ViEWS Competition

August 2020

1 Proposed Approach

Fatality prediction for ViEWS can be analyzed from different perspectives. On one hand, it can be considered as a typical time-series forecasting problem, and it aims to predict future length-Q number of fatalities based on previous P observations, and this case the observation is on a monthly basis:

$$[x_{(t-P+1)}, \cdots, x_{(t)}] \xrightarrow{f(\cdot)} [y_{(t+1)}, \cdots, y_{(t+Q)}] \tag{1}$$

On the other hand, we focus the research scope on the PRIO-GRID level in this study. More specifically, our target is to predict the future number of fatalities in each cell on a monthly basis. Given the geographical connections between these PRIO-GRID cells, the map can be defined as an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{A})$, where $\mathcal{V}$ is a set of cells with $|\mathcal{V}| = N$; $\mathcal{E}$ is a set of edges representing the connectivity between cells, and; $\mathbf{A} \in \mathbb{R}^{N \times N}$ denotes the adjacency matrix of graph $\mathcal{G}$. Notably, the overall raw inputs to our model are denoted by $X \in \mathbb{R}^{P \times N \times d}$.

Therefore, we conclude that the solution for the fatalities prediction of ViEWS needs to consider both spatial and temporal dependencies within the exiting data. Therefore, we propose a Spatio-Temporal Graph Convolutional Network (STGCN) to address the prediction challenges in this competition.

Parameter	Values	Meaning
P	54	Number of past steps used for generating predictions
Q	6	Number of future steps for forecasting
d	Variable	Input feature dimension for each instance
d'	Variable	Output feature dimension for each instance
S_{tcn}	3	Number of stacks for STGCN modules in STGCN-TCN
S_{lstm}	3	Number of stacks for GCN modules in STGCN-LSTM
k_{tp}	3	Kernel size of temporal convolution

Table 1: Notations and values of parameters.

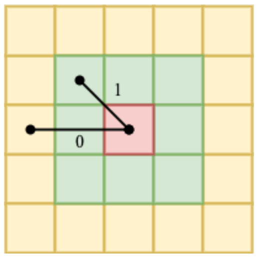


Figure 1: Demonstrations of graph connections w.r.t PRIO-GRID maps. In the adjacency matrix, the edge weights between red cell and green cells are $\mathbf{1}$s, which indicate **direct connection**. However, the edge weights between red cell and yellow cells are $\mathbf{0}$s, which indicate **no direct connection**.

1.1 Spatial Dependency Modeling

The Graph Convolutional Network (GCN) is an effective model when mining graph structured data, e.g., social networks, online advertising, and etc [Kipf and Welling2017]. In the ViEWS prediction competition, we define the map of Africa as an undirected graph. Here, nodes of graph are the PRIO-GRID cells. For a certain cell, it is connected to its 8 surrounding cells, and it is not directly connected to all other cells. This relationship can be referred to Fig. 1. The GCN module is then implemented to handle the spatial dependencies regarding PRIO-GRID cells in Africa. In other words, this module tries to pass the information to each cell from its surrounding cells by taking a weighted average (convolutions) regarding their features.

For this framework, we follow the derivations from [Kipf and Welling2017]. In particular, for spatial dependency modeling, the concentration is at each time step t of P. There are two inputs in this module. The first one is a variable $x_{sp} \in \mathbb{R}^{N \times d_0}$, where d_0 is the dimensions of feature for each PRIO-GRID cell history data. The second put is the adjacency matrix $\mathbf{A}$ of the graph.

Consequently, for each time step t of P, the layer-wise propagation rule of GCN can be shown as follows:

$$H^{(l+1)} = \sigma\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}\right) \tag{2}$$

where $H^{(l)} \in \mathbb{R}^{n \times d}$ is the matrix of activations in the l^{th} layer with $H^{(0)} = X$. I_N is the identity matrix with dimension of n, thus $\tilde{A} = A + I_N$ is the adjacency matrix of the undirected PRIO-GRID level graph $\mathcal{G}$ with self-connections. $\tilde{D}$ denotes the degree matrix which can be represented as $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$. $W^{(l)} \in \mathbb{R}^{d \times d'}$ is a layer-specific trainable weight matrix. d and d' are the number of feature dimensions for input and out respectively at the l layer. Finally, $\sigma(\cdot)$ denotes an activation function such as $\mathrm{ReLU}(\cdot) = \max(0, \cdot)$. The ReLU is an effective activation function in deep learning which introduce non-linearity to the neural network shown in Fig. 2.

