Supporting Information for "Regional heatwave prediction using Graph Neural Network and weather station data "

Peiyuan Li[1], Yin Yu[2], Daning Huang[2], Zhi-Hua Wang[3], Ashish Sharma[1,4]

[1]Discovery Partners Institute, University of Illinois System, Chicago, IL, 60606, USA

[2]Department of Aerospace Engineering, Pennsylvania State University, University Park, PA, 16802, USA

[4]School of Sustainable Engineering and the Built Environment, Arizona State University, Tempe, AZ, 85205, USA

[3]Environmental Science Division, Argonne National Laboratory, Lemont, IL 60439 USA

Contents of this file

Text S1: Purposed Architecture. The GNN-based architecture uses an encoder-processor-decoder architecture as shown in Fig. S3. The key components of the architecture are detailed in the following.

1. Encoder: First, the encoder is applied to each individual node. It maps the state vectors at a node $\mathbf{x}_i$ to a latent vector $\mathbf{h}_i^0 \in \mathbb{R}^D$. The latent vector is a set of high-dimensional nonlinear features that provide a continuous representation of the states on each station, which can be discrete-valued; continuous features are more amenable for NN

computations. For the i^{th} node at time step k, the encoder $\mathbf{f}_E$ is

$$\mathbf{h}_i^0 = \mathbf{f}_E(\mathbf{x}_i^{(k)}, \mathbf{x}_i^{(k-1)}, \cdots, \mathbf{x}_i^{(k-C+1)}; \mathbf{\Theta}^0), \tag{1}$$

where $\mathbf{f}_E$ is implemented as a standard fully-connected NN (FCNN) of N_M layers with a set of trainable parameters $\mathbf{\Theta}^0$. After the encoding, the latent vectors of all the nodes are denoted $\mathbf{H}^0 = \{\mathbf{h}_i^0\}_{i=1}^N \in \mathbb{R}^{N \times D}$.

2. Processor: Next, a stack of N_p GAL layers serve as the processors that successively aggregate the latent features from each node and its neighbors, and update the latent vectors at each node. Formally, the j^{th} processor step is written as

$$\mathbf{H}^{j+1} = \mathbf{f}_P^j(\mathbf{H}^j; \mathbf{\Theta}^j; \mathcal{G}), \tag{2}$$

where $\mathbf{f}_P^j$ is a GAL with parameter $\mathbf{\Theta}^j$ and $\mathcal{G}$ is the graph represented by the adjacency matrix. Note that when there are multiple MP steps, the latent vector of a node is influenced not only by its direct neighbors, but also by the more distant nodes. This distant influence behavior resembles the long-range interaction between different weather station locations. Together with the updated latent vectors $\mathbf{H}^{N_p}$, the GALs also generate the attention-based adjacency matrix $A_\alpha^{N_p}$, which is further studied for the geological relations among stations.

3. Decoder: Finally, the decoder maps the latent vector of each node to the desired output, i.e., the binary flags of the HW for the future time horizon,

$$[\beta^1, \beta^2, \cdots, \beta^{C_{\text{out}}}] = \mathbf{f}_D(\mathbf{H}^N; \mathbf{\Theta}^{N+1}) \tag{3}$$

where $\mathbf{f}_D$ is a FCNN of N_M layers with trainable parameters $\mathbf{\Theta}^{N+1}$.

Text S2: Description of Confusion Matrix. The heatwave forecasting problem is formulated as a binary classification, where one can use the confusion matrix to evaluate

the performance of an algorithm. The outcome of a binary classification is either *Positive (P)*, indicating the occurrence of a heatwave event, or otherwise *Negative (N)*. Then in the context of heatwave prediction, the four possible results of the confusion matrix can be described as follows:

1. True positive (TP): the model correctly predicts an event of heatwave.

2. True negative (TN): the model correctly predicts that heatwave does not occur.

3. False positive (FP): the model incorrectly predicts an event of heatwave, where in reality, heatwave does not occur. This is also known as the type I error.

4. False negative (FN): the model incorrectly predicts that heatwave does not occur, where in reality, heatwave does occur. this is also known as the type II error.

Based on the confusion matrix, three metrics that are commonly analyzed, defined as

1. Accuracy $= \frac{\mathrm{TP+TN}}{\mathrm{P+N}}$, observing the overall proportion of correct classifications.

2. Recall $= \frac{\mathrm{TP}}{\mathrm{TP+FN}}$, indicating the proportion the actual heatwave events that are correctly predicted by the model.

3. Precision $= \frac{\mathrm{TP}}{\mathrm{TP+FP}}$, showing how many of the predicted heatwave events actually occurred.

