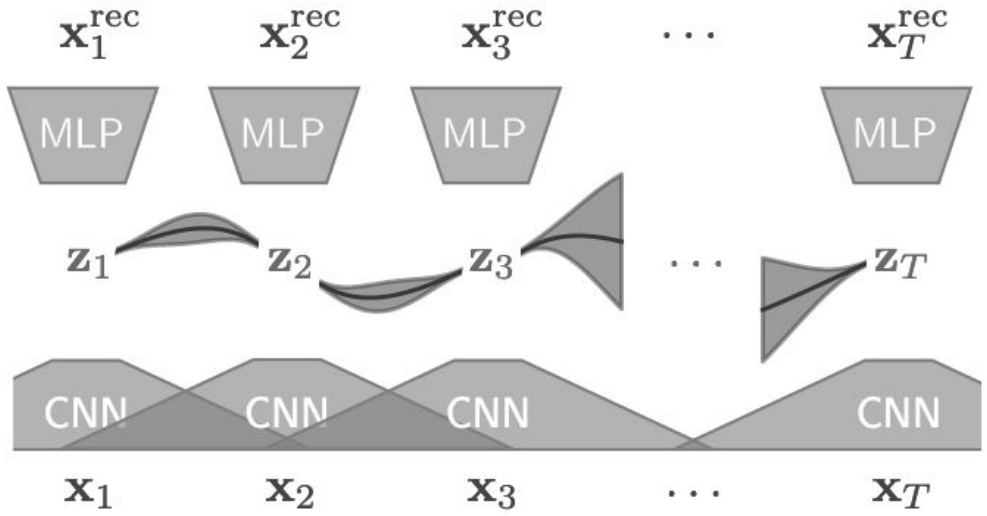
Pattern Matching

Prior Work – Deep Learning

- Deep Learning methods fail to work on general real world problems
- Highly Parametric



BRITS : Bidirectional RNNs **(2018)**

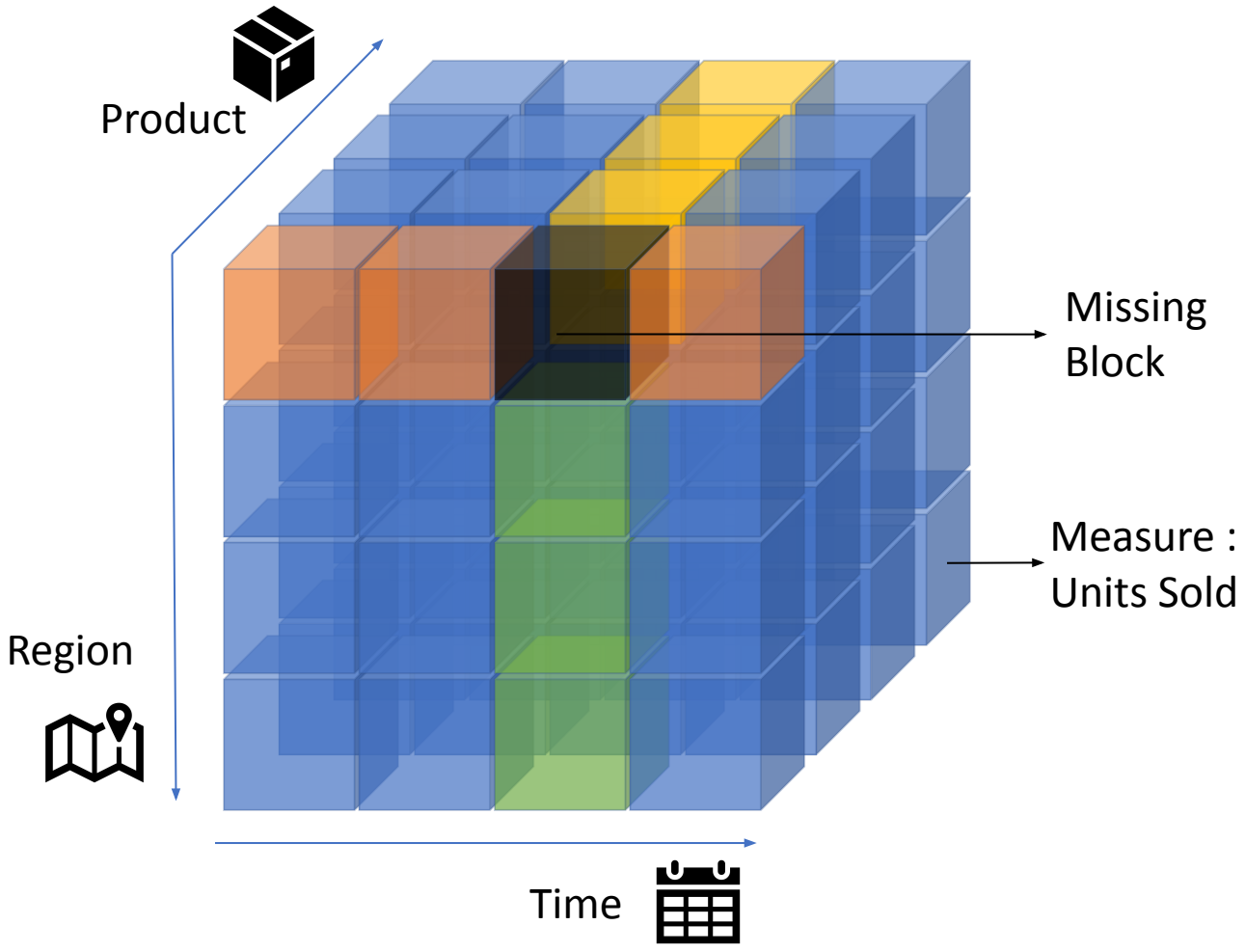


GPVAE : Gaussian Process on VAE Latents **(2020)**

Contributions

- Modular neural architecture to extract and combine temporal and cross-series signals
- Augment it with Robust and Scalable method of training parameters to deal with varying datasets and missing scenarios
- Temporal Transformer module specializing in dealing with temporal data

Proposal Outline



- Given missing block for store s, product p for time range t
  - Black in figure
- View MD as set of d, n size vectors
  - Each vector captures variation in 1 dimension keeping others constant
  - E.g. *Sale at store s for product p for all time indices*, *Sale at all stores for product p and time t*
- Get predicted imputed value from all n vectors in the set
- Linearly combine these predictions

Salient Features

- Better time complexity
  - Approx O(nd) instead of O(n^d)
- Disentanglement of Imputation signals from dimensions for Robust Imputation
- Final layers weights aid Interpretability and discovery of Causal signals