Notice that Eq. 2 indeed implies the two-step propagation rule of GCN network: 1. Aggregate information from each cell's neighbors; 2. Update the neural network. $\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}$ indeed represents the graph convolution, which takes the graph connections into consideration, and passes neighbor information to each node/cell, and aggregate them based on the connection weights. Then the aggregated matrix is put into a standard neural network, which essentially multiply it by a trainable weight matrix $W^{(l)}$, and feed it to the activation function $\sigma(\cdot)$.

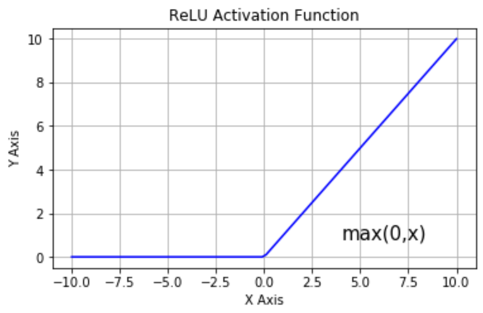


Figure 2: Illustration of ReLU function. Given function input $x \in \mathbb{R}$, the output y of this function is $y = 0$ if $x \leq 0$, else $y = x$.

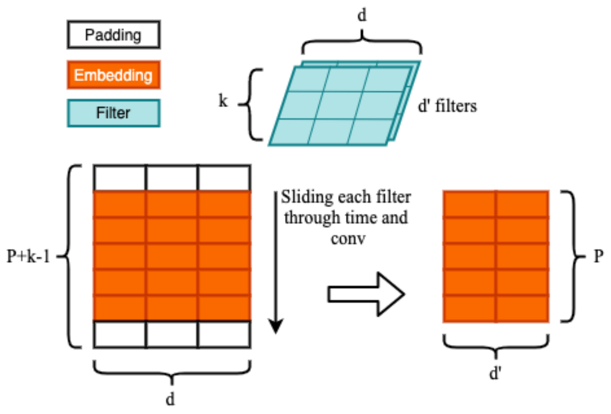


Figure 3: Demonstration of 1D Convolution. By applying tricks such as padding, we keep the time steps of outputs as the same as inputs. The feature dimension of outputs is determined by the number of filters.

1.2 Temporal Dependency Modeling

Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN) are two major types of deep learning approaches when it comes to sequence modeling [Lim and Zohren2020]. In this research, we have developed models based on both of these blocks, and preliminaries about them are as follows. It is worth noticing that for temporal dependency modeling, the focus is on each of the node. Therefore inputs for each node are $x_{tp} \in \mathbb{R}^{P \times d}$.

1.2.1 Temporal Convolution Networks (TCN)

The basic idea of CNN-based models is that temporal information in time series is aggregated by learnable convolution filters along time [Yu, Yin, and Zhu2018]. The following demonstration is regarding each node. We share the filters for all nodes, and concatenate the results for each node after all the calculation.

Fig. 3 shows the basic idea of 1D convolution with a kernel with size k. Basically this operation allows the model to explore the temporal dependency of each node for k months each time. The convolutional kernel K_t is designed to map the input x_{tp} to a embedding. In order to keep time steps w.r.t embeddings consistent along time as P, we pad zero to both beginning and end of the

