Visual Interpretation of Recurrent Neural Network on Multi-dimensional Time-series Forecast

Shen, Qiaomu; Wu, Yanhong; Jiang, Yuzhe; Zeng, Wei; Lau, Alexis K.H.; Vianova, Anna; Qu, Huamin

DOI
10.1109/PacificVis48177.2020.2785

Publication date
2020

Document Version
Accepted author manuscript

Published in
2020 IEEE Pacific Visualization Symposium, PacificVis 2020 - Proceedings

Visual Interpretation of Recurrent Neural Network on Multi-dimensional Time-series Forecast

Qiaomu Shen[1]*, Yanhong Wu[2]†, Yuzhe Jiang[1]‡, Wei Zeng[3]§, Alexis K H LAU[1]¶, Anna Vianova[4]‖, and Huamin Qu[1]**

Hong Kong University of Science and Technology[1], Visa Research[2],
Shenzhen Institutes of Advanced Technology[3], Delft University of Technology[4]

ABSTRACT

Recent attempts at utilizing visual analytics to interpret Recurrent Neural Networks (RNNs) mainly focus on natural language processing (NLP) tasks that take symbolic sequences as input. However, many real-world problems like environment pollution forecasting apply RNNs on sequences of multi-dimensional data where each dimension represents an individual feature with semantic meaning such as $PM_{2.5}$ and SO_2. RNN interpretation on multi-dimensional sequences is challenging as users need to analyze what features are important at different time steps to better understand model behavior and gain trust in prediction. This requires effective and scalable visualization methods to reveal the complex many-to-many relations between hidden units and features. In this work, we propose a visual analytics system to interpret RNNs on multi-dimensional time-series forecasts. Specifically, to provide an overview to reveal the model mechanism, we propose a technique to estimate the hidden unit response by measuring how different feature selections affect the hidden unit output distribution. We then cluster the hidden units and features based on the response embedding vectors. Finally, we propose a visual analytics system which allows users to visually explore the model behavior from the global and individual levels. We demonstrate the effectiveness of our approach with case studies using air pollutant forecast applications.

Index Terms: interpretable machine learning, recurrent neural networks, multi-dimensional time series, air pollutant forecast

1 INTRODUCTION

Recurrent neural networks (RNNs) have been widely applied in various natural language processing (NLP) tasks such as machine translation and sentiment analysis. Benefiting from the capacity to model sequential data, RNNs have been extended to other domains of sequential data beyond NLP, such as weather forecast [32] and air pollutant prediction [22]. Compared to NLP tasks that take latent embeddings as inputs, the input features in these applications are usually multi-dimensional time series where each dimension has its own semantic meaning. For example, in air pollutant forecast, RNN models are widely adopted by domain experts where input sequences are hourly recorded series of high-dimensional pollutants (*e.g.*, SO_2) and meteorology features (*e.g.*, *wind speed*).

Despite the competitive performance of RNNs, the lack of understanding of their internal mechanisms makes the models untrustworthy and further limits their extension to other domain applications. During model development, the domain experts usually want to better understand the models' forecast. They tend to learn whether the model's behavior confirms any hypotheses according to existing domain knowledge, which helps gain confidence in prediction for decision making. On the other hand, the experts also aim to identify the patterns that have not been observed before by exploring different cases to enrich their knowledge of the domain problem. In addition, understanding RNNs also helps model designers to choose appropriate model architecture and hyper-parameters.

Existing work on RNN interpretation such as LSTMVis [29] and RNNVis [20] mainly focuses on NLP tasks. Both of these two work interprets hidden units by providing their relevant words as language contexts. However, existing techniques cannot be directly applied in RNNs for high-dimensional time-series forecasts. First, the high dimensionality of the input sequence makes it difficult to discover relationships between features and hidden states. Since different features at various timestamps are correlated, traditional techniques are not applicable as they are not designed to reveal how feature importance changes over time, which hinders the domain experts' ability to analyze if the prediction is consistently generated. Analyzing this complex temporal multi-dimensional data requires an effective visualization design that can reveal both cross-dimension data relationships and the model's temporal behavior. Moreover, in real-world applications, the input dimension size may be from hundreds to thousands, and the hidden unit size of the RNN models can also be large. This requires the visualization methods to have good scalability in order to demonstrate the distribution and complex relationships of hidden units and input features.

To address these challenges, we propose MultiRNNExplorer, a visual analytics system to help domain experts understand RNNs in high-dimensional time-series forecasts from two aspects: model mechanism and feature importance. To understand model mechanism, we estimate the overall response from hidden units to features and generate response embedding for both hidden units and features. We then cluster the hidden units and features respectively to reveal the high-level knowledge captured by the models. To measure the feature importance, we calculate the gradient for all features at all timestamps with respect to a given case. We then design a visualization interface with coordinated views, enabling users to interactively explore model behaviors.

Our contributions can be summarized as follows:

- A new method for estimating neuron response to multi-dimensional time-series data by measuring hidden unit response distribution on different value ranges of input features.
- A visual analytics system that helps users to explore, interpret, and compare RNNs on multi-dimensional time-series data.
- Several case studies on air pollutant datasets to demonstrate the effectiveness of the proposed system and the insights revealed during exploration.

*e-mail: qshen@ust.hk
†e-mail: yanwu@visa.com
‡e-mail: yuzhe.jiang@ust.hk
§e-mail: wei.zeng@siat.ac.cn
¶e-mail: alau@ust.hk
‖e-mail: A.Vilanova@tudelft.nl
**e-mail: huamin@ust.hk

2 RELATED WORK

2.1 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) is a class of neural networks that contains feedback connections [11]. Compared with fully connected neural networks, this architecture is capable of processing temporal data and learning sequences. Many RNN variants have also been developed in the past decades. One typical work is Long-Short Term Memory (LSTM) [12]. By integrating several gate functions and appending an additional cell state to the vanilla RNN, LSTM avoids the gradient vanishing problem when processing the long sequences. A similar but computationally more efficient approach is the Gated Recurrent Unit (GRU) [6], which simplifies the model architecture by using only two gate functions: update gate and reset gate. To better utilize more data along the sequence, the bidirectional RNN [27] is proposed to capture both past and future information to improve prediction.

RNN-based models have not only been proven successful in performing Natural Language Processing (NLP) tasks, but have also been adopted in broader domains such as weather forecasting [32] and environmental factor prediction [5]. For example, Oprea et al. proposed an RNN-based architecture to forecast $PM_{2.5}$ air pollutants [22]. Compared with the NLP domain, these critical areas not only require good model performance but also need humans to understand how a prediction is generated and whether the results are trustable for guiding further decision makings [16]. However, understanding RNNs in such scenarios is challenging. On the one hand, as each input dimension is usually informative and usually indicates an interpretable factor compared with word embeddings used in NLP, analyzing which features have the most impact on prediction results is important. On the other hand, users also need to explore and understand how the hidden states evolve over time in order to reveal the underlying working mechanism of the RNN along the sequence. We propose MultiRNNExplorer to better understand and interpret RNN models, especially when the model input is multi-dimensional as with the above critical areas.

2.2 Machine Learning Interpretation

In recent years, many machine learning interpretation methods have been developed, which can mainly be categorized into two groups: model reduction and feature contribution.

Model reduction methods usually learn a surrogate model to approximate the original complex model. The surrogate model is usually simple and interpretable, such as linear regression [25] and decision trees [7]. Depending on the ways of approximating the original model's behaviors, there are three main ways to conduct model reduction: decompositional, pedagogical, and eclectic [2]. Decompositional methods are usually model dependent and simplify the original model structure, such as the layer and weights of the neural network. Pedagogical methods only utilize the input and output information to mimic the original model. Eclectic methods are either a combination of the previous two approaches or are distinctively different from them. Though model-reduction-based methods are flexible and easy to understand, it is questionable whether or when the surrogate model truly reflects the original model's behaviors. We thus discard this approach in our work.