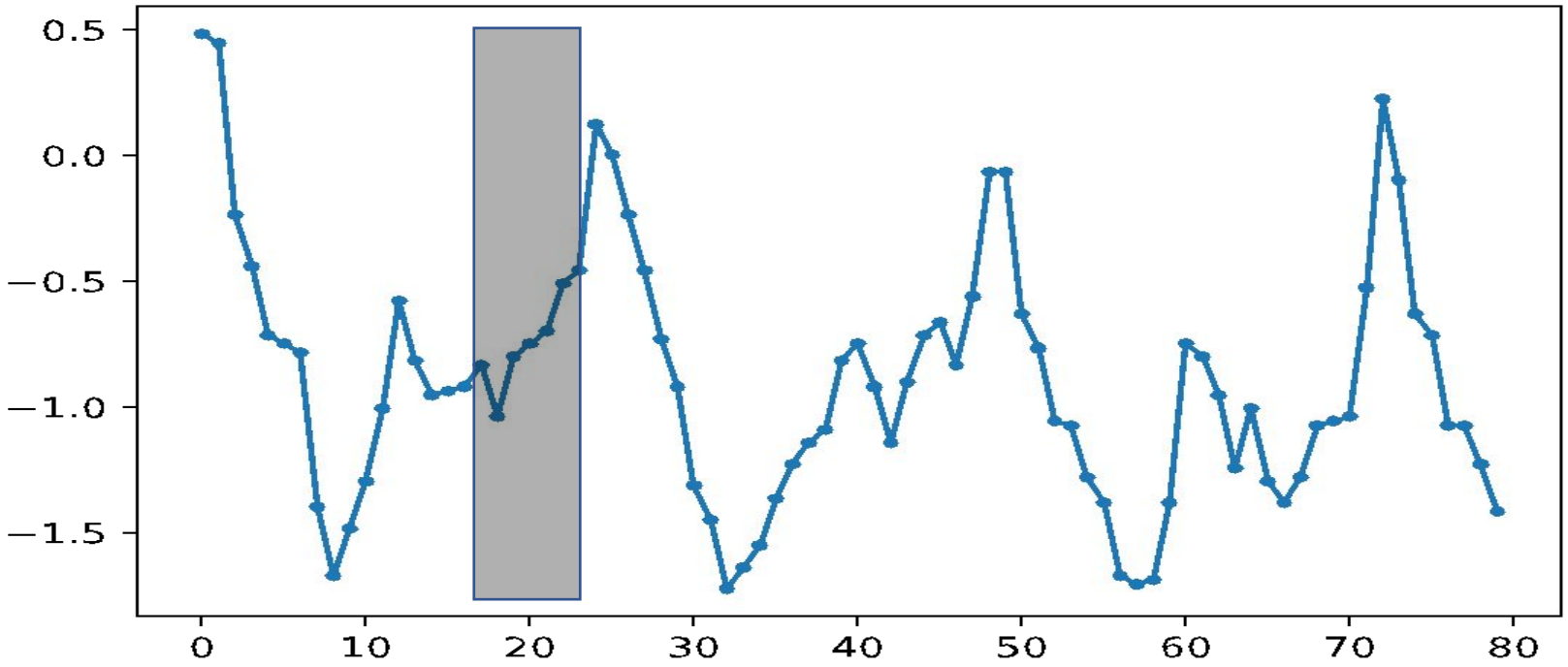
Salient Features

- Better time complexity
  - Approx O(nd) instead of O(n^d)
- Disentanglement of Imputation signals for Robust Imputation
  - Signals from temporal dimension work independent of signals from product dimension
- Interpretability of imputation
  - Weights of final linear combination indicative of causal dimension
  - i.e. some datasets have periodic series while others have strong correlations in sales across stores

Temporal Imputation

- Impute a single time series
- Time dimension is special : locality, periodicity
- Challenges to be resolved
  - Blocks of missing values
  - Long range dependency
- Design principals to be followed
  - Don’t over-parameterize
  - Fast training
  - GPU memory efficient

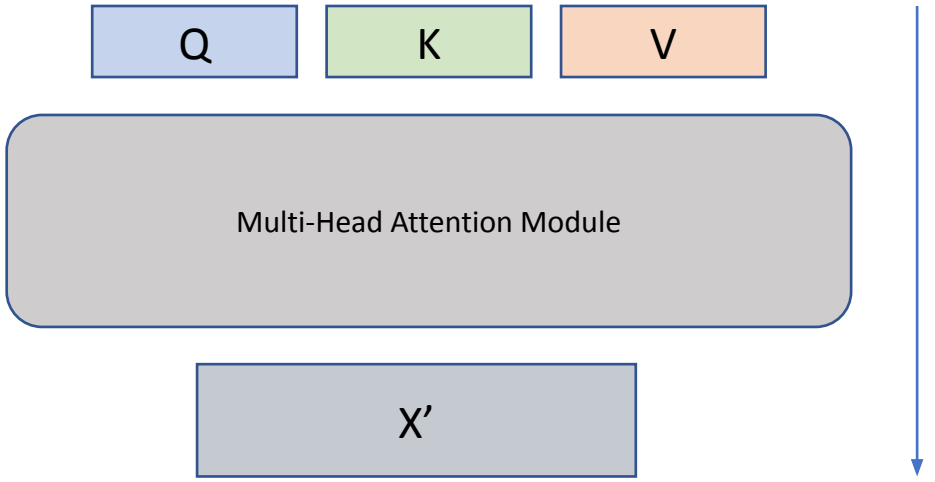
Temporal Transformer



- Consider a single time series (i.e. the time vector from the set)
  - Missing block shaded
- Temporal Transformer model to efficiently exploit temporal dependencies

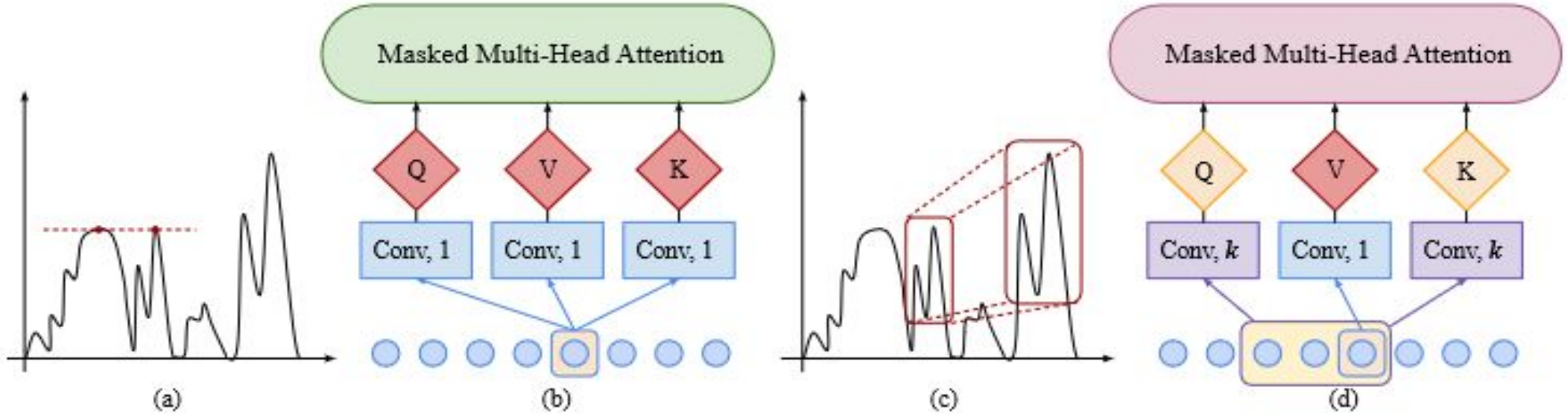
Would a simple Transformer work?

- A naïve solution to applying transformer to imputation task
- Given some time indices with missing values
- Construct Key and Value from given indices as
  - Positional encodings concatenated with scalar values
  - Key for missing indices is 0 (mask these indices in attention)
- Query for an index would be its positional encoding



Would a simple Transformer work?

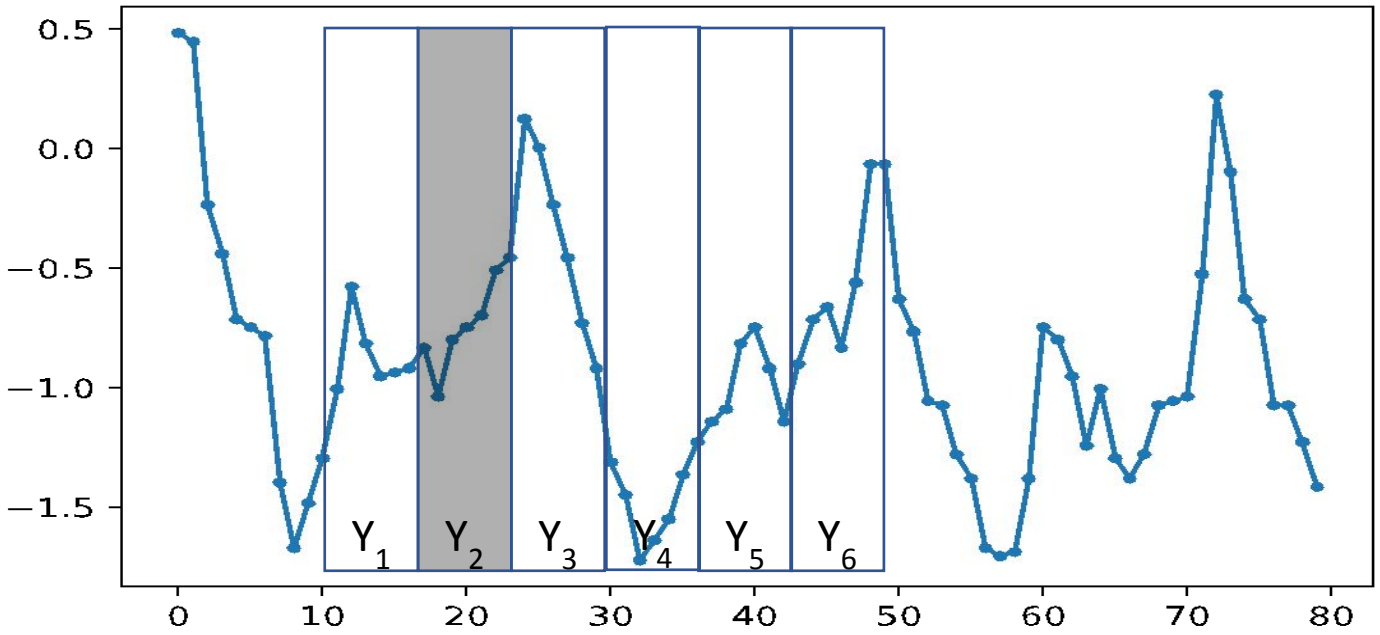
- Issues with the approach?
  - Queries just contains positional information (no contextual information)
  - Keys and Values contain just the scalar values with positional encodings. No contextual Information
  - Would work well when position is strong signal
    - Periodic series