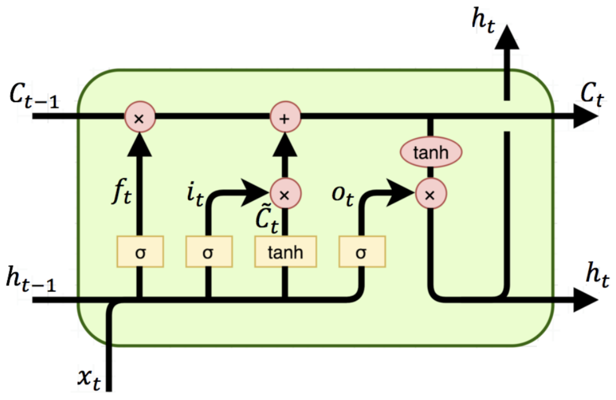


Figure 4: Architecture of one LSTM cell. Here σ and tanh are activation functions.

window. Thus the dimension of embedding after this convolution step is $P \times d'$. Then, an activation function is applied to this embedding and this whole process can be abstracted as:

$$H^{(l+1)} = \sigma\left(\mathrm{conv}(H^{(l)}, W^{(l)})\right) \tag{3}$$

where conv denotes the convolution operation, $W^{(l)}$ is the trainable weight representing convolutional kernel here with dimension of $k \times d \times d'$.

1.2.2 Long Short Term Memory (LSTM)

Recurrent Neural Network (RNN) model is another type of popular framework when it comes to time-series forecasting [Che et al.2018]. The notable success of RNNs has show their capabilities on many end-to-end learning tasks such as text sentiment classification. The mechanism of RNN allows it to exhibit temporal dynamic behavior. Taking advantage of their internal hidden state, RNN is capable to process variable length sequences of inputs [Li et al.2018].

In this research, we implement a variation of RNN models, which is called Long Short Term Memory Network. In this research, we follow the architecture and notations described by [Tai, Socher, and Mannir
Overall, the LSTM model can be described as:

$$\begin{aligned}
f_t &= \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) && (4)\\
i_t &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) && (5)\\
o_t &= \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) && (6)\\
\tilde{C}_t &= \tanh(W_c \cdot [h_{t-1}, x_t] + b_c) && (7)\\
C_t &= f_t \odot C_{t-1} + i_t \odot \tilde{C}_t && (8)\\
h_t &= o_t \odot \tanh(C_t) && (9)
\end{aligned}$$

where x_t is the input of current time step, which is of dimension $n \times d$. σ denotes the logistic sigmoid function, and $\odot$ denotes elementwise multiplication. h represents the short-term memory (embedding), and C represents the long term memory, and h, C has random initialization at time step 0.

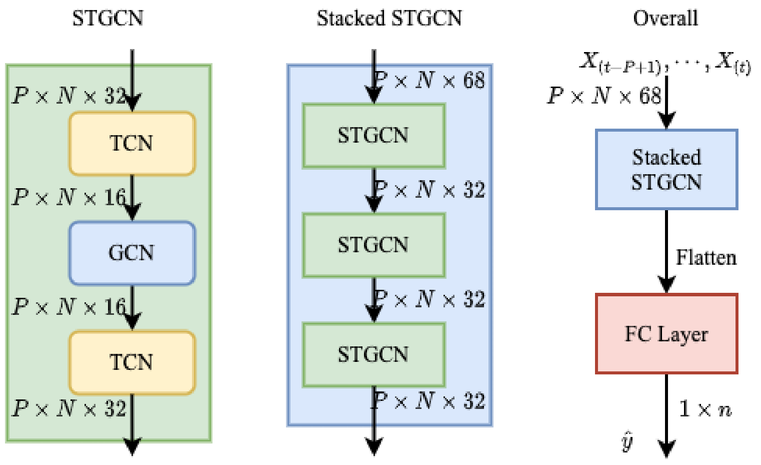


Figure 5: Architecture of STGCN-TCN. This framework consists three STGCN blocks with a fully-connected (FC) layer by the end to generate outputs.

1.3 Model Details

We have applied two types of networks in this study for modeling both spatial and temporal dependencies w.r.t PRIO-GRID level information.