Sensitivity analysis methods help users understand the relationships between input features and output prediction. They usually assign each feature an importance score to indicate how it impacts the final prediction. One classical work is Partial Dependence Plot (PDP) [9], which depicts how feature value changes affect predictions. A recent work, SHAP [19], also calculates feature attribution, but from a local perspective. SHAP is based on the ideas of Shapley Values from game theory, which calculates the feature sensitivity of a data instance by comparing it with a set of reference data points. Compared with global solutions, it is more consistent and locally accurate. One major limitation of sensitivity analysis methods is is that they are computationally expensive.

Apart from the above work that focuses on describing the input-output relationship, some work focuses on understanding and analyzing the internal working mechanism of RNNs, especially the hidden layer behaviors, by utilizing visualization techniques. Karpathy et al. first conducted some explorative studies on hidden cell activation on different sequence items [13]. LSTMVis [29] further analyzed and compared the activation changes of each individual cell to demonstrate that different cells may capture different language patterns over time. Ming et al. developed RNNVis [20] to depict the co-clustering patterns between hidden states and input words. There is also some other work that focuses on a particular model or domain. For example, Seq2Seq-Vis [28] mainly focuses on the sequence-to-sequence model and RetainVis [14] enables experts to analyze electronic medical data using a RNN model. Though these methods show how the relationships between hidden units and input data change over time, they fail to capture the importance evolution of each individual input dimension, which prevents them from being adopted in critical areas such as finance and environmental factor forecasting. We aim to solve this problem by revealing the activation sensitivity of hidden units to different features and highlighting what features mostly affect the final prediction the most over time.

3 Application and Models

3.1 Application

In this paper, we focus on a particular type of regression task on multi-dimensional time-series data: air pollutant forecast of **target pollutants** at **target locations**. We collaborate with a group of domain researchers who use RNNs to conduct air pollutant forecasting. Specifically, they choose 16 air quality monitoring stations in Hong Kong with the stations' locations as the **target locations** and select $PM_{2.5}$, PM_{10}, NO_2, SO_2, and O_3, the five air pollutants which effect the human's health the most, as the **target pollutants**. The ML models are trained to predict these target air pollutants at each station. Though the RNNs can provide high accuracy, researchers are more interested in understanding why models make certain predictions so that they can decide whether to trust the models for decision making. We thus develop MultiRNNExplorer to help the domain researchers explain the RNNs' behavior on their temporal multi-dimensional air pollutant dataset.

3.2 Data Description

The dataset includes hourly observations of the air pollutant and meteorology data from weather and air quality monitoring stations in Hong Kong from Jan. 01, 2015 to Dec. 31, 2018. The features are listed in Table 1.

Table 1: There are two types of features taken as input: air pollution and meteorology.

Category	Feature type
Air pollutant	$PM_{2.5}$, PM_{10}, NO_2, NO_x, SO_2, CO, O_3
Meteorology	*Wind Speed(Wind), Wind Direction(WD), Dew Point(DP), Relative Humidity(RH), Temperature(Temp), Sea Level Pressure(SLP), Station Pressure(SP), Cloud Cover(CC)*

Similar to other work on air pollutant forecasts [33], when forecasting the air pollutants at a target location, we divide the regions around the target location into different spatial partitions and aggregate the data observed by all the stations within each group in order to generate features. Specifically, as shown in Fig. 5B, we divide the nearby regions into five non-overlapping rings centered at the targeted location with radii of 10km, 30km, 100km, 200km, and 300km. Each ring is further divided into 8 sectors with the same angle such that the whole area around the target location is partitioned into 40 sub-regions. For each sub-region, we use the air pollutant and meteorology data observed by the monitoring stations located in the corresponding sub-region (including

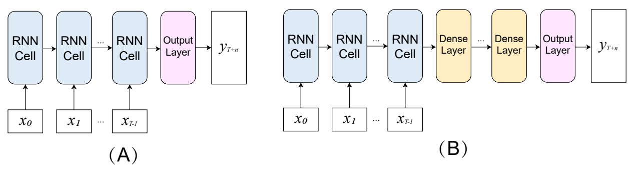

Figure 1: RNN Architectures considered in our experiments: A) RNN: the RNN layer is directly connected to the output layer; B) RNN-Dense: adding dense layers between the RNN layer and the output layer.

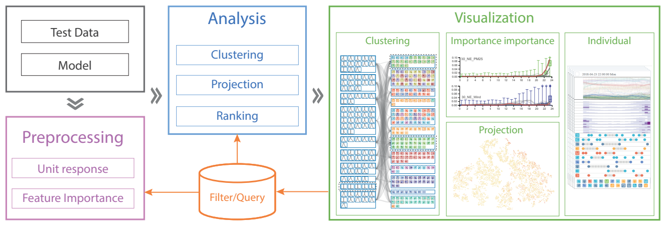

Figure 2: The system overview that includes three major modules: Preprocessing module, Analysis module and Visualization module.

the 16 target stations) as the features. If there are multiple monitoring stations in a sub-region, we aggregate their data and use mean values as the features. In this way, each feature can be identified as a triplet $(distance, direction, feature\_type)$; for example, $(10km, E, NO_2)$ indicates NO_2 concentration observed at *10km* away *East* of the target location. For each time step, we use a vector to represent all the features where each dimension indicates a feature triplet. In this way, the data within a time peroid can be represented as a multi-dimensional sequence.

3.3 Models Description

RNNs take a sequence as input with fixed length T: $\{x_0, x_1, ..., x_{T-1}\}$ and predict the value at timestamps equal to or greater than T, where $x_t \in \mathbb{R}^m$. A hidden state h_t is updated according to the input of timestamp t and previous hidden state h_{t-1}. In vanilla RNNs, the hidden states are updated by:

$$h_t = \sigma(Wh_{t-1} + Vx_t) \quad (1)$$

Where W and V are weight matrices and σ is the *tanh* function which constrains the output of the hidden states to $(-1, 1)$. We consider two types of architectures shown as Fig. 1. **RNN**: the final timestamp hidden state is directly connected to the output layer (Fig. 1A). **RNN-Dense**: there are several dense layers between the final timestamp and output layer (Fig. 1B). In addition to vanilla RNNs, we also consider two variants: GRU and LSTM, which mitigate the gradient vanishing issue and enable the models to memorize long-term information by adding the "gating" mechanism. In our application, these models take the historical data discussed in Sec. 3.2 as input and output the predicted concentration of target pollutants in the future.

4 System Design

In this section, the requirement and tasks are discussed. Over the past 12 months, we closely worked with two domain researchers in urban air quality analysis and forecasting. One researcher (R1) studies the atmospheric diffusion of air pollutants and has interest in what machine learning model learns. The other researcher (R2) is working on air pollutant forecasts through machine learning techniques.

4.1 Task analysis

We distill three general goals: G1: Understand the RNN model behavior/mechanism in high-dimensional forecasts. G2: Understand the feature importance. G3: Support case-based exploration. To fulfill the these analytical goals, we specify the following tasks:

T1: Encode hidden state statistics. Hidden states, a direct reflection of a model's intermediate results, are critical for revealing the information captured by a model (**G1, G2**). Visualizing hidden state statistics can provide a holistic picture of a model's behavior.

T2: Measure feature importance at multiple scales. The visual analytics system should allow users to explore feature importance at different scales(**G2**). For example, the overview level presents feature importance summarized from the whole dataset and the individual level focuses on the importance of a single case.

T3: Analyze the response between features and hidden states. Measuring the response relationship between features and hidden states is the key factor in revealing what patterns are captured by the model (**G1**). Targeting at the complicated many-to-many relationship, the hidden states as well as the features should be clustered to alleviate the burden on end users.

T4: Support temporal analysis. One major advantage of RNNs is that they can capture time-dependent sequence information. Showing what information is preserved along the sequence and or discarded helps users better understand how the temporal information is utilized by the model(**G2, G3**). In addition, users can identify the critical time steps that may cause a significant change in prediction.

T5: Identify data clusters and outliers. To support case-based reasoning, users need to first obtain a data overview by identifying the data clusters and outliers (**G1, G3**). This provides concrete examples to guide users in further exploring the data of interests. Users can also inspect the outliers that have distinct prediction results to detect if the model behaves incorrectly according to certain domain knowledge.