Text S3: F1-Score. A measurement that is typically used for imbalanced data is known as the *F1-score*. The F1-score combines the recall and precision into a single metric by taking their harmonic mean, defined as

$$\mathrm{F1} = \frac{2 \cdot \mathrm{precision} \cdot \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}} = \frac{2\mathrm{TP}}{2\mathrm{TP} + \mathrm{FP} + \mathrm{FN}} \tag{4}$$

The native F1-score function is based on discrete binary outputs and hence not differentiable. Therefore F1-score cannot be used as a loss function in ML model train-

ing. We deployed the soft-F1-score as the loss function to train our model. In this case, the number TP, FP, and FN are computed as a continuous sum of likelihood probability values instead of discrete integer values, which makes the resulting F1-score continuous and differentiable. The soft-F1-score used in the network training follows an open-source implementation can be found via `https://www.kaggle.com/code/rejpalcz/best-loss-function-for-f1-score-metric/notebook`.

Text S4: Graph Attention Layer. The Graph Attention Layer (GAL) is a popular implementation of message passing (MP). In the general attention mechanism, for node u and its neighbor v, with feature vectors $\mathbf{h}_u^j$ and $\mathbf{h}_v^j$, one may compute m attention coefficients $\{\alpha_{uv}^j\}_{k=1}^m$ as

$$\alpha_{uv}^j = \frac{\exp(\text{LeakyReLU}(\mathbf{f}(\mathbf{h}_u^j, \mathbf{h}_v^j; \mathbf{\Theta}_f^j)))}{\sum_{w\in\mathcal{N}(u)} \exp(\text{LeakyReLU}(\mathbf{f}(\mathbf{h}_u^j, \mathbf{h}_w^j; \mathbf{\Theta}_f^j)))}. \tag{5}$$

In the attention calculation (5), $\mathbf{f}_k$ is nonlinear function, such as a neural network, which characterizes the correlation between two feature vectors; LeakyReLU is a nonlinear activation function $f(x) = \max(-\epsilon x, x)$, where $\epsilon = 0.2$ as a typical choice; the sum-of-exp formulation normalizes the correlation to produce $\alpha_{uv}^j \in [0, 1]$. Subsequently, defining a set of new adjacency matrices, $[\mathbf{A}_\alpha^j]_{uv} = \alpha_{uv}^j$, the node features are updated as

$$\mathbf{H}^{j+1} = \sigma(\mathbf{A}_\alpha^j \mathbf{H}^j \mathbf{\Theta}_\alpha^j) \in \mathbb{R}^{N\times D}. \tag{6}$$

Note that in general $\mathbf{A}_\alpha^j$ is not symmetric. It is possible that a node u is strongly influenced by its neighbor v, quantified by a large α_{uv}, but not vice versa.

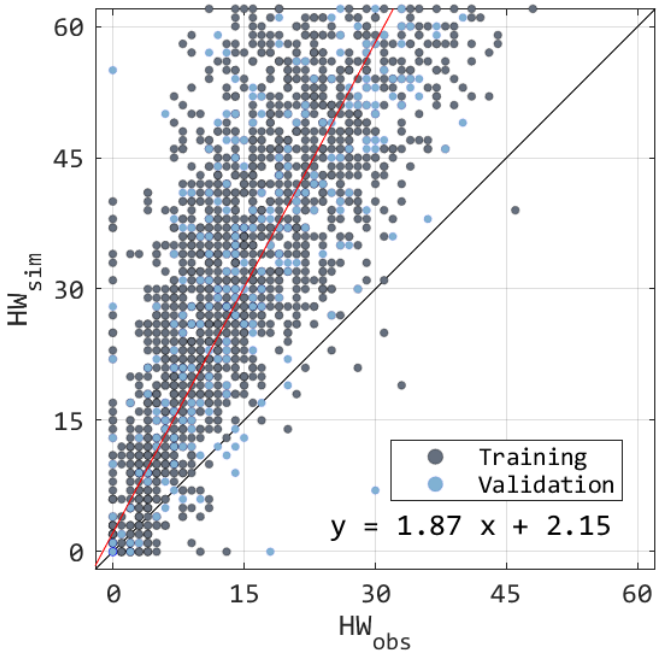

Figure S1. A model with a high recall rate ($> 92\%$). Compared a balanced model (Fig. 2a), this model overpredicts the occurance of HW events.

