Multivariate Time Series-based Solar Flare Prediction by Functional Network Embedding and Sequence Modeling

Shah Muhammad Hamdi[1,*], Abu Fuad Ahmad[2] and Soukaina Filali Boubrahimi[1]

[1]Utah State University, Logan, UT, 84322, USA
[2]New Mexico State University, Las Cruces, NM, 88003, USA

Abstract

Major flaring events on the Sun can have hazardous impacts on both space and ground-based infrastructure. An effective approach of predicting that a solar active region (AR) is likely to flare after a period of time is to leverage multivariate time series (MVTS) of the AR magnetic field parameters. Existing MVTS-based flare prediction models are based on training traditional classifiers with preset statistical features of univariate time series instances, or training deep sequence models based on Recurrent Neural Network (RNN) or Long Short Term Memory (LSTM) Network. While the earlier approach is affected by hand-engineered features, the latter approach uses only the temporal dimension of the MVTS instances. The variables of MVTS do not depend only on their historical values but also on other variables. In this work, we used the dynamic functional network representation of the MVTS instances to leverage higher-order relationships of the variables through Graph Convolution Network (GCN) embedding. In addition to finding spatial (inter-variable) patterns through functional network embedding, our model uses local and global temporal patterns through LSTM networks. Our experiments on a real-life solar flare dataset exhibit better prediction performance than other baseline methods.

Keywords

Solar flare prediction, Multivariate time series, GCN, LSTM

1. Introduction

Solar flares are characterized by sudden bursts of magnetic flux in the solar corona and heliosphere. Extreme Ultra-Violet (EUV), X-ray, and gamma-ray emissions caused by major flaring events can have disastrous effects on our technology-dependent society. The risks of life and infrastructure in both space and ground include radiation exposure-based health risks of the astronauts, disruption in GPS and radio communication, and damages in electronic devices. The economic damage of such extreme solar events can rise up to trillions of dollars [1]. In 2015, the White House released the National Space Weather Strategy and Space Weather Action Plan [2] as a roadmap for research aimed at predicting and mitigating the effects of solar eruptive activities.

In recent years, multiple research efforts of the heliophysics community aim to predict solar flares from the current and historic magnetic field states of the solar active regions. Due to the absence of direct theoretical relationship between magnetic field influx and flare occurrence in active regions (AR), solar physics researchers rely on data science-based approaches for predicting solar flares. The data is collected by the Helioseismic Magnetic Imager (HMI) housed in the Solar Dynamics Observatory. Near-continuous-time images captured by the instruments of HMI contain spatiotemporal magnetic field data of the active regions. The prediction of solar flares, which will identify active regions that will potentially flare after a period of time, requires time series modeling of the magnetic field data. For that, spatiotemporal magnetic field data of active regions are mapped into multiple MVTS instances [3]. The variables of the MVTS instances represent solar magnetic field parameters (e.g., flux, current, helicity, Lorentz force). The time series corresponding to the magnetic field parameters are extracted based on two time windows: *observation window* (the time window of data collection), and *prediction window* (the time window after the data collection and before the flare occurrence). Each MVTS instance is labeled as one of six classes - Q, A, B, C, M, and X, where Q represents flare quiet active regions, and other labels represent flaring events with increasing intensity. Among these classes, X and M-class flares are considered as most intense flaring events.

In comparison to the earlier single timestamp-based magnetic field vector classification models, recent MVTS-based models are more effective for predicting flaring activities [3]. MVTS classification models targeting flare prediction are divided in two categories: (1) statistical feature-based method [4], and (2) end-to-end deep learning-based method [5]. The models of the first category work in two steps. Firstly, low-dimensional repre-

AMLTS'22: Workshop on Applied Machine Learning Methods for Time Series Forecasting, co-located with the 31st ACM International Conference on Information and Knowledge Management (CIKM), October 17-21, 2022, Atlanta, USA
*Corresponding author.
s.hamdi@usu.edu (S. M. Hamdi); fuad@nmsu.edu (A. F. Ahmad); soukaina.boubrahimi@usu.edu (S. F. Boubrahimi)
0000-0002-9303-7835 (S. M. Hamdi); 0000-0001-5693-6383 (S. F. Boubrahimi)

sentations of MVTS instances are calculated from concatenation/aggregation of summarization statistics (e.g., mean, standard deviation, skewness, kurtosis, etc) of the univariate time series components. Lastly, traditional classifiers (e.g., kNN, SVM, etc) are trained with labeled MVTS representations. The two-step process of MVTS classification relies heavily on hand-engineered statistical features and the choice of downstream classifiers, which eventually complicates the application of these models in datasets with varying properties. In the second category, RNN/LSTM-based deep sequence models are trained by sequentially feeding vectors representing magnetic field parameters into sequence model cells, and optimizing the cell weights through gradient descent-based backpropagation. While the deep learning models ensure end-to-end learning bypassing the dependency on hand-engineered features, they can utilize only the time dimension of the MVTS instances, and this limited usage of underlying patterns results in poor classification performance.

In this work, we propose a deep learning-based MVTS classification approach for solar flare prediction leveraging the the fact that MVTS data is rich not only in temporal dimension, but also in spatial dimension which encodes inter-variable relationships [6]. For learning higher-order relationships of the MVTS variables, we used functional networks, where nodes represent variables, and edges represent positive correlation of the time series of corresponding variables. The MVTS instance is divided into equal-length temporal windows, and an edge-weighted functional network is constructed for each window. We trained Graph Convolution Network (GCN) to learn representation of each functional network. In addition, we used two LSTM networks for learning representations based on temporal dimension within and between the windows. Our model significantly outperforms existing MVTS-based flare prediction models on a dataset containing MVTS instances of solar events of different flare classes.

The contributions made by this paper are listed below.

1. Leveraging higher-order inter-variable relationships of the MVTS instances by GCN-based dynamic functional network embedding.
2. Utilizing local and global patterns of the temporal dimension of the MVTS instances through LSTM-based within-window and between-window sequence learning.
3. Experimentally demonstrating the better performance of our model in comparison with the state-of-the-art baselines on a benchmark solar flare prediction dataset.