T6: Explore the case-based model behaviour. The system needs to enable the users to explore how a model behaves for individual cases such as build the correlation between feature trend and feature importance trend(**G3**). Since hundreds of temporal features are taken as input for each case, the effective summary is required.

4.2 System Overview

We implement MultiRNNExplorer as a web-based system using Flask, VueJS, and D3. The system consists of three modules: 1) preprocessing, 2) analysis, and 3) visualization. With chosen models, the preprocessing module generates the raw data that need to be analyzed, including estimating the response relationship and feature importance. We then apply various data analysis techniques such as clustering, projecting, and ranking in the analysis module to provide the data structure required by the visualization module. The visualization module integrates coordinated views to support interactive interpretation of and reasoning about the model behavior at different perspectives.

5 Model Interpretation

This section first describes how we analyze the relationships between features and hidden states (**T3**). Specifically, we propose an efficient method to calculate how sensitive each hidden state is to certain feature changes and apply a clustering method to group response relationship patterns for better scalability. This provides users an overview on how the model categorizes different features and perturbing features to what value ranges may largely affect model behaviors. We also introduce a gradient-based method to identify the most important features that can impact the prediction over time (**T2, T4**). This provides another perspective on analyzing how feature importance changes along the sequence. These two approaches are complementary to each other in enabling users' understanding and explaination of model behaviors.

5.1 Relationships between Hidden States and Features

To measure how feature changes can affect hidden states (**T4**), one common approach is perturbating feature values and measuring how the hidden state distribution changes compared with the original data. However, perturbation-based methods are usually time-consuming and not applicable when different features are correlated. Inspired by [30], we adopt another method that directly compares the hidden state distributions of different feature value ranges. This approach is computationally efficient and provides a good approximation for whether a hidden state is sensitive to feature changes. This section introduces how we generate the hidden state distribution for different value ranges of each feature and how we quantitatively measure the relationship strength between features and hidden states based on the generated distribution.

5.1.1 Hidden State Distribution

As discussed in Sec. 3.2, the model input is a sequence of features $X = \{x_0, x_1, ..., x_{T-1}\}$ where x_t indicates a multi-dimensional feature vector at time step t. Each feature dimension is denoted as x_t^f in which f represents a feature triplet $(distance, direction, feature\_type)$ at time step t. Similarly, we use $H = \{h_0, h_1, ..., h_{T-1}\}$ to indicate the hidden state sequences at different time steps where $h_t = \{h_t^0, h_t^1, ..., h_t^{D-1}\}$ indicates the hidden state distribution at time step t and D denotes the hidden unit size. As h_t is computed by feeding x_t into an RNN model L, we denote $h_t = L(x_t)$. Considering a dataset $\mathbb{X} = \{X_0, X_1, ..., X_{N-1}\}$ consisting of N sequences, we can collect a feature vector set $\mathbb{V} = \{x \mid x \in X, X \in \mathbb{X}\}$ where $|V| = N \times T$. Based on the value ranges of a feature f, we can further divide $\mathbb{V}$ into different groups $\mathbb{V}_g^f = \{x \mid \Theta_g^{lower} \leq x^f < \Theta_g^{upper}, x \in \mathbb{V}\}$ where Θ_g^{lower} and Θ_g^{upper} denote the feature range thresholds of a group g. In this paper, we set the number of groups to be 3 where the thresholds are the 25^{th} and 75^{th} percentiles of each feature. We denote these three groups as $\mathbb{V}_{perc<0.25}^f$, $\mathbb{V}_{0.25 \leq perc<0.75}^f$, and $\mathbb{V}_{perc \geq 0.75}^f$. As we can obtain the hidden states by feeding the data into the RNN model, we can compute the corresponding hidden state set $\mathbb{H}_g^f = \{L(x) \mid x \in \mathbb{V}_g^f\}$ for a feature group $\mathbb{V}_g^f$. In this way, the distribution of the j^{th} hidden unit for feature group $\mathbb{V}_g^f$ can be denoted as $\mathsf{H}_g^{j,f} = \{h^j \mid h \in \mathbb{H}_g^f\}$. Measuring the distribution of $\mathsf{H}_g^{j,f}$ enables us to compare the outputs of different hidden units when a feature value falls into a certain range and infer if these hidden units are sensitive to feature value changes. For example, Fig. 3 shows the distribution of the 92^{th} and 93^{th} hidden units for feature $PM_{2.5}$ and SO_2 respectively. We can see that the 92^{th} hidden unit has distinct distributions for different value ranges on feature $PM_{2.5}$. Meanwhile, for feature SO_2, the distributions look identical. This indicates that the 92^{th} hidden unit is more sensitive when the value of $PM_{2.5}$ changes compared with SO_2. Similarly, we can observe that the 93^{th} hidden unit is more sensitive to SO_2 changes, which indicates that different hidden units can capture distinct feature patterns.

5.1.2 Relationship Strength Estimation

We estimate the relationship strength of a hidden unit with a feature by measuring the distances between the hidden unit distributions of different feature value ranges. To measure distribution distance, we apply Two-sample Kolmogorov Smirnov (KS) statistics which can be presented in following formulation:

$$KS(S1, S2) = max_{sup_x}(|F_{S_1}(x) - F_{S_2}(x)|) \tag{2}$$

where the sup_x is the supremum of the set of distances, F_{S_1} and F_{S_2} are the cumulative empirical distribution functions of the first and the second sample respectively, and sup is the supremum function. Given significance level α (generally 0.05) the null hypothesis of

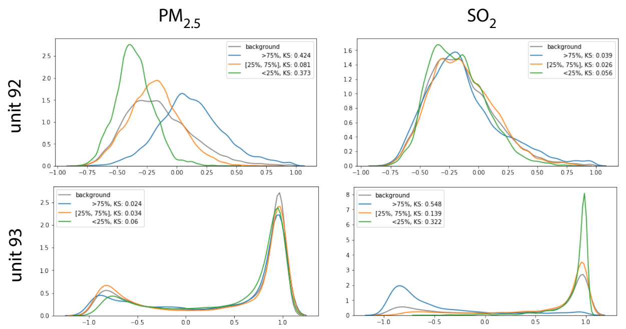

Figure 3: Compare the response of hidden units(92 and 93) to features ($PM_{2.5}$ and SO_2).

two samples having different contributions, the reject co-efficient can be calculated as follows:

$$Rej(S1, S2) = c(\alpha)\sqrt{\frac{|S1| + |S2|}{|S1||S2|}}, c(\alpha) = \sqrt{-\frac{1}{2}\ln\alpha} \tag{3}$$

Based on the KS statistics, the distance between two samples can be measured as follows:

$$Dis(S1, S2) = \begin{cases} KS(S1, S2), & \text{if } KS(S1, S2) > Rej(S1, S2) \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

To quantitatively measure the relationship strength between a hidden unit and a specific input feature, we compare the hidden unit distribution of data in different feature ranges with the distribution of all the data. A larger difference indicates a stronger relationship as the hidden unit will generate different values when the feature value changes. As shown in Fig. 3, the ks-statistics of unit 92-$PM_{2.5}$ and unit 93-SO_2 are significantly larger than the other two combinations, indicating the statistics can effectively measure the distribution difference. Specifically, the relationship strength between the j^{th} hidden unit and feature f can be measured as the maximum ks-statistics among all different feature selections:

$$\begin{aligned} RS(j, f) = max(& Dis(\mathsf{H}^{j,f}, \mathsf{H}_{perc<0.25}^{j,f}), \\ & Dis(\mathsf{H}^{j,f}, \mathsf{H}_{0.25 \leq perc<0.75}^{j,f}), \\ & Dis(\mathsf{H}^{j,f}, \mathsf{H}_{perc \geq 0.75}^{j,f})) \end{aligned} \tag{5}$$

5.2 Hidden Unit and Feature Clustering

Another major challenge for interpreting RNN models on multi-dimensional sequential data is scalability. RNN models usually contain hundreds to thousands of hidden units for each layer, which makes it ineffective to display the activation distribution of every hidden unit to users. To address this challenge, previous work on visual interpretation of machine learning models usually use clustering [17, 20] or sampling [24] techniques to reduce the number of visual elements displayed. In this work, we choose clustering methods over sampling since clustering can better preserve the hidden units' response relationship to features. It also provides a good summary of the knowledge that the model learned.