Our Solution : Temporal Transformer

Block representation of the data via non-overlapping convolutions

$$Y_j = W_f X_{jw:(j+1)w} + b_f$$



- Patterns emerge at block level
  - Scalar values not informative
- Reduces temporal span by factor of w increasing running time and decreasing memory by O(w^2)

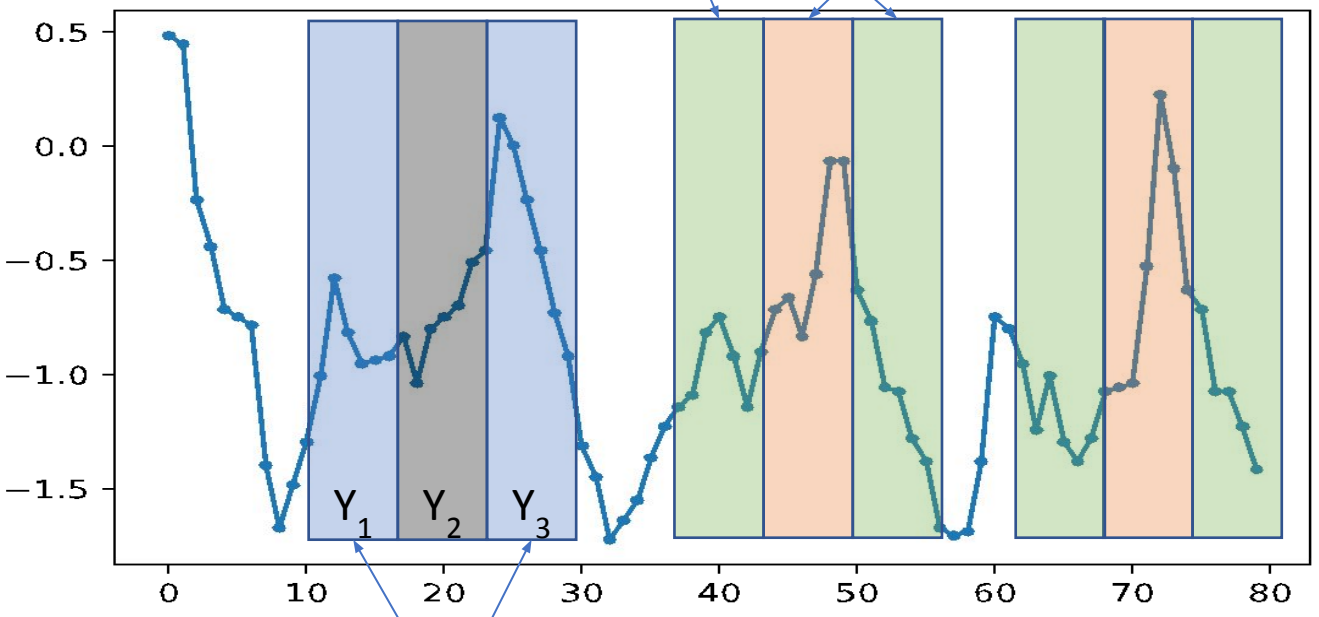
Temporal Transformer

Keys at all other positions

$$K(Y, j, A) = (([Y_{j-1}, Y_{j+1}] + e_j)W_k + b_k) \cdot \prod_{i=jw}^{(j+1)w} A_i$$

Values

$$V(Y_j) = Y_j W_v + b_v$$



Query at index j

$$Q(Y, j) = ([Y_{j-1}, Y_{j+1}] + e_j)W_q + b_q$$

- A is Availability matrix, used to mask out missing positions
- e is the positional encodings

$$\mathrm{Attn}(Q(\cdot), K(\cdot), V(\cdot), A, j) = \frac{\sum_{j'} \langle Q(Y, j), K(Y, j', A) \rangle V(Y_{j'})}{\sum_{j'} \langle Q(Y, j), K(Y, j', A) \rangle}$$

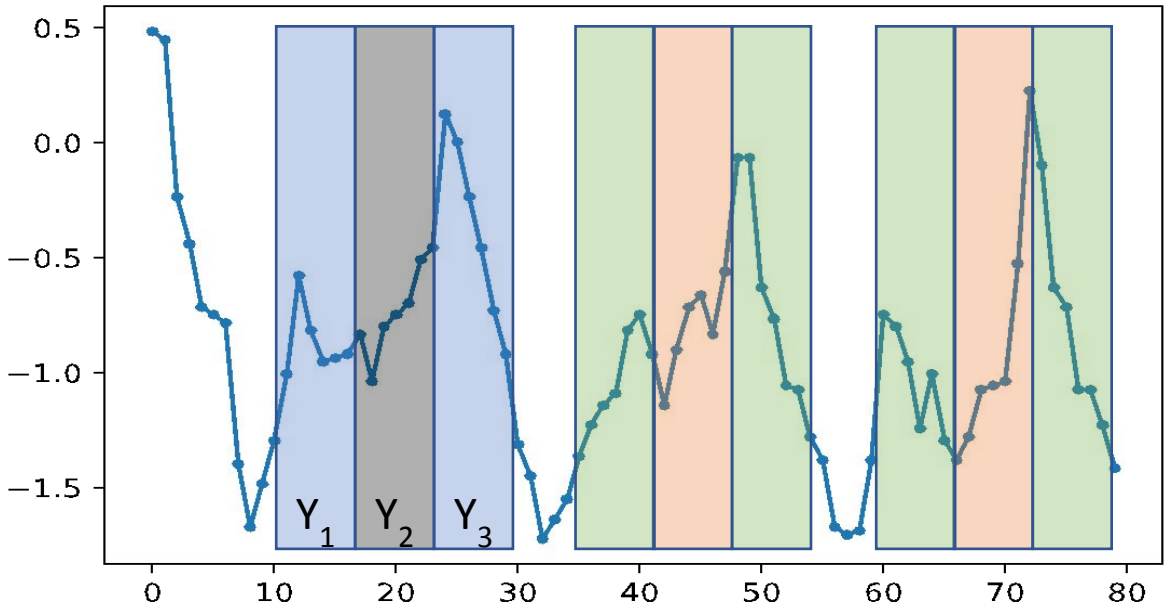
$$\mathbf{h}_j = [\mathrm{Attn}^1(\cdots), \ldots, \mathrm{Attn}^{n_{\mathrm{head}}}(\cdots)]$$

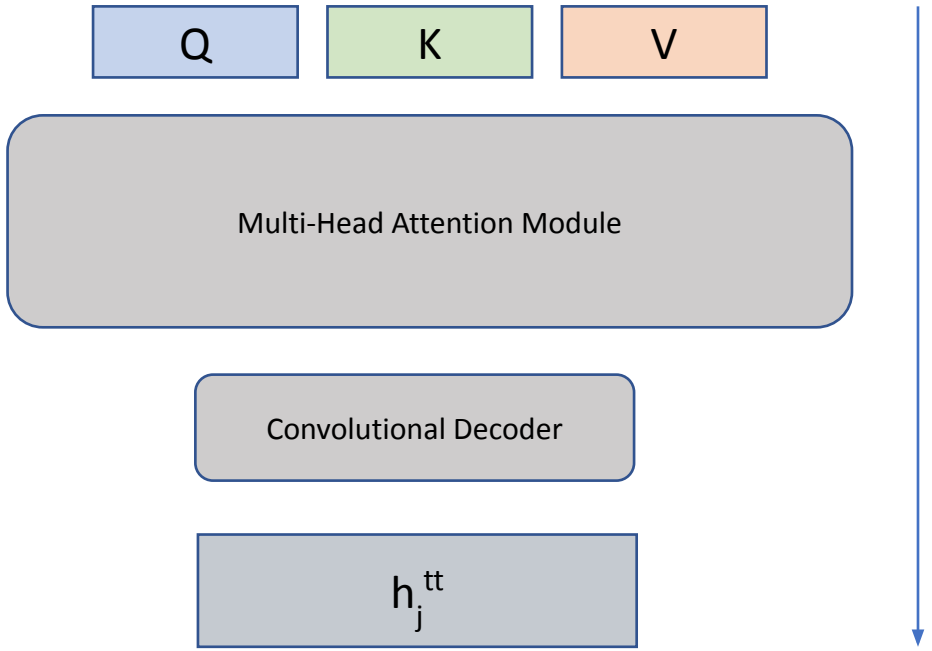
Multi-Head Attention module

$$\mathbf{h}_j^{\mathrm{ff}} = \mathrm{ReLU}(W_{d_2}(\mathrm{ReLU}(W_{d_1}(\mathrm{ReLU}(\mathbf{h}_j)))))$$

$$\mathbf{h}_j^{\mathrm{tt}} = \mathrm{ReLU}(W_d \mathbf{h}_j^{\mathrm{ff}} + b_d)$$

Convolutional Decoder
(Blocks to scalar values)





Why Temporal Transformer?