As we previously discussed, essentially we use similar GCN modules for spatial modeling. And the major differences are about the temporal modeling mechanisms. TCN and LSTM are used in this study to model the temporal dependencies respectively. The notations and parameters of this research are show in Table 2.

1.3.1 STGCN-TCN

Details regarding STGCN-TCN model are presented. Overall, this model is composed with a stacked STGCN module, and a Fully Connected (FC) layer at the end to generate predictions. Within the stacked STGCN module, we stack three identical STGCN modules altogether to jointly process graph-structured time series. It is possible to stack more STGCN modules depending on the requirement of the task. Also noticing that the input dimension for the first STGCN block is different from latter ones as the input here is the raw data with $d = 68$.

As Fig 5 shows, in each of the STGCN block, we stack the sub-modules in TCN-GCN-TCN fashion. In this architecture, the spatial layer is in the middle of two temporal layers and linked those two temporal layers together. Considering the dimensions of both temporal and spatial layers, we purposely design it similar to a “sandwich” structure. This trick is also know as “bottleneck strategy”, which is very useful when designing a neural network. Having such design encourages the network to compress feature representations to best fit in the available space, in order to get the best loss during training.

Based on existing parameters, each STGCN module is capable to aggregate 1-hop of spatial dependencies, and three months of temporal dependencies. Since the stacked STGCN model has three STGCN modules in it sequentially, overall our proposed framework can aggregate upto 3-hops of spatial dependencies, and upto 5-months of temporal dependencies.

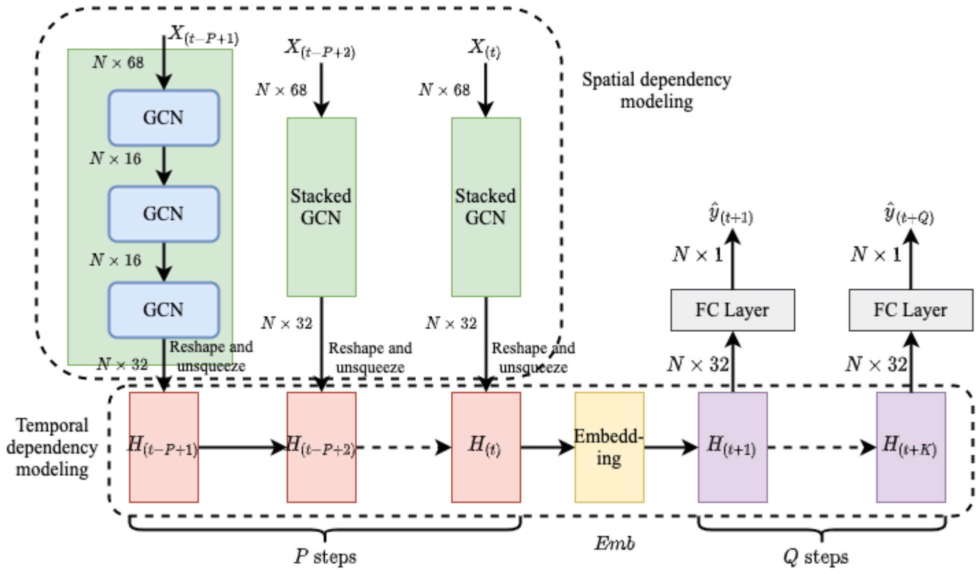


Figure 6: Architecture of STGCN-RNN. This framework uses GCN for spatial modeling, and applies LSTM as backbone for temporal modeling.

Parameter	Values	Meaning
P	54	Number of past steps used for generating predictions
Q	6	Number of future steps for forecasting
d	Variable	Input feature dimension for each instance
d'	Variable	Output feature dimension for each instance
S_{tcn}	3	Number of stacks for STGCN modules in STGCN-TCN
S_{lstm}	3	Number of stacks for GCN modules in STGCN-LSTM
k_{tp}	3	Kernel size of temporal convolution