2. Related Work

While the current approaches of flare prediction are mostly based on data science, the earliest flare prediction system was an expert system named *THEO* that required human inputs [7]. The Space Environment Center (SEC) of the National Oceanic and Atmospheric Administration (NOAA) adopted the system THEO in 1987. To distinguish flare classes, THEO was provided input data of sunspots and magnetic field properties.

Due to the abundance of magnetic field data collected by NASA's recent missions, research efforts of flare prediction of the last two decades are based on data science rather than on purely theoretical modeling. Data science-based approaches stemmed from both linear and nonlinear statistics. Based on the type of dataset used, these approaches are subdivided into two classes: line-of-sight magnetogram-based models and vector magnetogram-based models. Solar active regions are represented by the parameters of either photospheric magnetic field data that contain only the line-of-sight component of the magnetic field or by the full-disk photospheric vector magnetic field. Followed by NASA's launch of SDO in 2010, the HMI instrument has been mapping the full-disk vector magnetic field every 12 minutes [8]. Most of the recent models use the near-continuous stream of vector magnetogram data found from SDO, while the earlier models (dated before 2010) mostly used line-of-sight magnetic data.

The objective of the linear statistical models was to find the active region magnetic field features that are highly correlated with the flare occurrences. Cui et al. [9] and Jing et al. [10] used line-of-sight magnetogram data to find correlation-based statistical relationships between magnetic field parameters and flare occurrences. Even before the launch of SDO, Leka and Barnes [11] collected and curated vector magnetogram data from Mees Solar Observatory on the summit of Mount Haleakala, and used linear discriminant analysis (LDA) for classifying flaring events.

Nonlinear statistical models are mostly machine learning classifiers based on tree induction, kernel method, neural network, and so on. On the line-of-sight magnetogram-based active region datasets, Song et al. [12] used logistic regression, Yu et al. [13] used C4.5 decision tree, Ahmed et al. [14] used the fully connected neural network, and Al-Ghraibah et al. [15] used relevance vector machine as classification models. Bobra et al. [16] used Support Vector Machine (SVM) on SDO-based vector magnetogram data for classifying flaring and non-flaring active regions. Nishizuka et al. [17] used both line-of-sight and vector magnetograms and compared the performance of three classifiers - kNN, SVM, and Extremely Randomized Tree (ERT). Other examples of solar flare prediction on non-sequential data include various

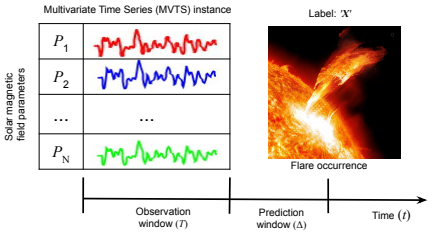


Figure 1: Multivariate time series instance with predefined observation and prediction window, and corresponding flare class label [5]

applications of convolutional neural network (ConvNet) on SDO AIA/HMI images [18, 19, 20, 21].

Angryk et al. [3] introduced temporal window-based flare prediction, which extends the earlier single timestamp-based models. The authors published an MVTS-based active region dataset, where each MVTS instance records magnetic field data for a preset observation time and uniform sampling rate, and is labeled by flare classes that occurred after a given prediction time. Among the MVTS classification approaches, Hamdi et al. [4] used statistical summarization of component univariate time series for training kNN classifier, Ma et. al. [22] applied MVTS decision trees that approached the problem using clustering as a preprocessing step, and Muzaheed et. al. [5] used LSTM-based deep sequence modeling for end-to-end flare classification that automated feature learning process avoiding hand-engineered statistical features.

Unlike previous models based on traditional ML and deep sequence learning, in this work, we present a model that leverages temporal as well as spatial relationships of the MVTS instances. Our model learns MVTS representations through an end-to-end fashion, and utilizes higher-order inter-variable relationships and local and global temporal changes.

3. MVTS representation learning by functional network and sequence embedding

3.1. Notations and Preliminaries

3.1.1. MVTS and Sub-MVTS

Each solar active region resulting in different flare classes (or staying as a flare quiet region) after a given prediction window represents a solar event. The solar event i is represented by a MVTS instance $S^{(i)}$, and associated by a class label $y^{(i)}$. The class label $y^{(i)}$ represents the flare quiet state, or flare classes of different intensities. The MVTS instance $S^{(i)} \in \mathbb{R}^{T \times N}$ is a collection of univariate time series of N magnetic field parameters, where each time series contains periodic observation values of the corresponding parameter for an observation period T. We denote the vector of t-th timestamp as $x^{<t>} \in \mathbb{R}^N$, and the time series represented by k-th parameter as $P_k \in \mathbb{R}^T$. After the observation period T and prediction period Δ, the event is labeled by the active region state (flare quiet or different flare classes). The active region state of a particular timestamp is found from the NOAA records of flaring events. Fig. 1 shows the MVTS-based data model of a solar event. Each MVTS instance is divided into η equal-length windows such that $T = \eta\tau$, where τ denotes window length. The *sub-MVTS* is denoted by $s \in \mathbb{R}^{\tau \times N}$, and s is a subsequence of S.

3.1.2. Node-attributed functional network

Functional network is a undirected and edge-weighted graph, and defined as $G = (V, E, W, X)$, where the set of nodes $V = \{P_1, ..., P_N\}$ denotes magnetic field parameters, $W : E \longrightarrow \mathbb{R}$ is a function of mapping edges to their weights, and node attribute matrix $X \in \mathbb{R}^{N \times \tau}$ contains the time series of each node in the sub-MVTS, i.e., $X = s^T$. The functional network is defined on the sub-MVTS, and the weight w_{ij} of edge e_{ij} (between node pair P_i and P_j) represents the statistical similarity of τ-length time series of P_i and P_j. Each functional network derived from a MVTS dataset has the same node set V.