- Would **time embedding** help the model? Common place in DL
  - Blackout Scenario has time ranges with no sample for training
  - Provides an easy way for model to memorize the data
- Temporal Transformers modeling restricts the parameters to pattern matching within the time series.
  - Parameters don’t explicitly encode the scalar values at time indices
- Longer context for Queries and Keys?
  - Experimentally didn’t yield significantly better results. Also computationally expensive
- Sensitivity to window size?
  - Periodic series don’t have much sensitivity, while non-periodic series benefit from larger window sizes
  - Typical choice for window size == mean missing block size

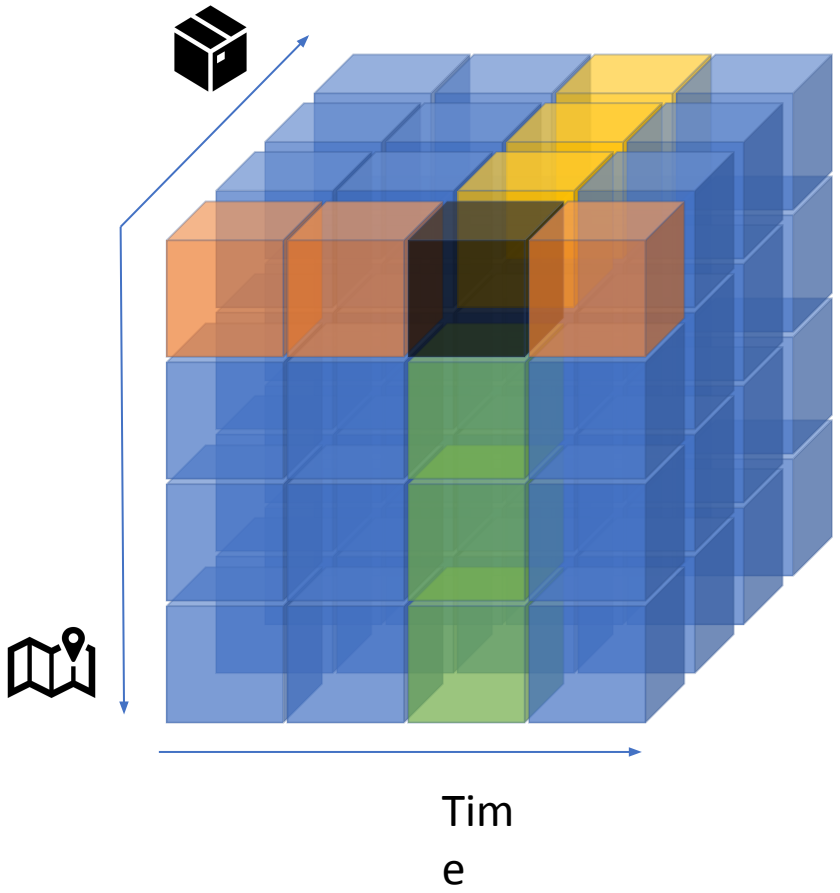
Why Temporal Transformer?

- Would **time embedding** help the model? Common place in DL
  - Blackout Scenario has time ranges with no sample for training
  - Provides an easy way for model to memorize the data

- Temporal Transformers modeling restricts the parameters to pattern matching within the time series.
  - Parameters don’t explicitly encode the scalar values at time indices

Other Dimensions

- We talked about capturing temporal relations in MD
- What about other dimension?
- Consider a dimension with k categories
  - E.g. Store dimension with k=3 different stores
- **Siblings** : All OLAP cells with just 1 dimension different
  - E.g. For a cell of Product *p* sold in Store *s* at time *t,* siblings along store dimension are all cells of Product *p* sold in Store *s'* at time *t,* where *s' != s*
  - All 1D colored strips correspond to different siblings in different dimensions