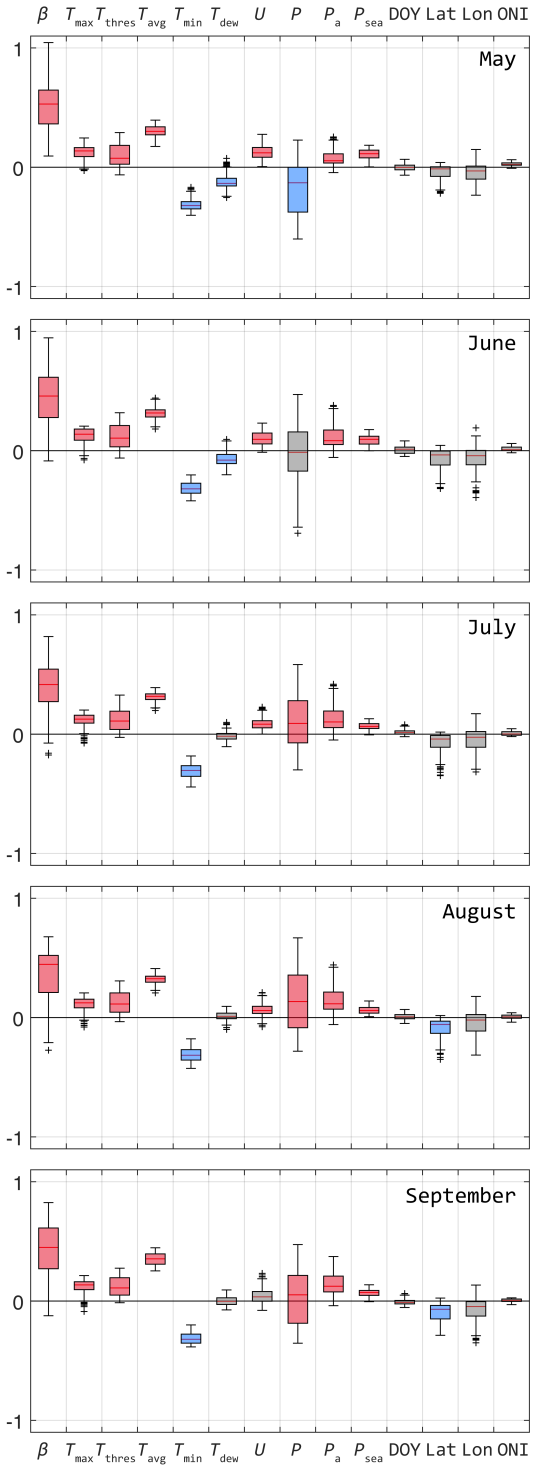

Figure S2. Monthly variations of the sensitivity scores (SS) for 14 model input features. Compared to the averaged SS (Fig. 3a), precipitation (P) shows the largest variation across the summer months.

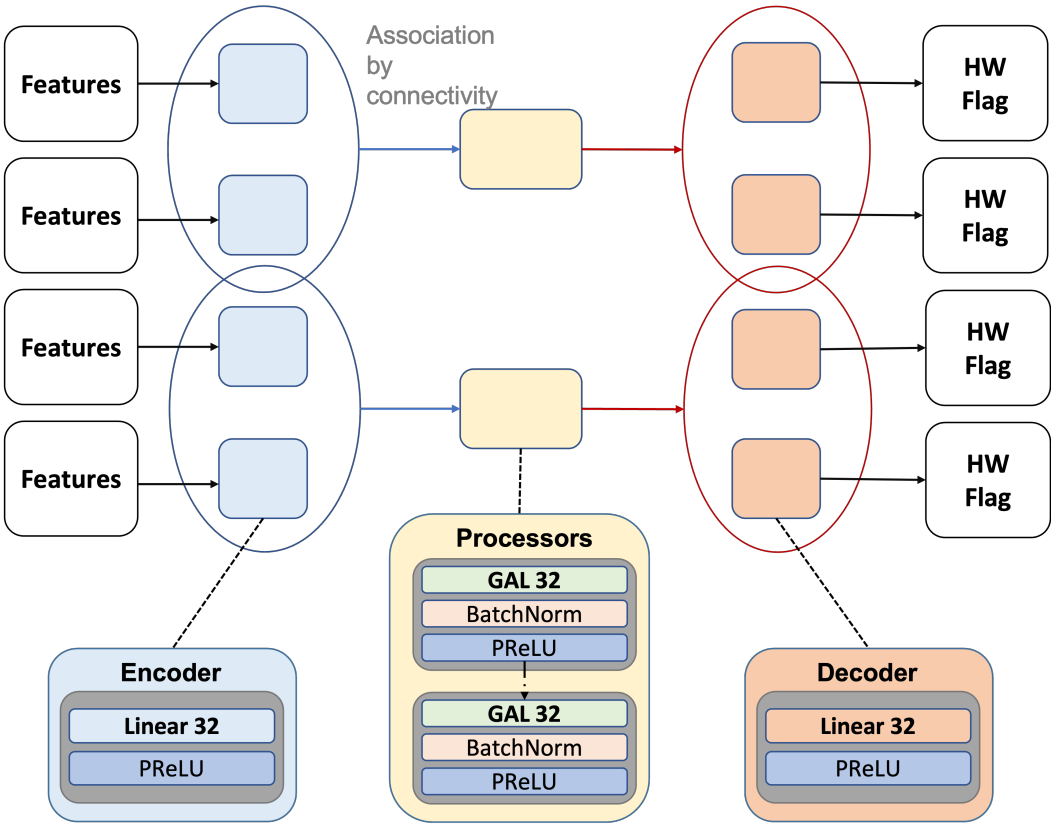

Figure S3. The purposed GNN framework uses an encoder-processor-decoder architecture.