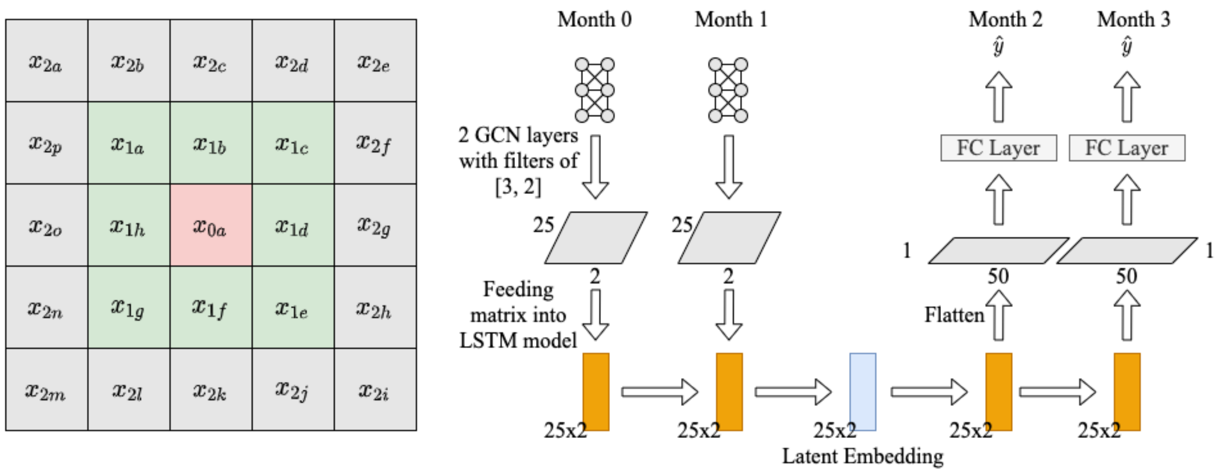
Table 2: Notations and values of parameters.

1.3.2 STGCN-LSTM

Fig. 6 demonstrates the architecture of our designed STGCN-LSTM framework. Unlike the STGCN-TCN framework, which mix the spatial layers and temporal layers together in each STGCN block, STGCN-RNN is a two step model which can be separated clearly by: 1. spatial dependency modeling, and; 2. temporal dependency modeling. More specifically, first we use stacked GCN blocks to extract spatial information out of our data, and embed them into the H_t at each time step. Each stacked GCN module contains three GCN blocks in order to obtain multi-hop information for the target cell. Similar strategy is applied here to design the stacked GCN module as a "sandwich" structure to get better loss during training.

Then, we feed H_t into the LSTM model in a sequential manner and generate a unique hidden embedding Emb after P steps. The LSTM model uses Emb and make predictions sequentially for Q steps.

Based on existing parameters, each stacked GCN module is capable to aggregate 3-hop spatial dependencies. The long term memory mechanism helps the LSTM module to maintain all previous



x_{2a} x_{2b} x_{2c} x_{2d} x_{2e}
x_{2p} x_{1a} x_{1b} x_{1c} x_{2f}
x_{2o} x_{1h} x_{0a} x_{1d} x_{2g}
x_{2n} x_{1g} x_{1f} x_{1e} x_{2h}
x_{2m} x_{2l} x_{2k} x_{2j} x_{2i}
Month 0
Month 1
Month 2
Month 3
$\hat{y}$
$\hat{y}$
2 GCN layers with filters of [3, 2]
FC Layer
FC Layer
25
25
2
2
1
1
50
50
Feeding matrix into LSTM model
Flatten
25x2
25x2
25x2
25x2
25x2
Latent Embedding

Figure 7: Example of STGCN-LSTM model

information, which can be tracked back to $t - P + 1$ at time step t, which makes the extracted temporal dependencies upto P steps.