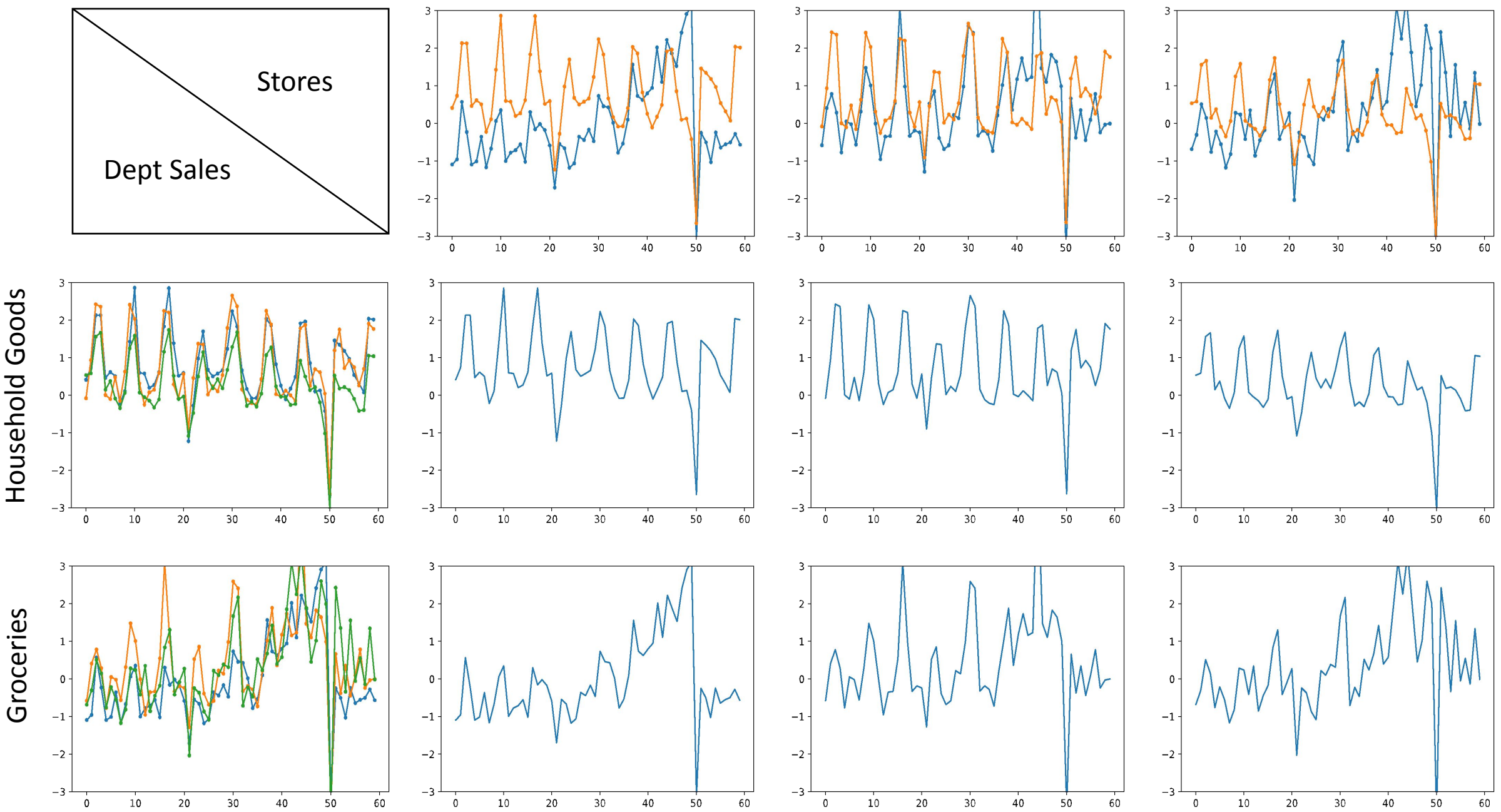


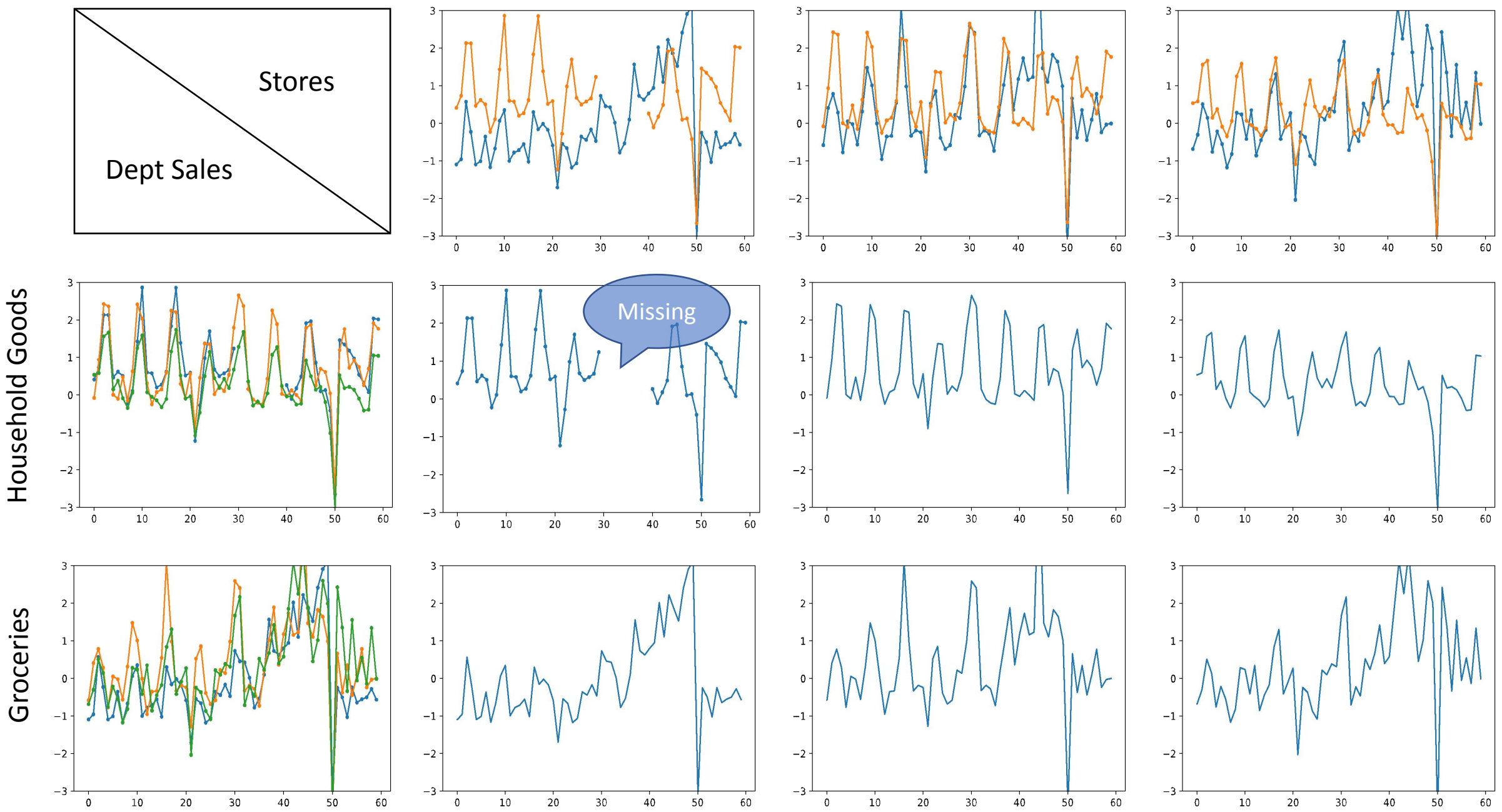
How to exploit Sibling Cells for Imputation?

- Each dimension is treated independently
- Each dimension gives its predicted imputed value
- Linear combination of these predictions taken as final prediction
  - Done by concatenating the predictions and applying a linear layer on top

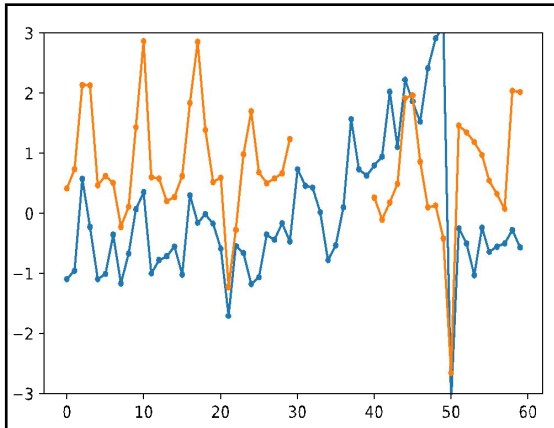
Kernel Regression Module

- We propose to solve the same using Kernel Regression
- Embed each dimension into a k dimensional space
- Each member of the dimension mapped to embedding in the space
  - E.g. For Shop dimension, Shop1,Shop2,Shop3 each are given some embedding
- These embeddings are learnt and their Euclidean distance is propotional to relatedness in the series