3.1.3. Graph Convolution

For learning the representations of node-attributed functional networks, we use Graph Convolution Network (GCN). GCN is a widely used graph neural network [23] that learns node representations from a graph through layer-wise neighborhood aggregation. Graph convolution of layer l aggregates the representations of l-hop neighbors. GCN updates representation of node v in a graph $G = (V, E, W, X)$ by following equations.

$$h_v^{[0]} = x_v \tag{1}$$

$$h_v^{[l+1]} = ReLU\left(W_g^{[l]} \sum_{u \in N(v)} \frac{w_{uv} h_u^{[l]}}{|N(v)|} + B_g^{[l]} h_v^{[l]}\right), \quad \forall l \in \{0, 1, ..., L-1\} \tag{2}$$

$$z_v = h_v^{[L]} \tag{3}$$

$$z_G = \frac{1}{|V|} \sum_{v \in V} z_v \tag{4}$$

Here, L is the number of GCN layers, $x_v \in \mathbb{R}^\tau$ is the vector of node v, $h_v^{[l]} \in \mathbb{R}^{d_g}$ is the representation of node v in layer l, $W_g^{[l]} \in \mathbb{R}^{d_g \times d_g}$ is the weight matrix of layer

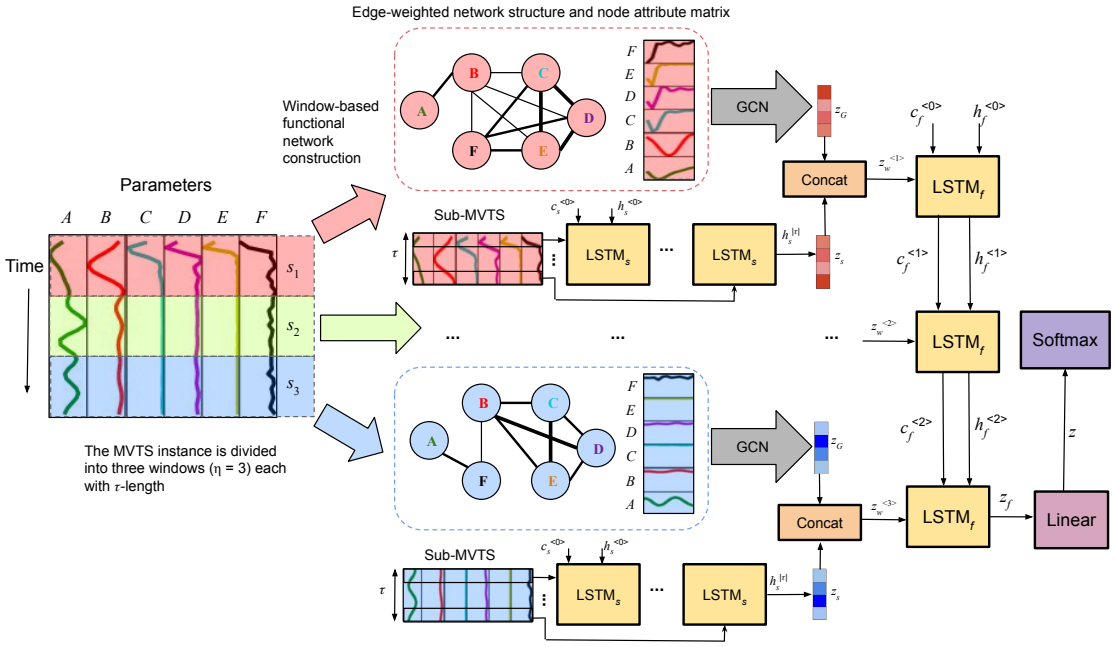


Figure 2: GCN-based node-attributed functional network embedding and LSTM-based local and global sequence embedding. For showing the functional network construction process, parameter set $\{P_1, P_2, .., P_N\}$ of the MVTS instance has been shown as $\{A, B, C, D, E, F\}$.

l, $B_g^{[l]} \in \mathbb{R}^{d_g}$ is the bias vector of layer l, $N(v)$ is the set of neighbor nodes of node v, w_{uv} is the weight associated in the edge between node v and its neighbor u, z_v is the final representation of node v after L iterations of neighborhood aggregation, and z_G is graph representation found by averaging the node representations.

3.1.4. Sequence embedding through LSTM

Long-short term memory (LSTM) networks [24] are frequently used for sequence representation learning which facilitates various tasks such as sequence classification, sequence-to-sequence translation, and so on. We use LSTM networks for learning low-dimensional representations of MVTS instances. The MVTS (and sub-MVTS) instances are sequences of N-dimensional timestamp vectors. The timestamp vector $x^{<t>} \in \mathbb{R}^N$ represents the magnetic filed state of the active region (N parameter values) in the timestamp t. We denote the inputs to the LSTM cells as $[x^{<1>}, x^{<2>}, x^{<3>}, ..., x^{<\gamma>}]$, cell state representations as $[c^{<0>}, c^{<1>}, c^{<2>}, ..., c^{<\gamma-1>}]$, and hidden state representations as $[h^{<0>}, h^{<1>}, h^{<2>}, ..., h^{<\gamma>}]$, where γ is the last timestamp of the sequence. After randomly initializing $c^{<0>}$ and $h^{<0>}$, we update the cell state and hidden state of the timestamp t by following LSTM equations [24].

$$\tilde{c}^{<t>} = tanh(W_c[h^{<t-1>}, x^{<t>}] + b_c) \quad (5)$$

$$\Gamma_u = \sigma(W_u[h^{<t-1>}, x^{<t>}] + b_u) \quad (6)$$

$$\Gamma_f = \sigma(W_f[h^{<t-1>}, x^{<t>}] + b_f) \quad (7)$$

$$\Gamma_o = \sigma(W_o[h^{<t-1>}, x^{<t>}] + b_o) \quad (8)$$

$$c^{<t>} = \Gamma_u \odot \tilde{c}^{<t>} + \Gamma_f \odot c^{<t-1>} \quad (9)$$

$$h^{<t>} = \Gamma_o \odot tanh(c^{<t>}) \quad (10)$$

We denote the number of dimensions of the cell state representation $c^{<t>}$ and hidden state representation $h^{<t>}$ of the LSTM cell as d_s. The concatenation of hidden state of previous timestamp and the input of current timestamp is $[h^{<t-1>}, x^{<t>}] \in \mathbb{R}^{d_s+N}$. The candidate cell state representation is $\tilde{c}^{<t>} \in \mathbb{R}^{d_s}$. The weight matrices are $W_c, W_u, W_f, W_o \in \mathbb{R}^{d_s \times (d_s+N)}$, and bias terms are $b_c, b_u, b_f, b_o \in \mathbb{R}$. The subscripts u, f, and o represents the activations of update gate, forget gate, and output gate respectively, while $\odot$ refers to elementwise multiplication, and σ represents sigmoid activation. Finally, we consider $h^{<\gamma>}$ as the final representation of the input MVTS.