1.3.3 Training

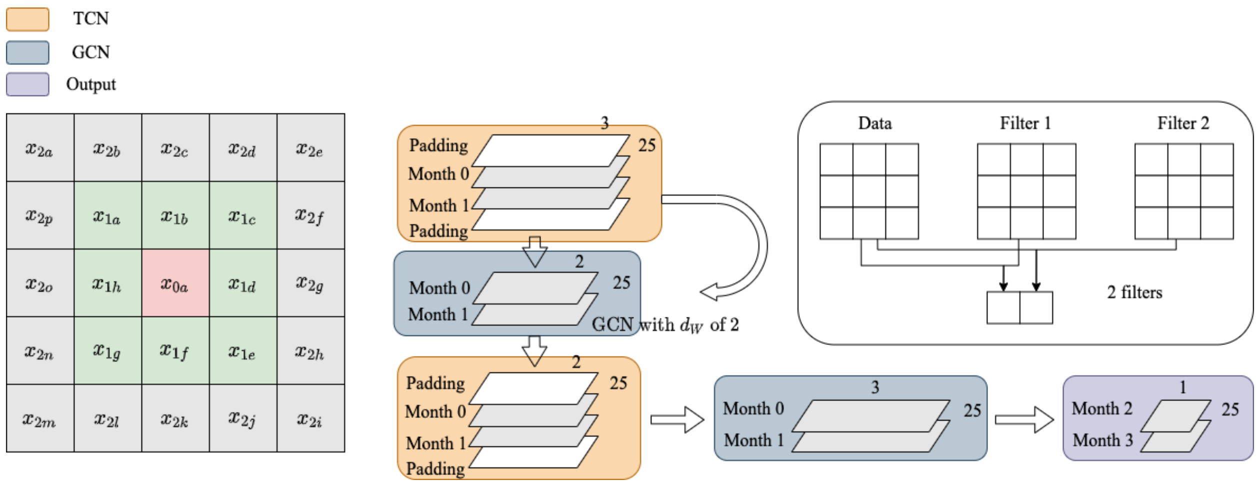
For both STGCN-TCN model and STGCN-LSTM model, we use Mean Squared Error (MSE) as the training objective function. Essential this objective function represents the differences between predictions for each grid cell of each month to the ground truth. Mathematically, this loss function is defined as:

$$L = \frac{1}{QN} \sum_{i=1}^{Q} \sum_{j=1}^{N} \left(\hat{y}_j^{(t+i)} - y_j^{(t+i)} \right)^2 \tag{10}$$

1.4 Demo of STGCN

Here we further demonstrate our proposed model by a simplified example. Suppose that there is a area with 5×5 PRIO-GRID cells, for each of the grid cell we have 3 dimensional features, and we have a total of 4 months of data available. Also, we want the model to have the ability of predicting two month ahead. Therefore for the training, we have 2 months of features (X_0 and X_1) and 2 months of labels (y_2 and y_3) available.

First, we demonstrate the model of STGCN-LSTM in Fig 7. Since there is at most 2-hop connection w.r.t the center cell x_{0a}, here we apply two layers of GCN module. Since each GCN layer can aggregate 1-hop information, therefore after the first GCN layer, those surrounding nodes information from x_{1u} where $u \in \{a - h\}$ are encoded into x_{0a}. Noticing that nodes such as x_{1a} also aggregate its surrounding information from nodes such as x_{2a} in this first GCN layer. Then, we feed embedding generated by the 1st layer to the 2nd GCN layer. Since x_{1u} already got x_{2v}, where $v \in \{a - p\}$ encoded from last layer, then here x_{0a} can theoretically aggregates information from 2-hops away.



TCN
GCN
Output
x_{2a} x_{2b} x_{2c} x_{2d} x_{2e}
x_{2p} x_{1a} x_{1b} x_{1c} x_{2f}
x_{2o} x_{1h} x_{0a} x_{1d} x_{2g}
x_{2n} x_{1g} x_{1f} x_{1e} x_{2h}
x_{2m} x_{2l} x_{2k} x_{2j} x_{2i}
Padding
Month 0
Month 1
Padding
3
25
2
GCN with d_W of 2
Data
Filter 1
Filter 2
2 filters
Month 2
Month 3
1

Figure 8: Example of STGCN-TCN model

	Starts	Ends
Training	Jan, 2000	Dec, 2014
Validation	Jan, 2015	Dec, 2015
Testing	Jan, 2016	Dec, 2017