Store 1
Store 2
Store 3
Stores
Dept Sales
Household Goods
Groceries

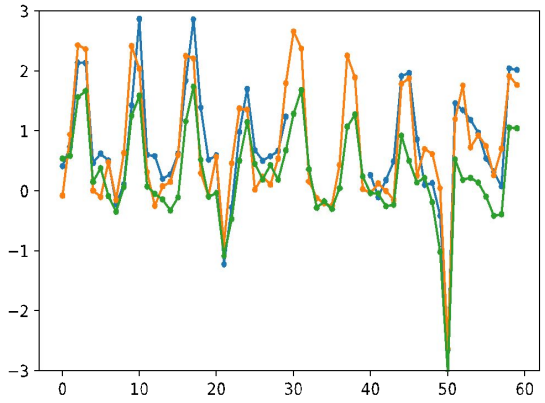

Store 1
Store 2
Store 3
Stores
Dept Sales
Household Goods
Groceries
Missing

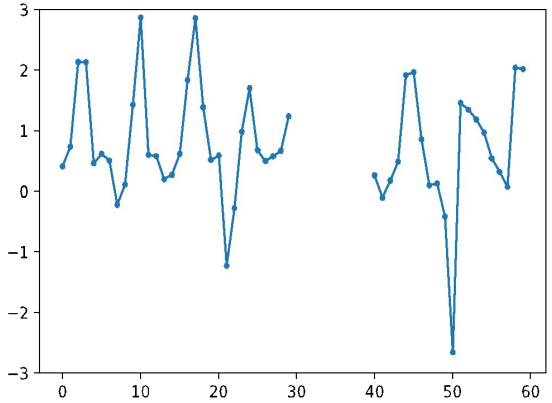
Store 1



Kernel Regression b/w
Different Products
Same Store

Household Goods





Kernel Regression b/w
Different Stores
Same Product

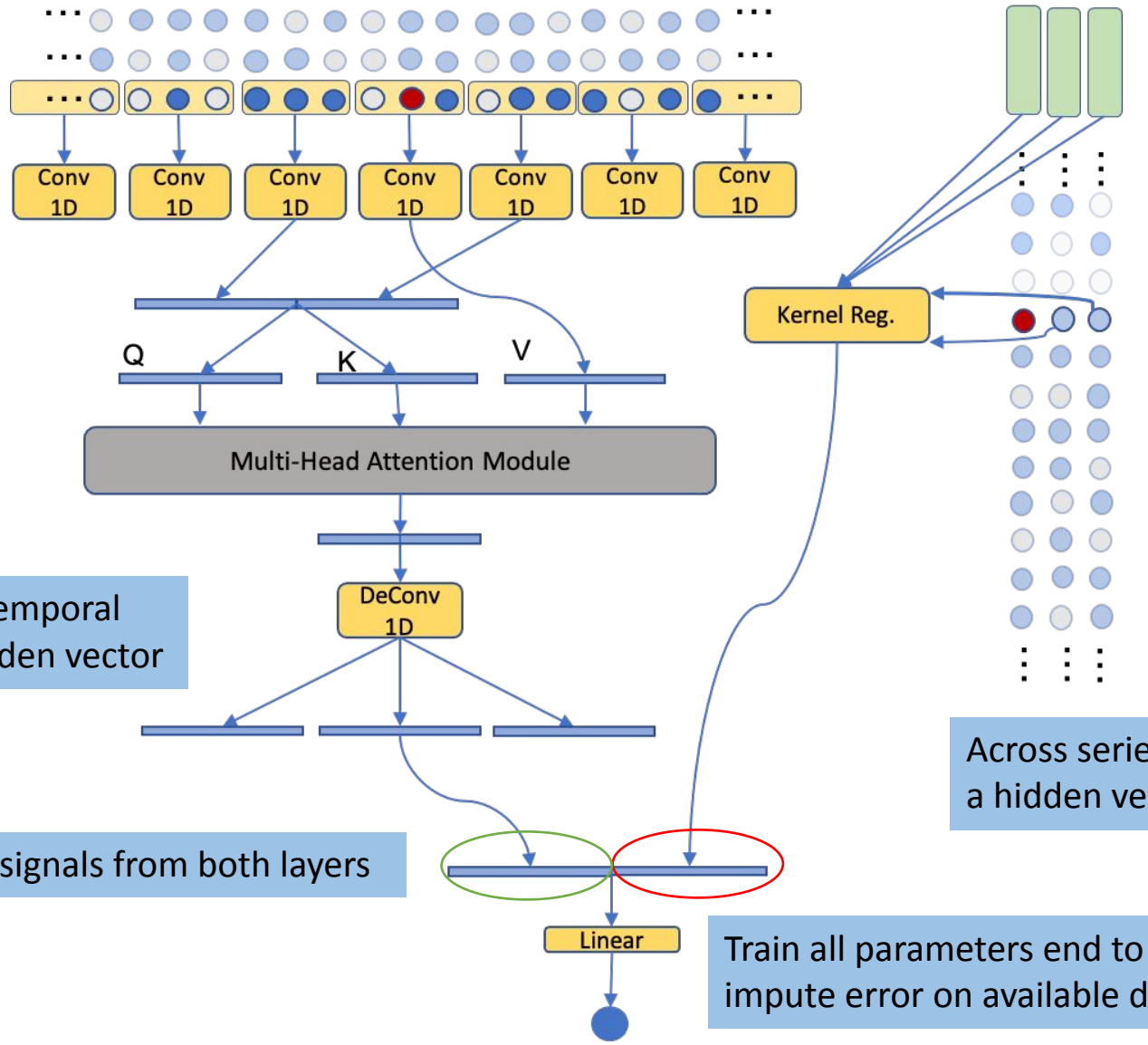
$$\mathcal{K}(k_i, k_i') = \exp\left(-\gamma * ||E[k_i] - E[k_i']||_2^2\right)$$

$$U_{(\mathbf{k},i),t} = \frac{\sum_{\mathbf{k}' \in \text{Sib}(\mathbf{k},i)} X_{\mathbf{k}',t} \mathcal{K}(k_i, k_i') A_{\mathbf{k}',t}}{\sum_{\mathbf{k}' \in \text{Sib}(\mathbf{k},i)} \mathcal{K}(k_i, k_i') A_{\mathbf{k}',t}}$$

$$\mathbf{h}^{\text{kr}} = \text{Concat}_i(U_{(\mathbf{k},i),t})$$

E[k] are the embeddings
Sib(k,i) gives all siblings of k along dimension I
A is the availability matrix masking points

Complete DeepMVI Model



Within series temporal signals as a hidden vector

Across series signals as a hidden vector

Combine signals from both layers

Train all parameters end to end to minimize impute error on available data

Scalable and Robust training procedure

- Naïve Solution for Training
  - Mask a block in Input and loss on output wrt GT input value

- 2 points of failure
  - Ignores the arbitrary missing scenario of input matrix
  - Wastes computation by having loss on just a block instead of whole series

Scalable and Robust training procedure

- Solutions? No masking in input

- Availability matrix for each cell is simulated missing patterns IID to input missing scenarios
  - Temporally Shift the missing blocks
  - Permute missing blocks across series

- Apply loss on whole imputed sequence
  - Single layer of attention with DeepMVI architecture ensures training

Datasets

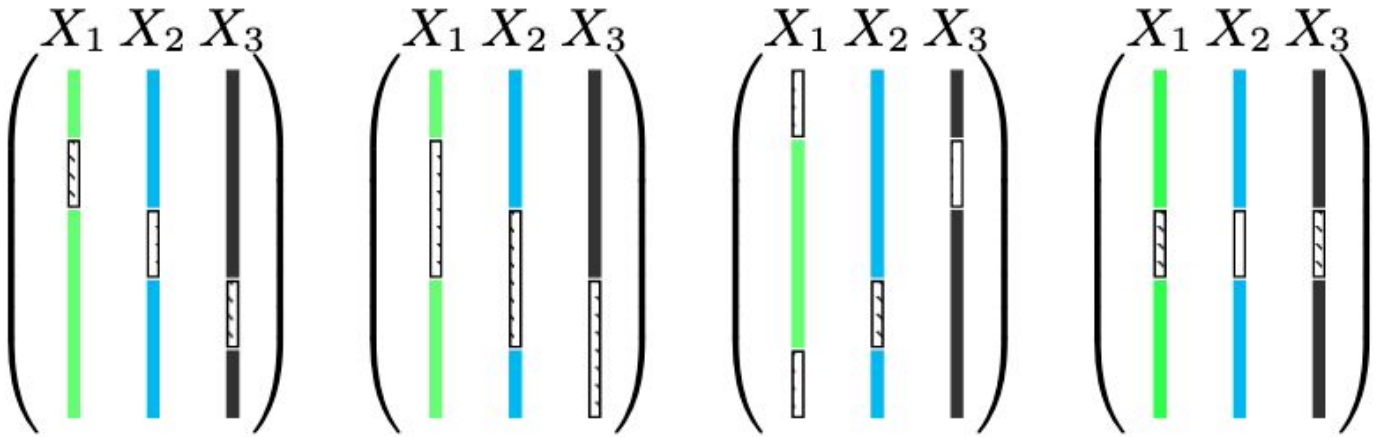
Dataset	Number of TS	Length of TS	Repetitions within TS	Relatedness across series	
AirQ	10	1k	Moderate	High	
Chlorine	50	1k	High	High	
Gas	100	1k	High	Moderate	
Climate	10	5k	High	Low	
Electricity	20	5k	High	Low	
Temperature	50	5k	High	High	
Meteo	10	10k	Low	Moderate	
BAFU	10	50k	Low	Moderate	
JanataHack	76*28	134	Low	High	Multi Dimensional Datasets
M5	10*106	1941	Low	Low	

Table 1: Datasets: All except the last two have one categorical dimension. Qualitative judgements on the repetitions of patterns along time and across series appear in the last two columns.

- Extensive analysis on 10 datasets
  - 8 from prior work
  - 2 datasets on OLAP introduced
- The Repetition and Relatedness analysis is qualitative
  - Ablation studies back up the same
- Relatedness and Repetitions play a major role in the error achieved by methods on the dataset

Results – Missing Scenarios Prelims

- Blackout has signals from just Temporal Transformer
  - Methods doing good temporal modeling would work here
- MCAR has signals from both Temporal and Across time series dimensions
  - Algorithm should effectively exploit both correlations (final linear layer)
- Disjoint and Overlap not very interesting (special case of MCAR)



(a) Disjoint (b) Overlap (c) MCAR (d) Blackout

Quantitative Results : Conventional Methods

- Better than All conventional methods
- Significant gains in Blackout and MCAR
  - Conventional methods don’t model temporal relations