With the measurement of unit response, we can generate a 2D table with the size of $D \times M$, where D and M are the size of hidden units and features respectively. The cell of j^{th} row and k^{th} columns is the response of hidden unit h^j to feature f^k: $RS(j, f^k)$. Then we can define the response embedding vector for both features and hidden units. For any feature f^k and hidden unit j, the response embedding vectors are $vec_{f^k} = [RS(0, f^k), RS(1, f^k), ..., RS(D-1, r^k)]$ and

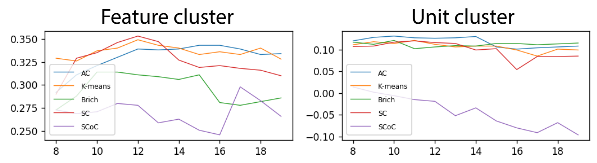

Figure 4: Cluster score with different cluster number. Left: feature cluster. Right: hidden unit cluster. The horizontal axis represents the cluster number, the vertical axis represents the cluster score.

$vec_j = [RS(j, f^0), RS(j, f^1), ..., RS(j, f^{M-1})]$, which are the specific columns and rows respectively.

To analyze the relationship between hidden units and features, Yao et al. [20] used a bipartite graph to model the many-to-many relationship and used co-clustering algorithms [8] to group hidden units and input features simultaneously. We test co-cluster techniques: Spectral Co-clustering(SCoC) as well as other techniques including Agglomerative Clustering(AC) and Spectral Clustering(SC) on our dataset. The clustering methods other than SCoC take response embedding vectors as input to cluster features and hidden units respectively. To rank the performance of different clusters with different cluster numbers, we use the Silhouette Coefficient [26] to evaluate the quality of the clusters. Silhouette Coefficient ranges from -1 to +1, with higher values of this coefficient meaning the cluster quality is more appropriate.

Fig. 4 shows cluster quality for features (left) and hidden units (right). We found that the Spectral Co-clustering method has a low Silhouette Coefficient score because it keeps creating a one-to-one relationship between the feature cluster and the hidden units cluster. In this case, it can be found that Agglomerative Clustering with cluster number of 12 and K-Means with cluster number of 10 show the best performance for feature and hidden units respectively. With the Silhouette Coefficient, our system can automatically choose the clustering algorithms and cluster number. Users can also manually choose different clustering algorithms and change the number of clusters based on their analysis requirement.

The clustering results can be modeled as bipartite graph $\mathbb{G} = (\mathbb{V}_H, \mathbb{V}_F, \mathbb{E})$, where $\mathbb{V}_H$ is the hidden unit cluster set and $\mathbb{V}_F$ is the feature cluster set. $\mathbb{E}$ indicates the weighted edge set between unit clusters and input dimension clusters with the weight of $E_{H,F} = \sum_{h \in H} \sum_{f \in F} RS(h, f)$ where $H \in \mathbb{V}_H$ and $F \in \mathbb{V}_F$. This bipartite graph of features and hidden units can help users understand the information captured by different hidden unit clusters by examining which feature clusters have strong relationships with them.

5.3 Local Feature Importance

Inspired by back-propagation in machine learning, we conduct the individual level analysis based on the local gradient which is used to present the word saliency in NLP tasks [15]. Given the output of feature y^l, we use the local gradient with respect to feature $x_t^k \in x$ to present the feature importance as:

$$w(y^l, x_t^k) = |\frac{\partial(y^l)}{\partial(x_t^k)}| \tag{6}$$

The absolute value of gradient $w(y^l, x_t^k)$ indicates the sensitiveness of x_t^k to the final decision of y^l with the given input sequence of x. This measurement shows how much the specific feature at a specific time contributes to the final output [15]. However, for input x with the length of T, the total number of all feature importance scores is $N \times T$ which causes difficulty in showing the overview. To address this challenge, we leverage the clustering result from Sec.5.2 and define the cluster importance of features as:

$$W(y^l, H_t^i) = \sum_{x_t^k \in H_t^i} |w(y^l, x_t^k)| \tag{7}$$

Thus, the size of the cluster importance for all timestamps is $C \times T$ where C is the number of clusters ($C < N$).

6 Visualization Design

In this section, we introduce the visual design based on the design tasks discussed in Sec.4.1. As shown in Fig. 5, the visual analytic system consists of six coordinated views. Starting from the configuration panel Fig. 5B, users are able to select the target feature and the model to be analyzed. The region partition will be shown as Fig. 5B after the model is selected. To support exploring the model mechanism, the Cluster View is displayed to summarize the hidden units' response to the features (Fig. 5A) and the Feature Importance View (Fig. 5C) is shown to visualize the temporal importance of each feature. Furthermore, users can select the individual cases in the Projection View (Fig. 5E) and the selected cases are grouped by similarity and displayed in the Individual View (Fig. 5D).

6.1 Cluster View

The Cluster View (Fig. 5A) shows the overview of response relationship (**T3**) between the hidden units and features. The hidden units and features are visualized as the Hidden State Distribution and the Feature Glyph respectively.

Hidden State Distribution. The left column on the Cluster View is the Hidden State Distribution component. As shown in Fig. 6A, each row represents a hidden unit cluster. Each hidden unit in a cluster is represented as a line chart that shows its activation distribution(**T1**). The x-axis represents the hidden unit output ranging from -1 to $+1$ and the y-axis represents the corresponding probability (Fig. 6B). From the line chart, users can observe and compare the activation distribution patterns of different hidden units.

Feature Glyph. The right column of the Cluster View is the Feature Glyph component (Fig. 6C). Similar to the Hidden State Distribution, each row represents a feature cluster in which a glyph (Fig. 6D) represents a feature. This enables users to quickly identify different features and compare the common attributes of multiple features for analysis. As described in Sec. 3.1, we define our usage scenario as air pollution forecasting where each feature has three identifiers: the feature category, the direction, and the distance from the feature to the target location. As a categorical feature, we first use the background color of the feature glyph cell to encode the feature category. Different hues encode different categories, and users can find the color legend at the top of the Cluster View. To intuitively encode target location direction, we draw a line segment starting from the glyph center that has the same direction angle. We also draw a square in the glyph where its radius, which is an appropriate channel to encode numerical values, encodes the distance to the target location.

Interactions. We also support various interactions to allow users to dynamically explore this view. The curves linking the hidden state cluster and feature cluster with the width indicate the response strength (Fig. 6E). Users can also filter the link according to the response strength by adjusting the slider bar. When hovering over a hidden state cluster or a feature cluster, the corresponding links and linked clusters will be highlighted.

In this view, the users can obtain an overview of the response relationship between hidden units and features, for example, we can find that there are not strong links connecting to cluster 8 (Fig. 5 A_3), this may be because that all the hidden units are "weakly" activated in cluster 8.

6.2 Feature Importance View

The feature importance view allows users to explore the feature contribution to model output (**T2**). As discussed in Sec.5.3, with an input case, we are able to measure the importance of a single feature as a sequence of importance scores which correspond to the importance at all timestamps (Fig. 5C).