3.2. Data Preprocessing

3.2.1. Node-level normalization

Since the magnetic field parameter values are recorded in different scales, we perform z-score normalization. Suppose that M number of MVTS instances each with N parameters and T time points are represented by a third-order tensor $\mathcal{X} \in \mathbb{R}^{M \times N \times T}$, where three modes represent events, parameters/nodes, and timestamps. For the better performance of the GCN-based graph embedding, we perform *node-level* z-normalization as a preprocessing step in the following three steps.

1. We perform mode-2 matricization, i.e., reshaping the tensor so that mode-2 (parameter/node) fibers become the columns of the matrix. The matrix is denoted by $X_{(2)} \in \mathbb{R}^{MT \times N}$. The columns are denoted by $P_1, P_2, \ldots, P_N$.
2. For each column P_j, we perform z-normalization as follows.

$$x_k^{(j)} = \frac{x_k^{(j)} - \mu^{(j)}}{\sigma^{(j)}}$$

   Here, $x_k^{(j)}$ is the k-th value of the column P_j, where $1 \leq k \leq MT$, $\mu^{(j)}$ is the mean of the column P_j, and $\sigma^{(j)}$ is the standard deviation of the column P_j.
3. We reshape the matrix $X_{(2)} \in \mathbb{R}^{MT \times N}$ back to third-order tensor, $\mathcal{X} \in \mathbb{R}^{M \times N \times T}$.

3.2.2. Functional network construction

We calculate the Pearson correlation matrix $C \in \mathbb{R}^{N \times N}$ for the sub-MVTS $s \in \mathbb{R}^{\tau \times N}$. In the correlation matrix, C_{ij} represents the Pearson correlation coefficient (in the range of [-1, 1]) between τ-length time series P_i and P_j. The symmetric matrix C can be considered as an adjacency matrix of a graph of N nodes. We apply a sparsity threshold of 0 so that only edges with positive weight (node pairs with positive correlation) are considered for functional network construction. We denote the sparse correlation matrix as the adjacency matrix $A \in \mathbb{R}^{N \times N}$. Although the functional network defined over a sub-MVTS encodes inter-variable interactions within a small temporal window, the adjacency matrix is not enough for the completeness of data, since negative correlation coefficients are discarded. To avoid the data missing, in addition to the adjacency matrix (graph structure), we extract the node attribute matrix $X = s^T$. In $X \in \mathbb{R}^{N \times \tau}$, each row represents node attributes in the form of τ-length time series (normalized in the previous step).

3.3. MVTS representation learning

In Fig. 2, we show the components of MVTS representation learning. Firstly, the window embedding learns the local spatiotemporal changes of the sub-MVTS instances through the models denoted as GCN and $LSTM_s$, and finally, the whole MVTS embedding learns global temporal changes of the local (window) representations through the model denoted as $LSTM_f$.

3.3.1. Window embedding

Our model learns the representation of the window s (sub-MVTS) of the MVTS instance S through GCN-based node-attributed functional network embedding and LSTM-based local sequence modeling.

- **GCN-based functional network embedding**: We input the node-attributed functional network $G(V, E, W, X)$ to a two-layer GCN. The initial node attributes are set as $X = s^T$ (Eq. 1). In the first layer, each node is embedded into a d'_g-dimensional space through 1-hop neighborhood aggregation, and after the second layer, each node is embedded into a d_g-dimensional space through 2-hop neighborhood aggregation (Eq. 2,3). Finally, the whole graph representation $z_G \in \mathbb{R}^{d_g}$ is computed through mean pooling (Eq. 4).
- **LSTM-based sub-MVTS embedding**: The sub-MVTS $s = [x^{<1>}, ..., x^{<\tau>}]$, where $x^{<t>} \in \mathbb{R}^N$, is sequentially input to the $LSTM_s$ (Eq. 5-10), and we extract the last hidden representation $z_s = h_s^{<\tau>}$, where $z_s \in \mathbb{R}^{d_s}$.

For the window embedding, we concatenate $z_G \in \mathbb{R}^{d_g}$ and $z_s \in \mathbb{R}^{d_s}$. Therefore, the window representation is $z_w \in \mathbb{R}^{d_g + d_s}$.

3.3.2. Whole MVTS embedding

After each of η windows is represented as $(d_g + d_s)$-dimensional vector, we feed the sequential data $[z_w^{<1>}, ..., z_w^{<\eta>}]$ into $LSTM_f$ for global temporal change modeling. Note that $LSTM_f$ and $LSTM_s$ have different learnable parameter sets (e.g., W_{u_s}, W_{u_f}, etc), although in this work the number of dimensions (d_s) in the cell state and hidden state are kept the same. We extract the final hidden state representation $z_f = h_f^{<\eta>}$, where $z_f \in \mathbb{R}^{d_s}$. We input z_f into a linear (fully connected) layer. In this layer, the parameters are $W_F \in \mathbb{R}^{n_c \times d_s}$, and $b_F \in \mathbb{R}$, where n_c is the number of classes. After this layer, we have a n_c-dimensional representation of the whole MVTS instance of event i.

$$z^{(i)} = ReLU(W_F z_f + b_F) \tag{11}$$

Finally, we input $z^{(i)} \in \mathbb{R}^{n_c}$ into a softmax layer, whose number of units is equal to the number of classes. The softmax layer gives us the normalized class probabilities, and we finally get $\hat{y}^{(i)} \in \mathbb{R}^{n_c}$.

$$\hat{y}^{(i)} = \frac{e^{z^{(i)}}}{\sum_{j=1}^{n_c} e^{z_j^{(i)}}} \tag{12}$$

The predicted labels of training MVTS instances are matched against true labels, and the Adam optimizer [25] updates the weight and bias parameter values of the GCN, $LSTM_s$, $LSTM_f$ and the fully connected layer through backpropagation algorithm. Algorithm 1 shows the training procedure of the proposed GCN-LSTM-based MVTS representation learning.