Table 3: Demonstrations of the training, validation, and testing splits for this paper

1.5 Discussions

2 Experiment

2.1 Dataset

Our model is validated based on two datasets at PRIO-grid level: ViEWS [Hegre et al.2019], and UTD Event Data [Kim et al.2019]. A total of 18 years data are used here in our research, between Jan, 2000 and Dec, 2017. To better align with the experiment settings of the benchmark codes provided by ViEWS, we further divide the data applying similar strategies, and details of training, validation and testing split is shown in Tab. 3

Features from ViEWS are what we are mainly use in this research to align with the competition settings. More specifically, 48 features are retrieved, which includes both conflicts or fatalities related indicators, and geographic related indicators. For the conflicts related features, an example could be the "pgd_count_sb", which is the count of state based conflicts at the PRIO-grid level. Regarding the geographic related features, one example could be the "pgd_forest_gc", which indicates the percentage area of the cell covered by the forest area. It is worth noticing that all ViEWS data used here is at monthly frequency. The full list of features used in this research from ViEWS dataset can be seen in Tab. 4.

In addition, we further include ICEWS data retrieved from UTD Event Data repository [Kim et al.2019]. The motivation is that even though applying conflict features from ViEWS could provide promisingly accurate predictions already, we can further push the model performance by introducing additional data sources with detailed categories. For example, in UTD Event Data, every ICEWS

acled_count_pr	acled_fat_pr	ged_best_ns	ged_best_os	ged_best_sb
ged_count_os	ged_count_sb	ged_count_ns	pgd_agri_ih	pgd_aquaveg_gc
pgd_barren_ih	pgd_bdist3	pgd_capdist	pgd_cmr_mean	pgd_diamprim
pgd_drug_y	pgd_excluded	pgd_forest_gc	pgd_forest_ih	pgd_gcp_mer
pgd_goldplacer	pgd_goldsurface	pgd_goldvein	pgd_grass_ih	pgd_gwarea
pgd_herb_gc	pgd_imr_mean	pgd_landarea	pgd_maincrop	pgd_mountains_mean
pgd_pasture_ih	pgd_petroleum	pgd_pop_gpw_sum	pgd_savanna_ih	pgd_shrub_gc
pgd_temp	pgd_ttime_mean	pgd_urban_gc	pgd_urban_ih	pgd_water_gc
pgd_agri_gc	pgd_barren_gc	pgd_diamsec	pgd_gem	pgd_harvarea
pgd_nlights_calib_mean	pgd_shrub_ih	pgd_water_ih		

Table 4: Full list of features used from ViEWS dataset

Month ID	PRIO-grid ID	RC01	RC02	$\cdots$	RC20

Table 5: Demonstration of processed data retrieved from UTD Event Data repository

event corresponds to a CAMEO root code, which varies between 01 and 20. Essentially these root codes represent categories of events besides "conflicts", such as "comment", "consult", "threaten", and etc [Gerner et al.2002]. We take the one-hot encoding regarding the CAMEO root code for each event, then we aggregate them and concatenate them to the existing ViEWS data. More specifically, we first assign each event to PRIO-grid cells based on its latitude and longitude coordinates. Then we group the data by its month id, and take summations of each one-hot encoded CAMEO root code. Therefore each column now represents the total count of events regarding each CAMEO root code for every PRIO-grid cell in a monthly basis. A colum representation of this data table can be seen in Tab. 5

2.2 Results

Our initial results are shown in Tab. 6. In addition, codes of our implementation for this research are publicly available for easier reproducing the results[1]. The Random Forest (RF) Regressor is the benchmark model that provided by the ViEWS competition organizer[2]. We compare our proposed methods (STGCN-LSTM and STGCN-TCN) with this benchmark model.