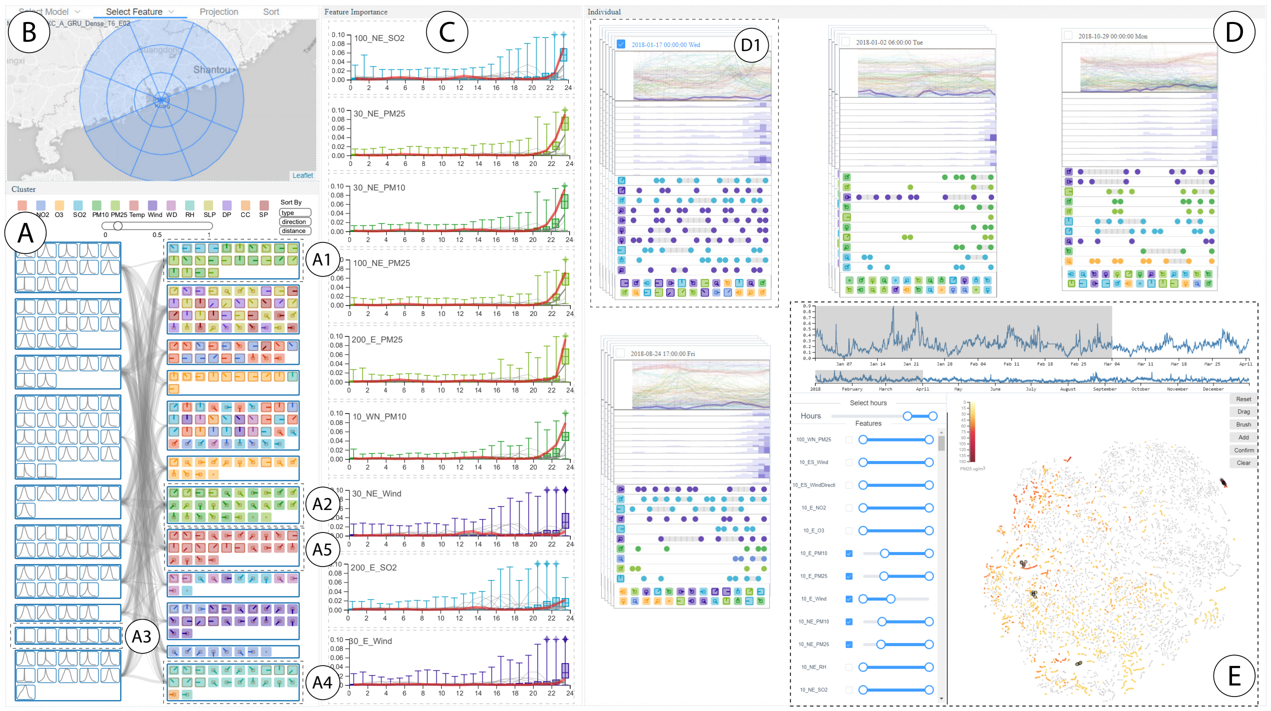

Figure 5: MultiRNNExplorer contains multiple coordinated views to support exploring and understanding RNNs' behaviors on multi-dimensional time-series data, especially on hidden unit response and feature importance. The Configuration Panel (B) allows users to select an RNN model and configure parameters. To reveal model mechanism, the Cluster View (A) summarizes the hidden unit clusters' response to feature clusters, and the Feature Importance View (C) summarizes the temporal importance of input features. The Projection View (E) displays a data overview, allowing users to select sequence instances of interest for further analysis. The selected instances will be shown in the Individual View (D).

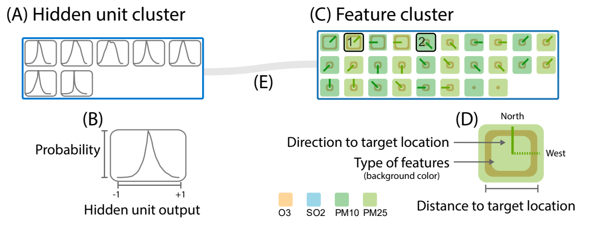

Figure 6: Design of Hidden unit distribution and feature glyph. A) Hidden unit cluster; B) Hidden unit distribution; C) Feature cluster; D) Feature glyph; E) Response link.

Since the importance score only provides a local description for the feature importance, an effective visualization is needed to show an overview of each feature's importance. We choose boxplot for this task since it can present the statistics overview. To show the temporal trend of a feature, we group the importance score of all test cases by the timestamps and make statistics group by group. For the test sequence with a length T, the feature importance charts which contain T boxplots shows the trend of feature importance (Fig. 5C).

The horizontal axis indicates the timestamps and the vertical axis indicates the feature importance score. The top line, upper edge, middle line, bottom edge and bottom line of the boxplot indicate the maximum, 75^{th} percentile, mean, 25^{th} percentile and minimum of the importance scores. Since sometimes the maximum will much larger than the 75^{th} percentile value, which makes the box vary flat and difficult for users to explore the temporal pattern, we limit the maximum score Ms shown in the view. If a boxplot has scores larger than Ms, a diamond symbol appears on the top of the boxplot. The opacity of the diamond indicates the magnitude of the absolute difference between the largest score and Ms.

We also define the overall importance score for a single feature as the sum of the mean score at all timestamps. By default, the boxplot charts will be ranked according to the overall feature importance score. Due to the large number of features, only the top 10 charts are visualized. Users may observe other features by using the scroll bar or filtering the features from the projection view (Fig. 5E).

6.3 Projection View

To help users obtain an overview of case clusters and outliers (**T5**), we design the Projection View (Fig. 5E) which supports various interactions such as zooming and brushing to allow users to select a subset of data for further examination.

In the Projection View, each circle represents a individual case. There are many multi-dimensional reduction methods such as MDS and PCA; we select t-SNE as it can strongly repel dissimilar points and show clusters clearly. For each case, we collect the feature cluster importance over all time steps (discussed in Sec.5.3) as the input vectors of t-SNE. Thus, the positions of the circles reflect the similarity of their cluster importance. We use a sequential color to encode the model's output of each case shown as the legend in Fig. 5E.

Furthermore, to improve the flexibility of the case selection, we add a two-scale timeline (Fig. 5E top) to show the target feature trend, enabling user filtering of the cases by time, and a feature selection component (Fig. 5E left) to filter the cases by feature value.

6.4 Individual View

After observing an overview on data similarities, users may need to drill down to a few individual cases of interest for detailed examination. We develop the Individual View for users to explore and compare the different individuals over time (**T6**).

The selected individual cases are visualized as several stacks of cell as in Fig. 7A. Each cell consists of three components: the Feature Trend Chart (Fig. 7A$_1$), the Cluster Importance Chart (Fig. 7A$_2$), and the Top Feature List (Fig. 7A$_3$). All these three components

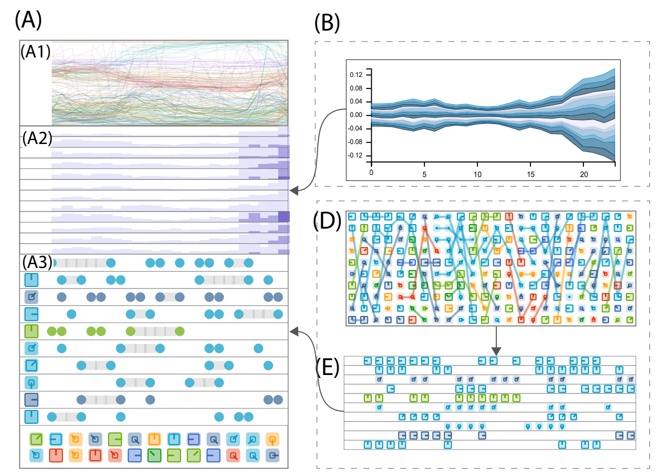

Figure 7: Individual design and the alternative designs. A) Individual View. A1) Feature Trend Chart; A2) Cluster Importance Chart; A3) Top Features Chart. B) themeriver as the alternative design of the Cluster Importance Chart; C) and D) node-like sequence and node sequence as the alternative design of Top Features List.

share the same x-axis which represents the time where time steps increase from left to right.

Feature Trend Chart is a multi-line chart that depicts how different features' values change over time. The y-axis represents normalized feature values where the feature value increases from bottom to top ranging between 0 and 1. Each line represents a feature and the line color encodes feature category the same as in the Cluster View. The corresponding lines are highlighted when hovering on any feature glyph in the Cluster View or hovering on feature importance view.

The middle component is the Cluster Importance Chart that summarizes how each feature group's gradient changes over time. As shown in Fig. 7A$_2$, each feature group is represented as a horizontal bar chart and aligned vertically in the same order as the Cluster View. For a single bar chart, each bar represents the averaged gradient for the corresponding feature group at one time step. We use both the bar height and bar color to encode the gradient value. The first visual channel to encode gradient value is bar height where a greater height indicates a larger gradient value. When the gradient value exceeds a certain limit, instead of further increasing the bar height, we overlay another darker bar where its height indicates the exceeding gradient value for better vertical space efficiency. We have also considered other design choices such as a themeriver (Fig. 7B) in which each colored flow indicates a feature group. However, comparing different feature groups may be difficult and it requires more space when the gradient is large. Thus, we abandon this alternative choice and adopt the current design.