Algorithm 1 Training of GCN-LSTM-based MVTS representation learning

Input: Training set $\mathcal{D}$ consisted of functional network adjacency matrices $X_{adj} \in \mathbb{R}^{n_{train} \times \eta \times N \times N}$ and node attribute matrices $X_{nat} \in \mathbb{R}^{n_{train} \times \eta \times N \times \tau}$, one-hot training labels $y_{train} \in \mathbb{R}^{n_{train} \times n_c}$, number of epochs n_{epochs}, learning rate α, and weight decay factor of the Adam optimizer λ.
Output: Learned parameters of GCN, $LSTM_s$, and $LSTM_f$.

```
Randomly initialize parameter set W, which contains GCN, LSTM_s, and LSTM_f parameters
for number of training epochs n_epochs do
    for MVTS instance i = 1, 2, ..., n_train do
        Window matrix, Z_w = [0]_{η×(d_g+d_s)}
        for window j = 1, 2, ..., η do
            A ← X_adj[i, j, :, :]
            X ← X_nat[i, j, :, :]
            z_G ← GCN(A, X) //Eq. 1-4 (L = 2)
            z_s ← LSTM_s(X^T) //Eq. 5-10
            Z_w[j, :] ← Concat(z_G, z_s)
        end for
        z_f ← LSTM_f(Z_w) //Eq. 5-10
        z_f ← Linear(z_f) //Eq. 11
        z^(i) ← Softmax(z_f) //Eq. 12
        //negative log likelihood loss calculation
        L ← NLLLoss(z^(i), y_train^(i))
        Update W minimizing L by Adam(α, λ)
    end for
end for
return W
```

4. Experiments

In this section, we demonstrate our experimental findings. We compared the performance of our model with six other MVTS-based flare prediction baselines on a benchmark dataset. We used PyTorch 1.10.0 with CUDA 11.1 for implementing our GCN-LSTM-based MVTS classifier. The source code of our model and the experimental dataset are available at our GitHub repository. [1]

4.1. Dataset

As the benchmark dataset of our experiments, we used the solar flare prediction dataset published by Angryk et. al. [3]. Each MVTS instance in the dataset is made up of 25 time series of active region magnetic field parameters (for the full list of parameters, see [16]). The time series instances are recorded at 12 minutes intervals for a total duration of 12 hours (60 time steps). The MVTS instances are labeled according to the flaring event that occurred after 12 hours. Therefore, the dataset has the number of the observation points $T = 60$, and the number of dimensions in timestamp vectors $N = 25$, while the prediction window is $\Delta = 12$ hours. Our experimental dataset consists of 1,540 MVTS instances evenly distributed across four classes (X, M, BC, and Q), where BC represents events from both B and C classes (less intense flares). We split the dataset into train and test using the stratified holdout method (two-thirds for training and one-third for the test).

4.2. Baseline methods

We evaluated our GCN-LSTM-based MVTS classification model with six other baselines.

- **Flattened vector method (FLT)**: This is a naive method, where each 60×25 MVTS instance is flattened into a $1,500$-dimensional vector.
- **Vector of last timestamp (LTV)**: This method was introduced by Bobra et al [16], where vector magnetogram data (feature space of all magnetic field parameters) were used for classification. Since the last timestamp of the MVTS is temporally nearest to the flaring event, we sampled the vector of the last timestamp (25-dimensional) to train the classifier.
- **Time series summarization-based MVTS representation (TS-SUM)**: This method, proposed by Hamdi et al [4] summarizes each individual time series of length T by eight statistical features: mean, standard deviation, skewness, and kurtosis of the original time series, and the first-order derivative of the time series. As a result, we get an 8×25-dimensional vector space, which is used for training the downstream classifier.
- **Long-short term memory (LSTM):** This LSTM-based approach was proposed by Muzaheed et.

[1] https://github.com/FuadAhmad/GCN-LSTM

Table 1
Multiclass classification performance of the proposed method with the baselines

Measures	*FLT*	*LTV*	*TS-SUM*	*RNN*	*LSTM*	*ROCKET*	*GCN-LSTM*
Accuracy	0.259 ± 0.012	0.323 ± 0.02	0.609 ± 0.091	0.427 ± 0.025	0.628 ± 0.03	0.742 ± 0.021	$\mathbf{0.817 \pm 0.014}$
Precision (X)	0.232 ± 0.024	0.342 ± 0.041	0.712 ± 0.054	0.534 ± 0.031	0.757 ± 0.028	0.92 ± 0.034	$\mathbf{0.932 \pm 0.022}$
Recall (X)	0.264 ± 0.053	0.392 ± 0.043	0.772 ± 0.024	0.631 ± 0.028	0.947 ± 0.023	0.981 ± 0.016	$\mathbf{0.99 \pm 0.023}$
F1 (X)	0.244 ± 0.032	0.362 ± 0.04	0.741 ± 0.034	0.582 ± 0.019	0.841 ± 0.014	0.952 ± 0.028	$\mathbf{0.961 \pm 0.013}$
Precision (M)	0.254 ± 0.012	0.324 ± 0.033	0.522 ± 0.031	0.411 ± 0.014	0.594 ± 0.018	0.661 ± 0.042	$\mathbf{0.803 \pm 0.054}$
Recall (M)	0.26 ± 0.023	0.331 ± 0.061	0.552 ± 0.022	0.402 ± 0.03	0.544 ± 0.014	0.704 ± 0.038	$\mathbf{0.824 \pm 0.063}$
F1 (M)	0.257 ± 0.026	0.327 ± 0.042	0.537 ± 0.023	0.406 ± 0.029	0.568 ± 0.02	0.687 ± 0.028	$\mathbf{0.811 \pm 0.033}$
Precision (BC)	0.232 ± 0.044	0.263 ± 0.024	0.453 ± 0.033	0.282 ± 0.031	0.495 ± 0.013	0.58 ± 0.026	$\mathbf{0.682 \pm 0.03}$
Recall (BC)	0.241 ± 0.053	0.212 ± 0.02	0.472 ± 0.014	0.261 ± 0.021	0.409 ± 0.023	0.573 ± 0.052	$\mathbf{0.664 \pm 0.05}$
F1 (BC)	0.236 ± 0.041	0.234 ± 0.024	0.462 ± 0.041	0.271 ± 0.031	0.448 ± 0.031	0.577 ± 0.031	$\mathbf{0.673 \pm 0.032}$
Precision (Q)	0.324 ± 0.034	0.343 ± 0.044	0.583 ± 0.045	0.483 ± 0.024	0.603 ± 0.024	0.81 ± 0.046	$\mathbf{0.831 \pm 0.018}$
Recall (Q)	0.251 ± 0.042	0.362 ± 0.071	0.663 ± 0.034	0.413 ± 0.042	0.683 ± 0.023	0.724 ± 0.034	$\mathbf{0.772 \pm 0.021}$
F1 (Q)	0.282 ± 0.014	0.352 ± 0.013	0.62 ± 0.043	0.445 ± 0.032	0.64 ± 0.024	0.771 ± 0.036	$\mathbf{0.798 \pm 0.017}$