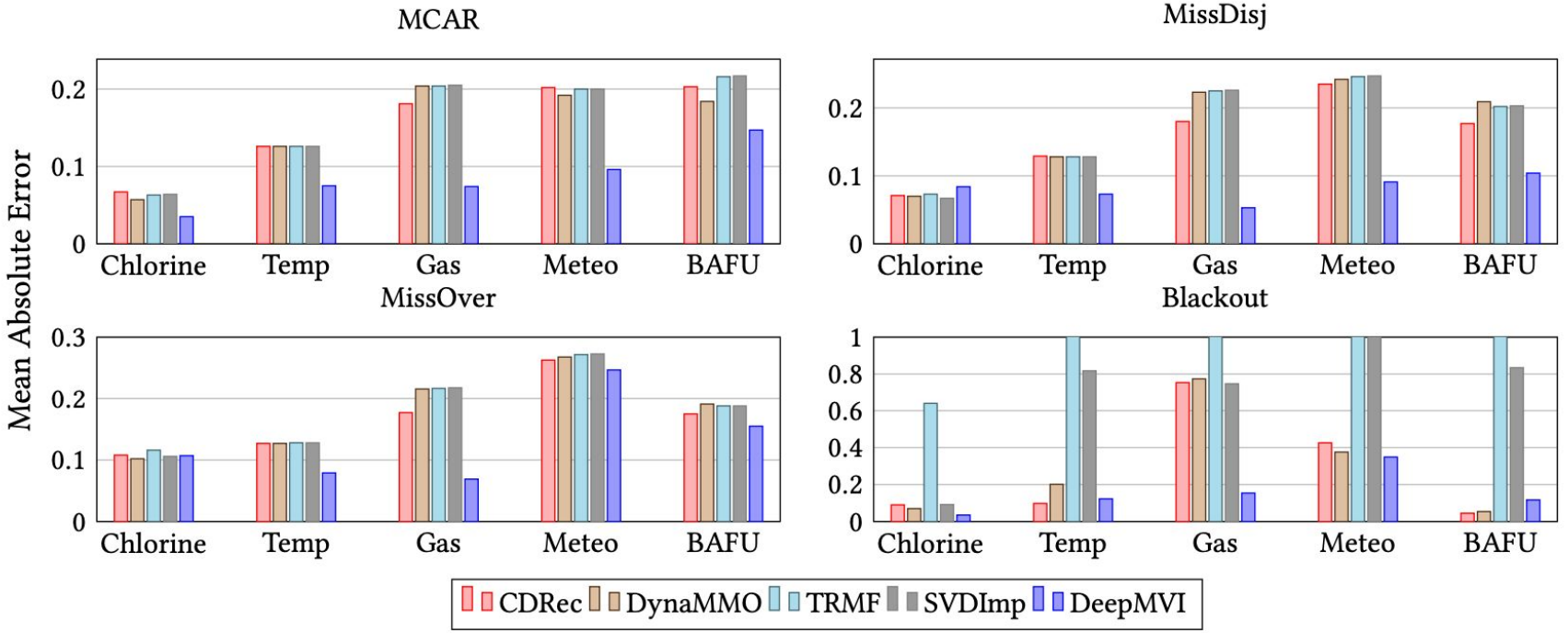


Figure 5: Mean Absolute Errors (y-axis) on five other datasets (on x-axis) on all four scenarios – MCAR, MissDisj, MissOver, and Blackout. Here, a fixed $x = 10\%$ of the series in each dataset has missing blocks.

Quantitative Results : Conventional Methods

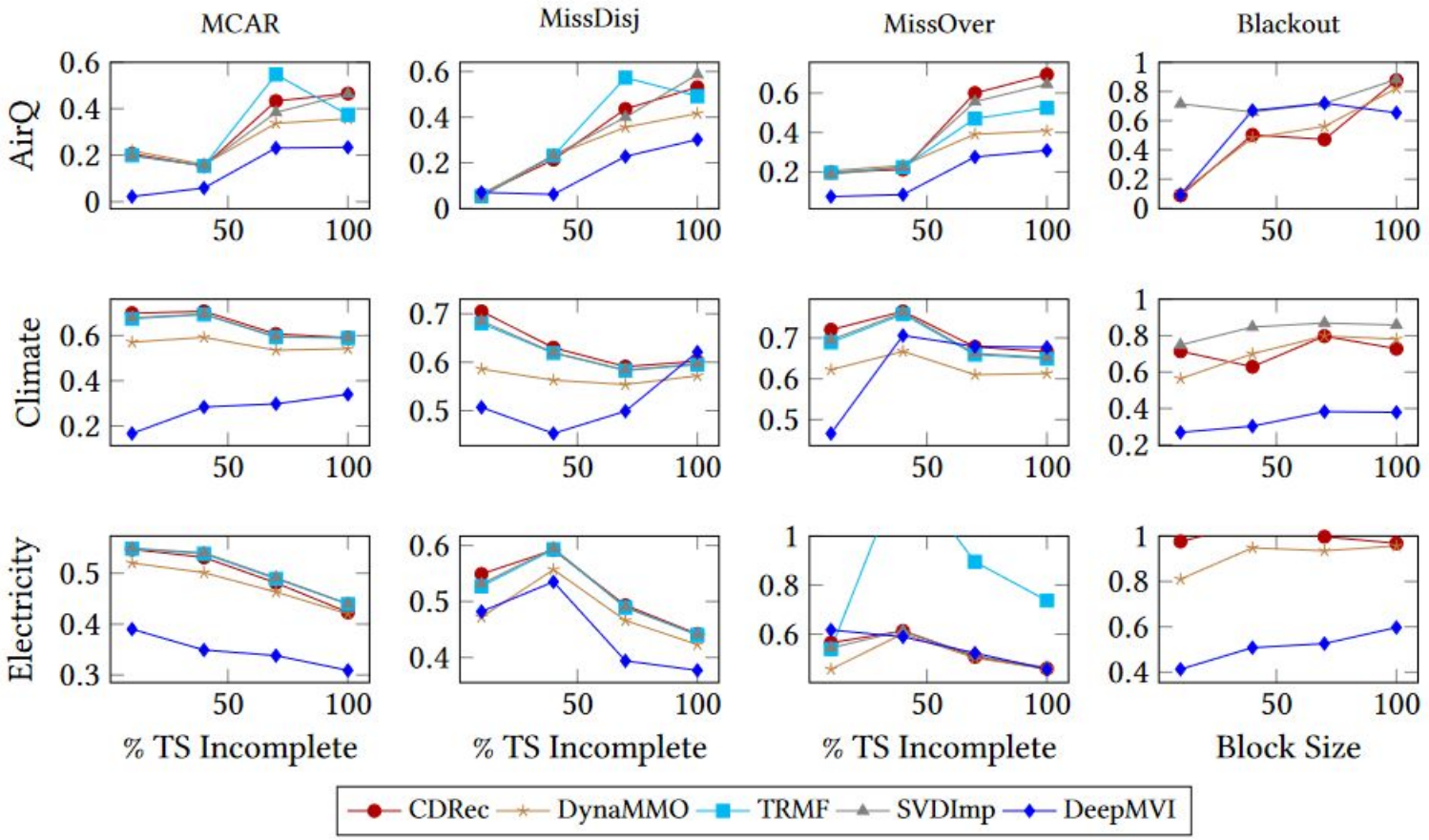


Figure 6: Mean Absolute Errors (y-axis) on three datasets along rows and under four missing scenarios along columns. X-axis is percent of time-series with a missing block for MCAR, MissDisj, MissOver and size of the missing block for Blackout.

x axis is percentage of the time series having missing values

Quantitative Results : Deep Learning Methods

- Best method on OLAP data (Walmart M5, JanataHack)
- Not better than BRITS on MCAR scenario
- Significantly better results on Blackout
  - The robust training procedure of DeepMVI helps in same
  - BRITS training procedure already supports Blackout
    - Doesn't have dependency on scalars on the same time step

Model	Walmart M5	JantaHack	Climate	
	MCAR	MCAR	MCAR	Blackout
BRITS [4]	0.69	0.22	0.26	0.69
GPVAE[8]	0.60	0.28	0.43	0.81
Transformer [25]	0.56	0.24	0.29	0.67
DeepMVI(Ours)	0.53	0.16	0.28	0.38

Qualitative Results

- Top Row is MCAR
- Bottom Row is Blackout
- Different blocks are different time ranges imputed
- Conventional methods completely fail on Blackout Scenario
  - Just interpolation between the endpoints