[1]`https://github.com/evanli05/ViEWS_Competition`
[2]`https://github.com/UppsalaConflictDataProgram/OpenViEWS2`

Metrics		RF Regressor				STGCN-LSTM (b48-w48)				STGCN-TCN (b32-w60)			
Harware		CPU: i9-7920X				GPU: Quadro RTX 8000							
Features		v		v+u		v		v+u		v		v+u	
Metrics		MSE	CRPS	MSE	CRPS	MSE	CRPS	MSE	CRPS	MSE	CRPS	MSE	CRPS
Time steps	1	3.03×10^{-2}		2.69×10^{-2}		2.59×10^{-2}	4.47×10^{-2}	2.37×10^{-2}	1.25×10^{-2}	2.32×10^{-2}	1.61×10^{-2}	2.42×10^{-2}	3.54×10^{-2}
	2	3.09×10^{-2}		3.04×10^{-2}		2.31×10^{-2}	2.25×10^{-2}	2.42×10^{-2}	1.66×10^{-2}	2.32×10^{-2}	3.03×10^{-2}	2.40×10^{-2}	3.40×10^{-2}
	3	3.12×10^{-2}		3.06×10^{-2}		2.59×10^{-2}	4.47×10^{-2}	2.53×10^{-2}	1.32×10^{-2}	2.21×10^{-2}	7.07×10^{-2}	2.24×10^{-2}	7.33×10^{-2}
	4	3.26×10^{-2}		3.21×10^{-2}		3.00×10^{-2}	8.36×10^{-2}	2.55×10^{-2}	3.00×10^{-2}	2.36×10^{-2}	2.50×10^{-2}	2.13×10^{-2}	6.54×10^{-2}
	5	3.34×10^{-2}		3.31×10^{-2}		3.07×10^{-2}	5.60×10^{-2}	2.63×10^{-2}	3.81×10^{-2}	2.14×10^{-2}	1.04×10^{-2}	2.22×10^{-2}	1.60×10^{-2}
	6	3.35×10^{-2}		3.32×10^{-2}		3.07×10^{-2}	7.66×10^{-2}	2.47×10^{-2}	2.26×10^{-2}	2.21×10^{-2}	2.01×10^{-2}	2.21×10^{-2}	2.63×10^{-2}

Table 6: Comparisons of results between our proposed methods and baseline methods. "**v**" represents "ViEWS" dataset features and "**u**" represents "UTD Event Data" features.

References

[Che et al.2018] Che, Z.; Purushotham, S.; Cho, K.; Sontag, D.; and Liu, Y. 2018. Recurrent neural networks for multivariate time series with missing values. *Scientific reports* 8(1):1–12.

[Gerner et al.2002] Gerner, D. J.; Schrodt, P. A.; Yilmaz, O.; and Abu-Jabr, R. 2002. Conflict and mediation event observations (cameo): A new event data framework for the analysis of foreign policy interactions. *International Studies Association, New Orleans*.

[Hegre et al.2019] Hegre, H.; Allansson, M.; Basedau, M.; Colaresi, M.; Croicu, M.; Fjelde, H.; Hoyles, F.; Hultman, L.; Högbladh, S.; Jansen, R.; et al. 2019. Views: a political violence early-warning system. *Journal of peace research* 56(2):155–174.

[Kim et al.2019] Kim, H.; D'Orazio, V.; Brandt, P. T.; Looper, J.; Salam, S.; Khan, L.; and Shoemate, M. 2019. Utdeventdata: An r package to access political event data. *Journal of Open Source Software* 4(36):1322.

[Kipf and Welling2017] Kipf, T. N., and Welling, M. 2017. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

[Li et al.2018] Li, Y.; Yu, R.; Shahabi, C.; and Liu, Y. 2018. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

[Lim and Zohren2020] Lim, B., and Zohren, S. 2020. Time series forecasting with deep learning: A survey. *arXiv preprint arXiv:2004.13408*.

[Tai, Socher, and Manning2015] Tai, K. S.; Socher, R.; and Manning, C. D. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, 1556–1566. The Association for Computer Linguistics.

[Yu, Yin, and Zhu2018] Yu, B.; Yin, H.; and Zhu, Z. 2018. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. In Lang, J., ed., *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, 3634–3640. ijcai.org.