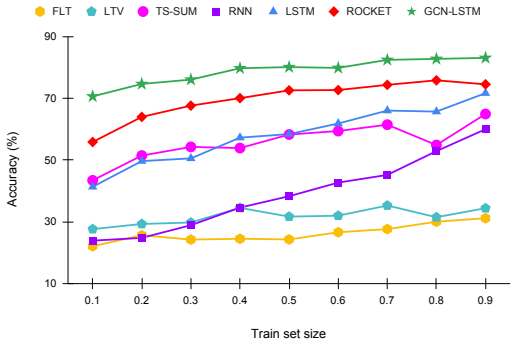
al. [5]. Each MVTS instance was considered as a T-length sequence of $x^{<t>} \in \mathbb{R}^N$ timestamp vectors. After sequentially feeding the LSTM model with each timestamp vector, the last hidden representation was considered as the MVTS representation. Following the same experimental setting, we use the number of both cell state and hidden state dimensions as 128, the number of training epochs as 500, and the learning rate in stochastic gradient descent as 0.01.

- **Recurrent Neural Network (RNN):** As the fifth baseline, we replace LSTM cells of the model of [5] with standard RNN cells. Similar to the experimental setting of [5], we use the number of RNN hidden dimensions as 128, the number of training epochs as 1,000, and the learning rate in stochastic gradient descent as 0.01.
- **Random Convolutional Kernel Transform (ROCKET):** We use ROCKET [26] as the sixth baseline for MVTS-based solar event classification. ROCKET was shown as the best performing algorithm in the MVTS classification benchmarking study by Ruiz et al [27], which included 26 MVTS datasets of the UEA archive [28]. ROCKET uses a large number of random convolution kernels in conjunction with a linear classifier (ridge regression or logistic regression), where each kernel is applied to each univariate time series instance. Similar to the experimental setting of [27], we used the number of kernels in ROCKET as 10,000.

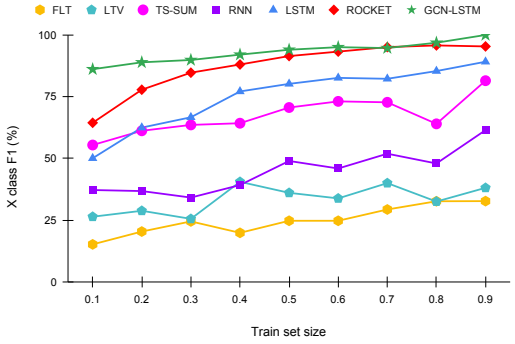
The first three baselines are *embedding followed by classification* methods. After performing the embedding of MVTS instances using those methods, we use logistic regression classifier with L2 regularization. In all the experiments, we split the dataset into train and test using the stratified holdout method (two-thirds for training and one-third for the test). In the experiments of the proposed GCN-LSTM model, we have following hyperparameters: # windows, η : 4, window length, τ: 15, # hidden dimensions d'_g in first GCN layer: 64, # node embedding dimensions d_g in second GCN layer: 4, # dimensions in cell state and hidden state representations d_s of both $LSTM_s$ and $LSTM_f$: 128, # training epochs: 100, Adam learning rate α: 10^{-4}, and weight decay (regularization factor) λ: 10^{-3}.

4.3. Multiclass classification performance

In Table 1, we show the classification performances of our GCN-LSTM-based MVTS classifier along with that of the baseline methods. For a comprehensive classification report, we show accuracy along with precision, recall, and F1 of each class. We performed five experiments with different train/test sets sampled by stratified holdout (two-thirds for training and one-third for the test) and reported the mean and standard deviation of the experiments. From the results, it is visible that the GCN-LSTM-based MVTS classification model outperforms all other baselines in all the performance measures. In overall evaluation, ROCKET achieves second-bast performance, while the LSTM model becomes third. GCN-LSTM model achieves around 20% more accuracy in comparison with the LSTM model, which proves the importance of learning MVTS representations in both spatial and temporal domains rather than learning only from the temporal domain. Among shallow ML models, TS-SUM performs better than FLT and LTV models. In general, the high performances of TS-SUM, RNN, LSTM, ROCKET, and GCN-LSTM prove the importance of time series representations of solar events.