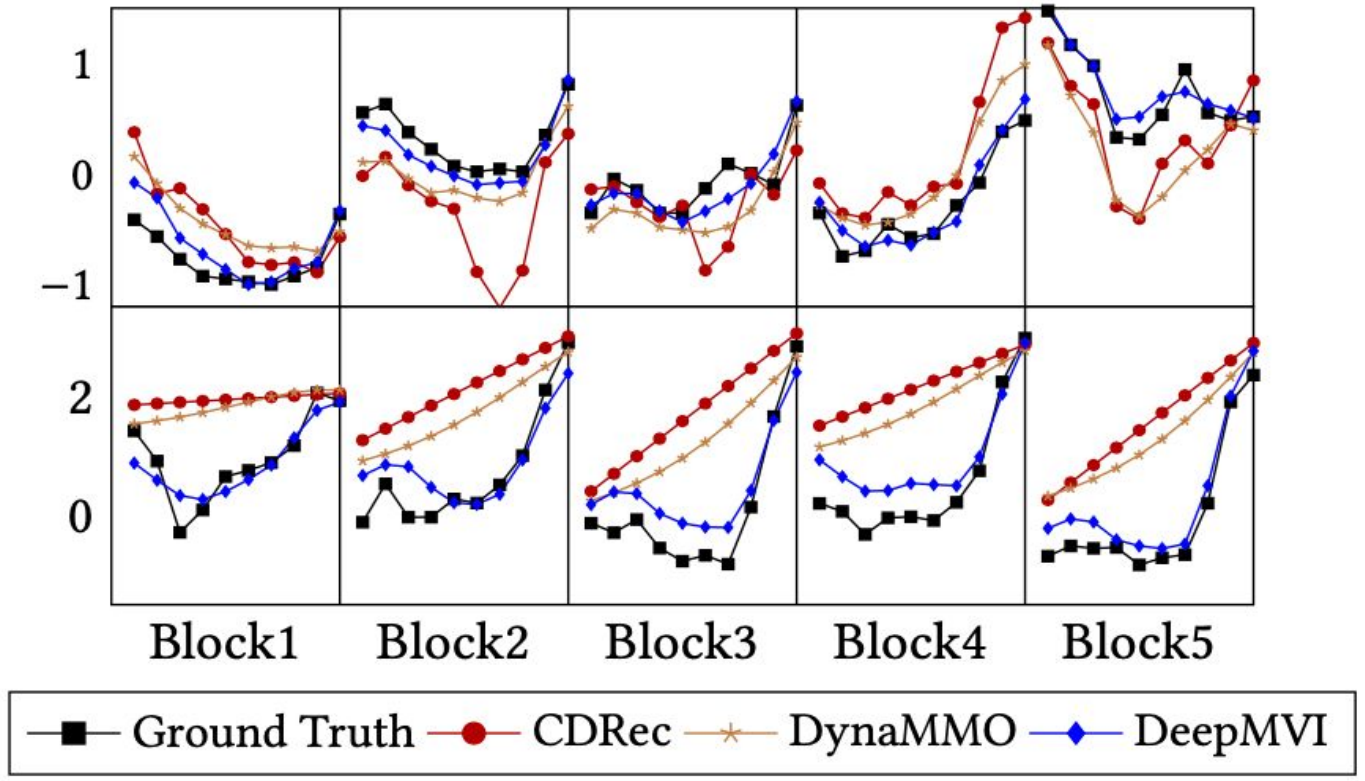
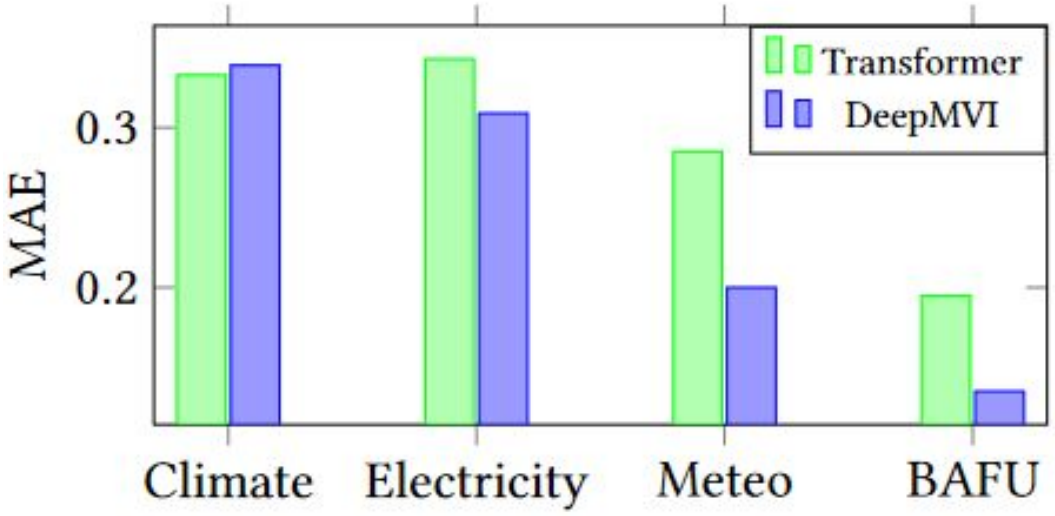


Figure 4: Visualised Imputations on Electricity Dataset. The top row shows MCAR missing blocks while the bottom rows is for Blackout scenario.

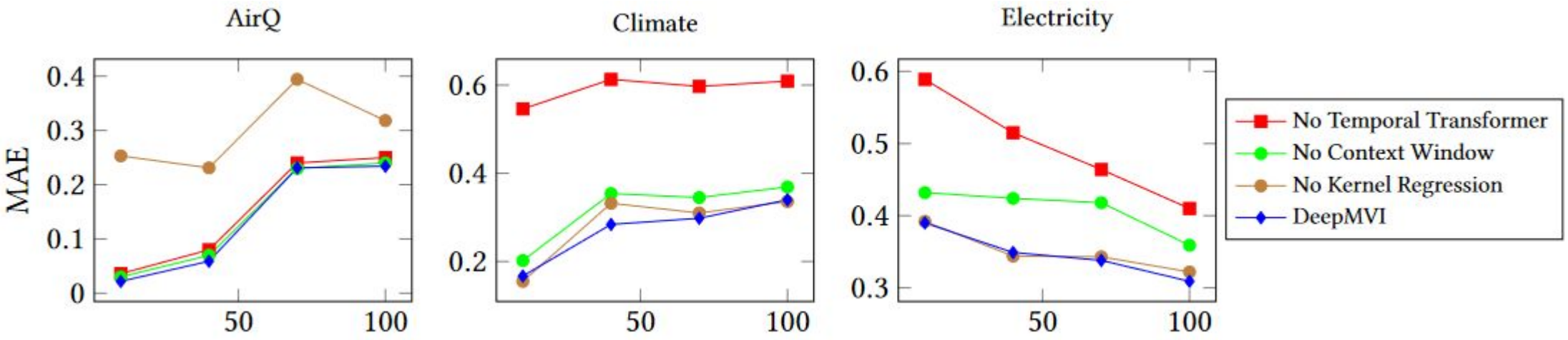
Ablation Studies – Windowing Helps

- Comparison with Vanilla Transformer model
  - No convolutions. Attention on Scalars
- Apart from Climate (strongly periodic) all datasets benefit from windowing
  - Just positional information sufficient for strongly periodic series



Ablation Studies

- We have a system able to do combine both **Across Time series** and **Temporal Patterns**
- AirQ : Strong across series correlation, Need Kernel Regression
- Climate : Periodic series, Need Temporal, Contextual Query not needed
- Electricity : Temporal Patterns, Non-periodic, Contextual Query helps



Downstream Analytics

- Consider an aggregate query
  - Sales average over different stores
- In case of missing data
  - We might just average over the available stores : DropCell
- y axis is difference in MAE using DropCell vs Aggregating after Imputation

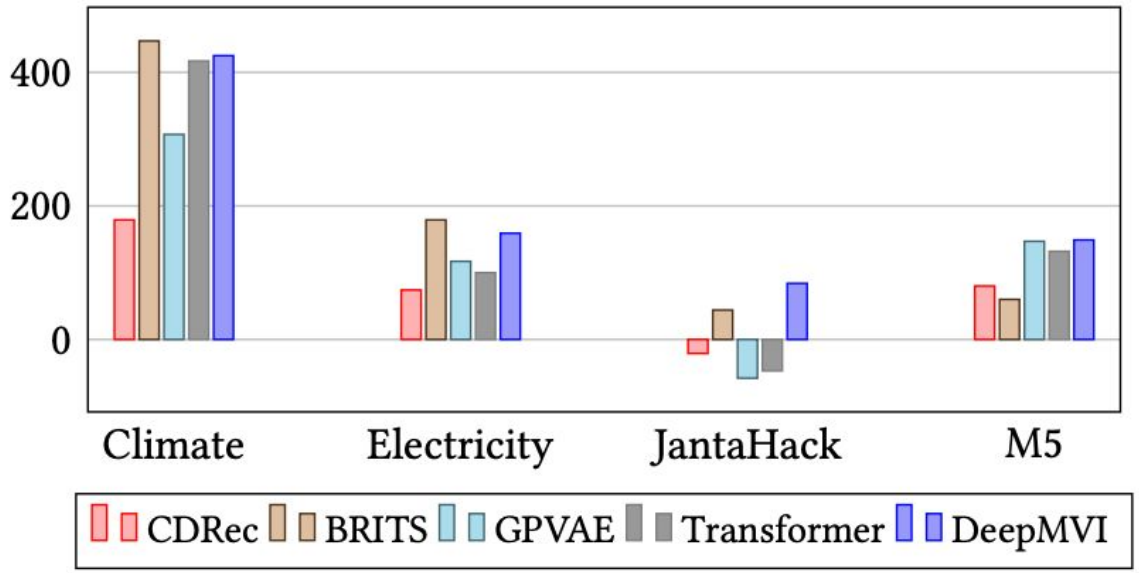


Figure 11: Difference between Mean Absolute Error of Drop-Cell and each algorithm is on y-axis, and four other datasets on x-axis.

Other Works

- Prior exploration includes problems around Outlier detection
  - Used insights from the analyses for developing DeepMVI

- Currently working on Joint Probabilistic Long-Range Query Forecasting
  - Extends DeepMVI design principles to forecasting setting
  - Incorporate developments in Probabilistic Joint Models