Though the Cluster Importance Chart provides an overview of how each feature cluster's importance changes over time, users still need to link this component to the Cluster View to observe which features are considered important by the model. We design a Top Feature List to visualize the important features over time. Our first design is shown in Fig. 7D. The x-axis represents time and the y-axis represents feature importance rank. The top N features at each time step are visualized as feature glyphs and are positioned vertically according to their importance. We draw links to connect glyphs that represent the same feature in consecutive time steps to enable users tracking same features along time. However, in the discussion with domain experts, this design leads to serious visual clutter due to link overlap when feature ranks frequently change over time. To alleviate users' mental burden in tracking same features and to reduce visual clutter, we develop another design alternative as shown in Fig. 7E. In this design, each feature is represented as a row and aligned vertically with a fixed position on y-axis. We draw its corresponding feature glyphs at the time steps where this feature is ranked among the top N most important features. Thus, users can track a single feature's importance along time by observing how many feature glyphs are drawn on the corresponding rows.

To reduce redundant information, we further simplify this design (Fig. 7A$_3$.) First, instead of drawing duplicate feature glyphs in a row, we draw a grey line segment to mark the time steps that the corresponding feature is among the top N important features. Two colored circles are positioned at the endpoints of a line segment to indicate the starting and ending time steps. The circle color is consistent with the feature glyph color. At last, we only draw a single feature glyph at the beginning of each row to indicate the corresponding feature. To make the Top Feature List space efficient, we only show ten rows by default, and other features are collapsed as feature glyph rows as shown in the bottom at Fig.7A$_3$. Users can click the glyph rows to select different features to analyze.

The three components enable users to observe which features are considered important by the model over time and how the importance is related to feature value changes. Users can also append multiple cells to the Sequence View to compare different sequences side by side. When the number of cells becomes large, we use dbscan to cluster the similar individual cases by the fatten cluster importance(discussed in Sec. 5.3) into one stacked cells with one randomly selected case as the representative case at the top of stack.

6.5 Interactions and Linkage

To better facilitate the interactive exploration of RNNs, our system supports cross-view interactions.

Cross-view highlight. There are three key visualization components appearing across different views: **features**, **feature clusters**, and **cases**. These components are visualized and encoded in different approaches in across views to support various analysis requirements. If one feature is selected, its corresponding visual elements in other views will be highlighted.

Linkage between individual view and feature importance view. When multiple individual cases are selected, the feature importance by time will be visualized as multi-line charts as Fig. 5C shows. When users are hovering over an individual case, the corresponding line-chart will be highlighted.

7 Case Study

In this section, we demonstrate the effectiveness of MultiRNNExplorer in analyzing model behaviors and feature importance. We use the air pollutant data between 2015 to 2017 to train the model and use 8,375 cases in 2018 as testing data for analysis. The models training is conducted on a workstation with 2 × Intel Xeon E5-2650 v4 CPUs and 4 × Nvidia Titan x (Pascal Architecture) 12GB GDDR5X graphics cards. The hyper-parameters, average training time (seconds) and accuracy of different models are listed in Table 2. We demonstrate our system to the domain expert and analyze the trained models on several tasks.

Table 2: Configuration and performance of RNNs, including vanilla RNN, GRU, LSTM, and the RNNs with dense layer (e.g., RNN-Dense). The performance is evaluated by the mean square error (MSE) of $PM_{2.5}$; low MSE represents better performance.

Model	Size	Dense Layer	Time	MSE ($PM_{2.5}$)
Vanilla RNN	100	No	364	5.31 ± 0.98
GRU	100	No	1081	4.32 ± 0.51
LSTM	100	No	1377	4.81 ± 0.31
GRU-Dense	100	3	1387	4.25 ± 0.21
LSTM-Dense	100	3	1525	4.53 ± 0.53

7.1 Changes Over Epochs

To explore the model behavior over the training process, we manually select the RNN model trained after 5, 40, 120, and 200 epochs. Fig. 8 shows the projections and top five most important features at different epochs.

In the Projection View (Fig. 8A), we choose $PM_{2.5}$ as the target feature and use a sequential color schema to indicate the predicted value where a darker color indicates a higher $PM_{2.5}$. At early stages (5th and 40th epochs), we find that the points in a darker color are distributed uniformly in the projection and are mixed together with the points with a light color. This indicates that the cluster gradients are not able to present the distribution of the target feature yet. After training more epochs, the dark points become more concentrated.

In addition, the Feature Importance View (Fig. 8B) shows that the magnitude of the gradient starts from a small value and then keeps increasing in the training process. We also find that in the 5th and 40th epochs, the top five important features are $PM_{2.5}$ and PM_{10} while in the 120th epoch, the feature of wind speed is also ranked in the top five most important features. In the 200th epoch, more features related to wind speed are listed in the top five features. Another finding is that in the 5th epoch, we observe that only the features from the last time steps are considered important while the models at the 40th, 120th, and 200th epochs leverage more timestamps in the forecast. The domain experts indicate that the $PM_{2.5}$ and PM_{10} at nearby locations are the most intuitive features to forecast $PM_{2.5}$ (PM_{10} and $PM_{2.5}$ are always highly correlated). Moreover, the last time step is very important because it is the closest one to the final prediction. Based on these observations, we infer that the features that are directly related with the targeted air pollutant are considered important in the early stage of the training process. After more epochs, the model starts to learn other features that may indirectly influence the forecast, such as the *Wind Speed* and other pollutants (SO_2) shown as Fig. 8B, 20th,200th epochs.

According to official website of United States Environmental Protection Agency (EPA): "SO_x can react with other compounds in the atmosphere to form small particles. These particles contribute to particulate matter (PM) pollution" [1]. We also discussed the reason that SO_2 becomes important later in training with the domain experts. They said it is also possible that the SO_2 relates some industrial activity which results in the high-air pollutant. Moreover, the domain experts explain that the wind speed is important for several reasons: 1) the air pollutants will be blown away if the wind speed is high; 2) since the north and west of the target location have more factories which are the major source of air pollutants, the appropriate wind speed and direction will bring the air pollutants to Hong Kong. Moreover, the data also show different patterns in the Cluster View during the training process. For example, as shown in Fig. 8C, we found that in the 5th epoch almost all three types of features: *Sealevel Pressure*, *Dew-point* and *Station Pressure* are grouped into one cluster. After the 40th epoch, we notice that this cluster is split into two clusters. With these observations, we derive the conclusion that the model gradually learns the high-level knowledge in the training process.

7.2 Understand Model Behaviors

7.2.1 Model Mechanism

This case study is conducted to understand what RNN models learn and to compare different models trained for air pollutant forecasting. With MultiRNNExplorer, we are able to select models at any epoch. Fig. 5A shows the Cluster View of GRU-Dense; by observing the cluster of features, we find that the features with same feature types are likely to cluster together such as the *Relative Humidity* and *Temperature*, are exactly grouped into two separated clusters shown as Fig. 5A$_4$ and A$_5$. Moreover, we find that the $PM_{2.5}$ and PM_{10} are always clustered together as shown in Fig. 5A$_1$ and Fig. 5A$_2$. The

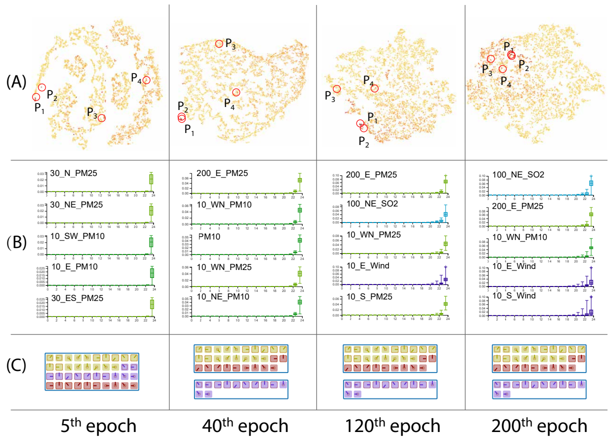

Figure 8: The RNN model shows different behaviors along the training process. A, B, and C show the Projection View, top five important features, and feature clusters respectively at different epochs.