(a) Multiclass classification accuracy with increasing training data



(b) F1 of X class with increasing training set size

Figure 3: Multiclass classification with varying train set size

4.4. Classification varying train set size

To verify the adaptability of our model with bigger training datasets, we experimented by varying the training set size. We varied the training set size from 10% to 90% of the dataset size, while testing the models with the rest of the instances (Fig. 3). We performed stratified train/test sampling with a given training set size, and evaluated the classification performance of the classifiers five times with five distinct samples of training and test sets. In Fig. 3a and 3b, we plotted the mean accuracy values and mean F1 (X class) values found in all runs of different train/test samples with different training data sizes. GCN-LSTM consistently outperforms other baselines in all settings of training set sizes. ROCKET is the second-best performing classifier in this experiment, and especially in F1 measure ROCKET exhibits similar robust performance to GCN-LSTM. With only 10% training data, GCN-LSTM achieved 70% classification accuracy, while the third-best performing LSTM model achieve that level of high performance by using 90% training data. Although all models gain more accuracy with a gradual

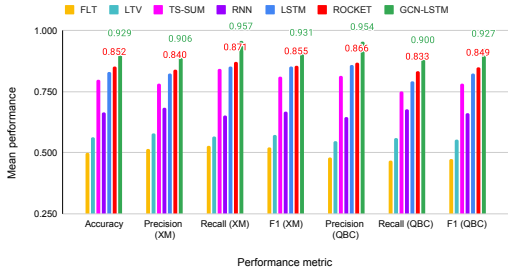


Figure 4: Binary classification performance of all baselines

increase of training set size, we observe more consistent increasing patterns in deep learning and kernel-based methods, e.g., GCN-LSTM, ROCKET, LSTM, and RNN. It proves that with sufficiently large datasets, deep learning models can outperform the traditional classifiers or embedding methods in a larger margin. The time series summarization-based method TS-SUM shows promising performance throughout the experiments, but the generalization capability of this model can be limited in a more complex dataset due to its less flexible learning methodology consisting of hand-engineered features. Compared to the deep learning-based and time series-based methods, the LTV and FLT models perform poorly, which proves the importance of time series in avoiding underfitting.

4.5. Binary classification performance

In addition to classifying the solar active regions in different flare classes, a major use case in data-driven flare prediction is the binary classification, i.e., distinguishing major flaring events from minor flaring events or flare quiet events. In this experiment, we considered X and M class MVTS instances as flaring events, while we considered all other instances (Q and BC) as non-flaring events. In Fig. 4, we show the mean binary classification performances of all models over five different train/test samples in terms of accuracy, precision, recall, and F1 of flaring and non-flaring classes. It is clearly visible that the GCN-LSTM model outperforms all other baselines. We reported the performances of the two best-performing models in numbers along with their bars. In all performance metrics, GCN-LSTM achieves an average of $\sim 8\%$ better performance than the second-best performing ROCKET algorithm. In general, we observe the similar performance of the models as that of multiclass classification. Although one deep learning model, i.e., the RNN-based model performed poorer than the TS-SUM method, the RNN-based model is an end-to-end classification model, which might outperform TS-SUM with more training data, more complex model, and more efficient hyperparameter tuning.

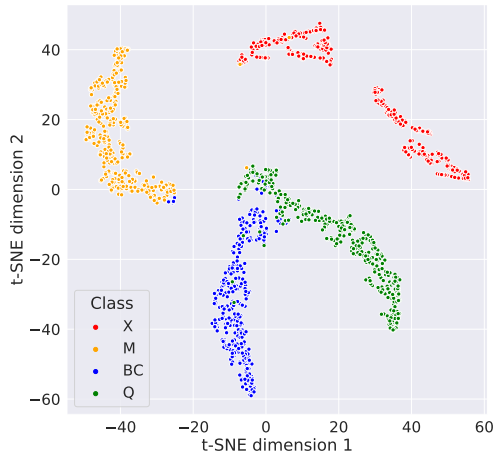


Figure 5: t-SNE embedding of GCN-LSTM generated representations of all MVTS instances in the dataset

4.6. Embedding performance

Visualization of high-dimensional data in 2D/3D space is a well-known method of demonstrating the effectiveness of learned representations. To investigate the quality of learned MVTS representations, we provide a visualization of t-SNE [29] transformed MVTS representations extracted by the final layer of the GCN-LSTM model. Similar to section 4.3, the stratified holdout strategy is taken to pre-train the model, and all instances are projected to t-SNE-reduced 2D space (Fig. 5). The 2D projection exhibits discernible clustering of the MVTS instances. Some meaningful insights are observed by the t-SNE scatter plot such as (1) patterns of four classes are easily recognizable, (2) flare-quiet events (Q) and minor flaring events (B and C) are comparatively similar, (3) X and M class flares exhibit significant dissimilarity from other classes, (4) some flare-quiet events are similar to the minor flaring events, (5) few minor flares show similar characteristics to M-class flares, and (6) the characteristics of the X-class flares are exclusive, and other class instances do not show any similarity with X-class instances.

5. Conclusion

In this work, we presented an end-to-end deep learning-based flare prediction model from multivariate time series (MVTS) represented datasets that leverages inter-variable relationships by graph convolutional network-based functional network embedding, and local and global temporal change modeling through LSTM-based sequence embedding. In contrary to other MVTS classification models applied for flare prediction, our model utilizes spatial and temporal features of the MVTS instances, and does not depend on predefined statistical features. Our experiments on a real-life solar flare prediction dataset demonstrate the superior performance of our model in performing multiclass and binary MVTS classification.

In the future, we look forward to designing more efficient models by techniques such as (1) learning attention coefficients in spatial and temporal feature spaces, (2) customizing transformer models for MVTS representations, and (3) analyzing the effects of univariate sequence embedding towards MVTS representation learning. We will also apply our models in other MVTS-based solar event datasets (e.g., solar energetic particles) [30], and MVTS datasets generated from other sources such as functional MRI (fMRI)-based time series of brain regions [31].

6. Acknowledgments

This project has been supported in part by funding from CISE and GEO directorates under NSF awards #2153379 and #2204363.

References

[1] J. Eastwood, E. Biffis, M. Hapgood, L. Green, M. Bisi, R. Bentley, R. Wicks, L.-A. McKinnell, M. Gibbs, C. Burnett, The economic impact of space weather: Where do we stand?, Risk Analysis 37 (2017) 206–218.

[2] N. Science, T. Council, National space weather action plan, https://obamawhitehouse.archives.gov/sites/default/files/microsites/ostp/final_nationalspaceweatheractionplan_20151028.pdf, 2015. [Accessed: 10-Feb-2022].

[3] R. A. Angryk, P. C. Martens, B. Aydin, D. Kempton, S. S. Mahajan, S. Basodi, A. Ahmadzadeh, X. Cai, S. F. Boubrahimi, S. M. Hamdi, et al., Multivariate time series dataset for space weather data analytics, Scientific data 7 (2020) 1–13.

[4] S. M. Hamdi, D. Kempton, R. Ma, S. F. Boubrahimi, R. A. Angryk, A time series classification-based approach for solar flare prediction, in: 2017 IEEE Intl. Conf. on Big Data (Big Data), IEEE, 2017, pp. 2543–2551.

[5] A. A. M. Muzaheed, S. M. Hamdi, S. F. Boubrahimi, Sequence model-based end-to-end solar flare classification from multivariate time series data, in: 20th IEEE Intl. Conf. on Machine Learning and Applications, ICMLA 2021, Pasadena, CA, USA, December 13-16, 2021, IEEE, 2021, pp. 435–440.

[6] Z. Wu, S. Pan, G. Long, J. Jiang, X. Chang, C. Zhang, Connecting the dots: Multivariate time series forecasting with graph neural networks, in: KDD '20: The 26th ACM SIGKDD Conf. on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020, ACM, 2020, pp. 753–763.

[7] P. S. McIntosh, The classification of sunspot groups, Solar Physics 125 (1990) 251–267.

[8] J. P. Mason, J. Hoeksema, Testing automated solar flare forecasting with 13 years of michelson doppler imager magnetograms, The Astrophysical Journal 723 (2010) 634.

[9] Y. Cui, R. Li, L. Zhang, Y. He, H. Wang, Correlation between solar flare productivity and photospheric magnetic field properties, Solar Physics 237 (2006) 45–59.

[10] J. Jing, H. Song, V. Abramenko, C. Tan, H. Wang, The statistical relationship between the photospheric magnetic parameters and the flare productivity of active regions, The Astrophysical Journal 644 (2006) 1273.

[11] K. Leka, G. Barnes, Photospheric magnetic field properties of flaring versus flare-quiet active regions. ii. discriminant analysis, The Astrophysical Journal 595 (2003) 1296.

[12] H. Song, C. Tan, J. Jing, H. Wang, V. Yurchyshyn, V. Abramenko, Statistical assessment of photospheric magnetic features in imminent solar flare predictions, Solar Physics 254 (2009) 101–125.

[13] D. Yu, X. Huang, H. Wang, Y. Cui, Short-term solar flare prediction using a sequential supervised learning method, Solar Physics 255 (2009) 91–105.

[14] O. W. Ahmed, R. Qahwaji, T. Colak, P. A. Higgins, P. T. Gallagher, D. S. Bloomfield, Solar flare prediction using advanced feature extraction, machine learning, and feature selection, Solar Physics (2013) 1–19.

[15] A. Al-Ghraibah, L. Boucheron, R. McAteer, An automated classification approach to ranking photospheric proxies of magnetic energy build-up, Astronomy & Astrophysics 579 (2015) A64.

[16] M. G. Bobra, S. Couvidat, Solar flare prediction using SDO/HMI vector magnetic field data with a machine-learning algorithm, The Astrophysical Journal 798 (2015) 135.

[17] N. Nishizuka, K. Sugiura, Y. Kubo, M. Den, S. Watari, M. Ishii, Solar flare prediction model with three machine-learning algorithms using ultraviolet brightening and vector magnetograms, Astrophysical Journal 835 (2017) 156.

[18] Y. Zheng, X. Li, X. Wang, Solar flare prediction with the hybrid deep convolutional neural network, The Astrophysical Journal 885 (2019) 73.

[19] X. Li, Y. Zheng, X. Wang, L. Wang, Predicting solar flares using a novel deep convolutional neural network, The Astrophysical Journal 891 (2020) 10.

[20] E. Park, Y.-J. Moon, S. Shin, K. Yi, D. Lim, H. Lee, G. Shin, Application of the deep convolutional neural network to the forecast of solar flare occurrence using full-disk solar magnetograms, The Astrophysical Journal 869 (2018) 91.

[21] N. Nishizuka, Y. Kubo, K. Sugiura, M. Den, M. Ishii, Operational solar flare prediction model using deep flare net, Earth, Planets and Space 73 (2021) 1–12.

[22] R. Ma, S. F. Boubrahimi, S. M. Hamdi, R. A. Angryk, Solar flare prediction using multivariate time series decision trees, in: 2017 IEEE Intl. Conf. on Big Data, BigData 2017, Boston, MA, USA, December 11-14, 2017, IEEE Computer Society, 2017, pp. 2569–2578.

[23] T. N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, in: 5th Intl. Conf. on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings, OpenReview.net, 2017.

[24] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (1997) 1735–1780.

[25] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, in: 3rd Intl. Conf. on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conf. Track Proc., 2015.

[26] A. Dempster, F. Petitjean, G. I. Webb, ROCKET: exceptionally fast and accurate time series classification using random convolutional kernels, Data Min. Knowl. Discov. 34 (2020) 1454–1495.

[27] A. P. Ruiz, M. Flynn, J. Large, M. Middlehurst, A. Bagnall, The great multivariate time series classification bake off: a review and experimental evaluation of recent algorithmic advances, Data Mining and Knowledge Discovery 35 (2021) 401–449.

[28] A. J. Bagnall, H. A. Dau, J. Lines, M. Flynn, J. Large, A. Bostrom, P. Southam, E. J. Keogh, The UEA multivariate time series classification archive, 2018, CoRR abs/1811.00075 (2018). URL: http://arxiv.org/abs/1811.00075. arXiv:1811.00075.

[29] L. Van der Maaten, G. Hinton, Visualizing data using t-sne., Journal of machine learning research 9 (2008).

[30] S. F. Boubrahimi, S. M. Hamdi, R. Ma, R. A. Angryk, On the mining of the minimal set of time series data shapelets, in: IEEE Intl. Conf. on Big Data, Big Data 2020, Atlanta, GA, USA, December 10-13, 2020, IEEE, 2020, pp. 493–502.

[31] S. M. Hamdi, B. Aydin, S. F. Boubrahimi, R. A. Angryk, L. C. Krishnamurthy, R. D. Morris, Biomarker detection from fmri-based complete functional connectivity networks, in: IEEE Intl. Conf. on Artificial Intelligence and Knowledge Engineering, AIKE 2018, Laguna Hills, CA, USA, September 26-28, 2018, IEEE, 2018, pp. 17–24.