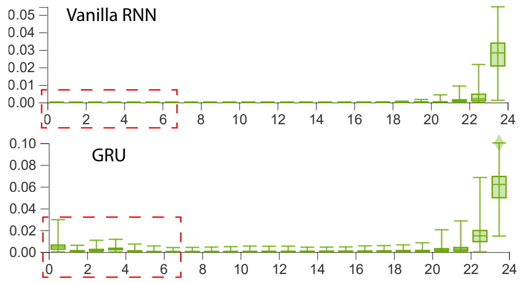

Figure 9: Compare the temporal importance of *100_NE_PM25* across different models.

domain expert explains that $PM_{2.5}$ and PM_{10} are highly correlated because they are always generated together. This shows that the model GRU-Dense learns the information related to spatial locations.

7.2.2 Feature importance

We select the individual sequences of Fig. 5D$_1$ and re-sort the features by the importance. From the Feature Importance View, the top important features are listed as Fig. 5C, and we observe that most of them are related to feature SO_2 and *Wind Speed*. Then, top features change to SO_2 and $PM_{2.5}$ in later epochs. This observation shows that feature importance may vary across different individual cases. By default without selecting any sequences, the features are ranked according to their average importance over of all the test cases. We find that the *Wind Speed* is a major factor that influences the forecast because the wind related features are ranked in front. The domain experts point out that as there are very few factories in Hong Kong, the local emissions are not a major reason that influences the forecast result. Instead, the *PM* pollutants are easily transported from the north, west, and east of mainland China, thus the *wind* plays an important role in the forecast of $PM_{2.5}$ and PM_{10}.

During the exploration of different models, we notice that the temporal pattern of the Feature Importance View is very different across different models. Fig. 9 compares the feature importance for vanilla RNN and GRU. With the selected feature of *100_NE_PM25*, one observation is that the Feature Importance Views of model GRU has a "tail" (Fig. 9) especially for the first three timestamps, which is not observed in vanilla RNN. This solves one question raised by the domain experts: is it necessary to use such complex models like RNN to conduct air pollutant forecasting tasks? It is a controversial question [3] as in some applications all important information is included within within recent small time ranges [10] and do not necessarily require complex machine learning models. By using our system, we can find that the GRUs are able to memorize more

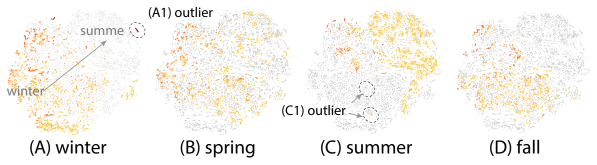

Figure 10: Projection View across four seasons whose time range are defined by local standards.

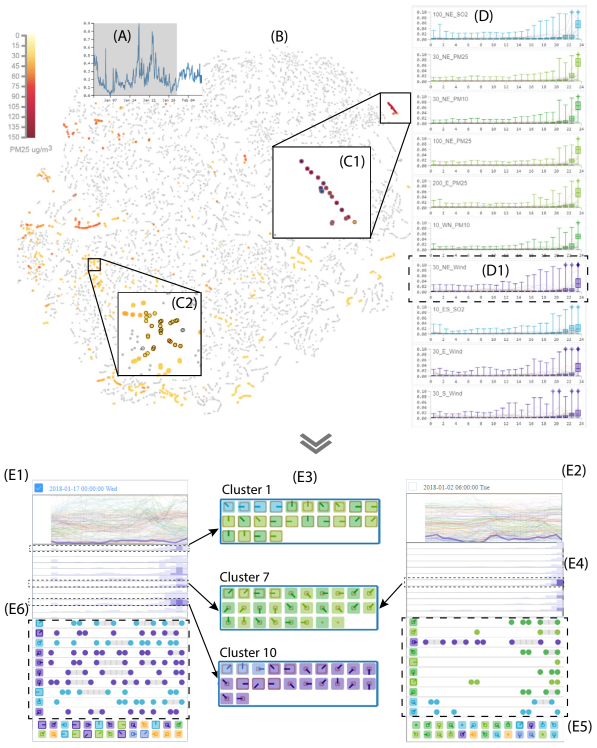

Figure 11: Case exploration for the $PM_{2.5}$ forecast in winter. A, B) Filter and highlight the individual cases in January 2018; C1, C2) Two groups of individual cases are selected by brushing; D) The top 10 importance features for the group selected in C1; E1, E2) Representative cases of C1 and C2.

long-term information than vanilla RNNs. Since the air pollutant and meteorology features (such as temperature) have a daily periodic pattern, the input features around 24 hours ahead are also considered relevant to the forecast. In addition, since the GRU model have better performance compared with vanilla RNNs from Table 2. We think the gate RNNs are applicable for the air pollutant forecast.

7.3 Case exploration

In Hong Kong, air quality always have seasonal patterns such as the "large winter-summer contrasts in $PM_{2.5}$ mass" due to the Asiatic monsoon [18]. We conduct this case study to understand the model behavior in different seasons.

7.3.1 Overview of the model behavior across seasons

In this case, we choose the GRU-Dense model and $PM_{2.5}$ as the target feature. We switch the time ranges to filter the cases in winter, spring, summer, and fall. To highlight the selected cases, the unfiltered cases are then colored in light grey as shown in Fig. 10. We found that the fall and winter cases are clustered at the left-bottom part (Fig. 10A and D). The cases in summer are located at the right-top part (Fig. 10C). Despite some outliers exist in the Projection View as shown in Fig. $10A_1$ and C_1, the model is able to capture the complex seasonal features in data based on above observation.

7.3.2 Explore the model behavior in winter

Furthermore, we notice that several cases indicate that the target location has a serious $PM_{2.5}$ pollutant in winter. In the Projection View, we first select the time ranges of January 2018 (Fig. 11A) to highlight all the cases within this time range (Fig. 11B). We notice there are many cases are colored in dark red, which indicates that these cases suffer a heavy $PM_{2.5}$ pollutant (Fig. $11C_1$). We select these cases to explore their feature importance distributions and rankings. The top ten important features are *Wind Speed*, $PM_{2.5}$ and PM_{10}. The selected cases also appear in the Individual Views; and we choose one representative case as shown in Fig. $11E_1$. In the Cluster Importance Chart, cluster 1, 7, and 10 show great influence in the forecast. These corresponding feature clusters are shown in Fig. $11E_3$, which presents the clusters of PM_x and *Wind Speed* respectively. The above exploration further indicates that the most important features in the forecasting are about $PM_{2.5}$, PM_{10}, and *Wind Speed*. Moreover, by hovering on the feature "*30_NE_Wind*" in feature importance view(Fig. $11D_1$), the corresponding feature is highlighted in the temporal trend view as Fig. $11E_1$ shown, which illustrates a weak wind speed.

The domain experts state: "During winter, strong radiative cooling over the continent creates a high-pressure anticyclone that drives cold, dry polar air from the continent into the surrounding oceanic areas, resulting in weak to moderate northeasterly winds or strong northerly winds" [18, 21]. The weak to moderate wind is able to bring the air pollutant from Pearl River Delta region in China, which is a highly populated region and has a lot of heavy industry [4]. Thus, the major factors that mostly influence Hong Kong's air quality in winter are the air pollutants and the wind. Specifically, the $PM_{2.5}$ and PM_{10} from the north and east mostly affects the prediction results.

We are also interested in comparing how the model generates prediction for the high-pollutant and low-pollutant cases in the winter. We select individual cases with low $PM_{2.5}$ from the Projection View as Fig. $11C_2$ shows, and the these cases appear in the Individual View. By observing the representative case (Fig. $11E_2$), we find only cluster 7 is very important at the last timestamp in the Cluster Importance Chart (Fig. $11E_4$), which indicates nearby $PM_{2.5}$ and PM_{10}. Both the Top Feature List (Fig. $11E_5$) and the Cluster Importance Chart show that the features of recent time steps are very important for the forecast, which is different from the heavy pollution cases (Fig. $11E_6$). We infer that for the low air pollutant cases, only recent important features are leveraged by the model.

8 Discussion and Conclusion

In this paper, we present MultiRNNExplorer, a visual analytic system for understanding RNN models for high-dimensional time-series forecasting. Specifically, we use air pollutant forecasting as the target application. To understand the the model mechanism from a global perspective, we propose a technique to estimate the hidden units' response to an individual feature by measuring how different feature selections affect the hidden units' output distribution. From a finer granularity, we further use the gradient-based method to measure the local feature importance for each sequence instance. Based on these techniques, we design a visual analytic system which enables the users to explore and reason about the model behavior from different perspectives. Our evaluation includes three case studies that demonstrate the effectiveness of the proposed system for comprehensive analysis of RNNs. Meanwhile, there are some issues need to be discussed:

Scalability. Several views may suffer scalability issues when the number of cases increase. In Projection View, thousands of individual cases need to be visualized and cause serious visual clutter due to the overlap of circles. In our case, more than 8000 points are visualized. If data size keeps increasing, we may also apply other advanced projection techniques [23, 31] for Projection View. In addition, the context + focus technique can also be applied for users to first obtain an overview of data then explore the regions of interests to reduce visual clutter and mental burden. The Individual View also has such a problem: it is easy for users to brush hundreds of individual cases from Projection View and generate tens of clusters. Due to the limited screen size, our current design allows 9 groups of individual cases to be shown at the same time and uses the scroll bar to enable the observation of more groups.

Generalization. Though we use air pollutant forecasting as example in this paper, the proposed method can be extended to other high-dimensional time-series forecasts with few changes. The current feature glyph design supports encoding three spatial attributes including direction, type, and distance and more design choices can be explored based on different domain requirements. There are also some future directions to improve MultiRNNExplorer. One approach is improving the individual comparison. In our current design, the individual comparison requires comparing data instances side by side. Supporting interactions to highlight the differences would be benefitial. We also consider applying our system on other high-dimensional forecasting applications such as fraud detection.

Acknowledgments

We thank all the reviewers for their constructive comments. This research is supported in part by the HSBC 150th Anniversary Charity Programme through the PRAISE-HK project (Grant No.HKBF17RG0) and National Natural Science Foundation of China (Grant No.61802388).

References

[1] Sulfur dioxide (so_2) pollution. https://www.epa.gov/so2-pollution/sulfur-dioxide-basics#effects. Accessed: 2019-09-07.

[2] R. Andrews, J. Diederich, and A. B. Tickle. Survey and critique of techniques for extracting rules from trained artificial neural networks. *Knowledge-based Systems*, 8(6):373–389, 1995.

[3] J. Brownlee. *Long Short-term Memory Networks with Python: Develop Sequence Prediction Models with Deep Learning*. Machine Learning Mastery, 2017.

[4] J. Cao, S. Lee, K. Ho, X. Zhang, S. Zou, K. Fung, J. C. Chow, and J. G. Watson. Characteristics of carbonaceous aerosol in pearl river delta region, china during 2001 winter period. *Atmospheric Environment*, 37(11):1451–1460, 2003.

[5] Y. Chen, Q. Cheng, Y. Cheng, H. Yang, and H. Yu. Applications of recurrent neural networks in environmental factor forecasting: A review. *Neural Computation*, 30(11):2855–2881, 2018.

[6] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[7] M. Craven and J. W. Shavlik. Extracting tree-structured representations of trained networks. In *Advances in Neural Information Processing Systems*, pp. 24–30, 1996.

[8] I. S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *Proceedings of the seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 269–274. ACM, 2001.

[9] J. H. Friedman. Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, pp. 1189–1232, 2001.

[10] F. A. Gers, D. Eck, and J. Schmidhuber. Applying lstm to time series predictable through time-window approaches. In *Neural Nets WIRN Vietri-01*, pp. 193–200. Springer, 2002.

[11] S. Hochreiter. Untersuchungen zu dynamischen neuronalen netzen. *Diploma, Technische Universität München*, 91(1), 1991.

[12] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[13] A. Karpathy, J. Johnson, and L. Fei-Fei. Visualizing and understanding recurrent networks. *arXiv preprint arXiv:1506.02078*, 2015.

[14] B. C. Kwon, M.-J. Choi, J. T. Kim, E. Choi, Y. B. Kim, S. Kwon, J. Sun, and J. Choo. Retainvis: Visual analytics with interpretable and interactive recurrent neural networks on electronic medical records. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):299–309, 2019.

[15] J. Li, X. Chen, E. Hovy, and D. Jurafsky. Visualizing and understanding neural models in nlp. *arXiv preprint arXiv:1506.01066*, 2015.

[16] Z. C. Lipton. The doctor just won't accept that! *arXiv preprint arXiv:1711.08037*, 2017.

[17] M. Liu, J. Shi, Z. Li, C. Li, J. Zhu, and S. Liu. Towards better analysis of deep convolutional neural networks. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):91–100, 2017.

[18] P. K. Louie, J. G. Watson, J. C. Chow, A. Chen, D. W. Sin, and A. K. Lau. Seasonal characteristics and regional transport of pm2. 5 in hong kong. *Atmospheric Environment*, 39(9):1695–1710, 2005.

[19] S. M. Lundberg and S.-I. Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, pp. 4765–4774, 2017.

[20] Y. Ming, S. Cao, R. Zhang, Z. Li, Y. Chen, Y. Song, and H. Qu. Understanding hidden memories of recurrent neural networks. In *2017 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pp. 13–24. IEEE, 2017.

[21] T. Murakami. Winter monsoonal surges over east and southeast asia1. *Journal of the Meteorological Society of Japan. Ser. II*, 57(2):133–158, 1979.

[22] M. Oprea, M. Popescu, and S. F. Mihalache. A neural network based model for pm 2.5 air pollutant forecasting. In *2016 20th International Conference on System Theory, Control and Computing (ICSTCC)*, pp. 776–781. IEEE, 2016.

[23] N. Pezzotti, T. Höllt, B. Lelieveldt, E. Eisemann, and A. Vilanova. Hierarchical stochastic neighbor embedding. In *Computer Graphics Forum*, vol. 35, pp. 21–30. Wiley Online Library, 2016.

[24] N. Pezzotti, T. Höllt, J. Van Gemert, B. P. Lelieveldt, E. Eisemann, and A. Vilanova. Deepeyes: Progressive visual analytics for designing deep neural networks. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):98–108, 2018.

[25] M. T. Ribeiro, S. Singh, and C. Guestrin. Why should i trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1135–1144. ACM, 2016.

[26] P. J. Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65, 1987.

[27] M. Schuster and K. K. Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.

[28] H. Strobelt, S. Gehrmann, M. Behrisch, A. Perer, H. Pfister, and A. M. Rush. Seq2seq-vis: A visual debugging tool for sequence-to-sequence models. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):353–363, 2019.

[29] H. Strobelt, S. Gehrmann, H. Pfister, and A. M. Rush. Lstmvis: A tool for visual analysis of hidden state dynamics in recurrent neural networks. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):667–676, 2018.

[30] Y. Sun, X. Wang, and X. Tang. Deeply learned face representations are sparse, selective, and robust. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 2892–2900, 2015.

[31] V. van Unen, T. Höllt, N. Pezzotti, N. Li, M. J. Reinders, E. Eisemann, F. Koning, A. Vilanova, and B. P. Lelieveldt. Visual analysis of mass cytometry data by hierarchical stochastic neighbour embedding reveals rare cell types. *Nature Communications*, 8(1):1740, 2017.

[32] S. Xingjian, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-c. Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. In *Advances in Neural Information Processing Systems*, pp. 802–810, 2015.

[33] Y. Zheng, X. Yi, M. Li, R. Li, Z. Shan, E. Chang, and T. Li. Forecasting fine-grained air quality based on big data. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 2267–2276. ACM, 2015.