Geoscientific
Model Development

Supplement of

Interpolating turbulent heat fluxes missing from a prairie observation on the Tibetan Plateau using artificial intelligence models

Quanzhe Hou et al.

Correspondence to: Zhiqiu Gao (zgao@nuist.edu.cn)

S1. To comprehensively assess the reliability and efficacy of the Transformer_CNN model in estimating turbulent heat fluxes (H and LE), this study employed a Taylor diagram for comparative analysis with six other advanced artificial intelligence models.

The Taylor diagram serves as an effective tool for quantifying the correlation and discrepancies between model predictions and observed data. Within the diagram, angles represent the correlation coefficient between model predictions and observed values; the bold dashed line signifies the standard deviation of model predictions, while the light-colored semi-circle dashed line represents the root mean square error (RMSE) between the model predictions and the actual values. As a reference benchmark, the EC (eddy covariance) observed values were used to evaluate the performance of each model. The results, as depicted in Fig. 1, detail the performance of each model on the test set. Encouragingly, the Transformer_CNN model's estimation of turbulent heat fluxes, whether viewed from the RMSE or correlation coefficient perspective, markedly surpasses the other six artificial intelligence models. Specifically, the Transformer_CNN model exhibits the smallest RMSE, with 0.29 for sensible heat flux and 0.31 for latent heat flux, while its model correlation coefficient consistently exceeds 0.94, again outperforming the other six AI models. These outcomes not only validate the model's exemplary performance in fitting turbulent heat fluxes but also underscore its superior generalization capability. Compared to traditional statistical methods and other artificial intelligence techniques, the Transformer_CNN model offers a more precise and robust solution, bearing significant practical implications for areas like climate research and meteorological forecasting.
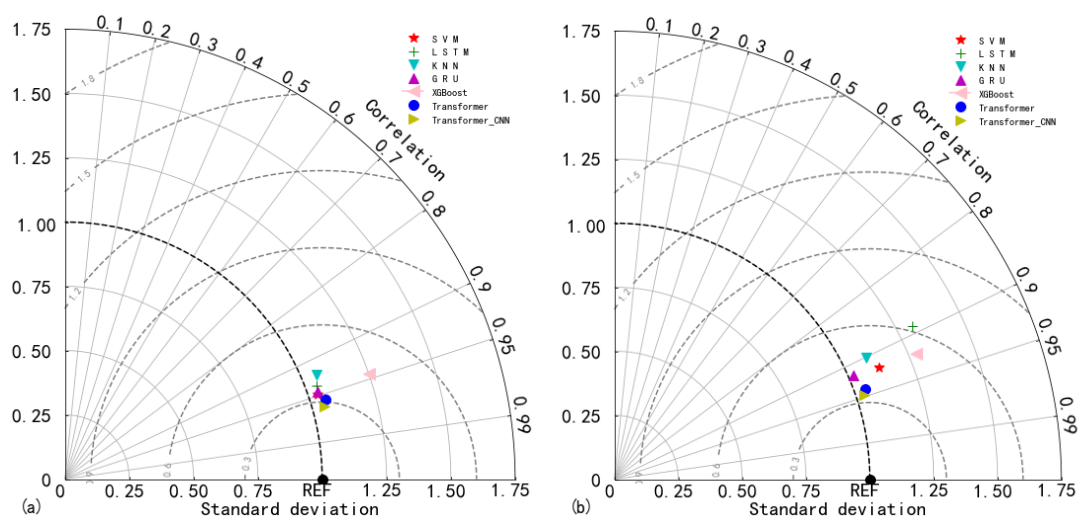
Fig. S1: The Taylor diagram displays the performance of seven models on the test set data: a) Sensible heat flux (H); b) Latent heat flux (LE).

S2. Because of the limited length of the paper, in order to describe the LSTM, GRU and Transforme models more clearly, the specific parameters of the models are given here.

S2.1 LSTM:

S2.1.1 Architecture-Related Hyperparameters

Number of Channels in Convolutional Layer (128): The input convolutional layer is set to have 128 output channels. This number determines the amount of features the model can capture in the initial processing of input data. A higher number of channels can improve the model's ability to capture complex features.

Hidden State Size of LSTM Layer (64): The dimension of the hidden state for the LSTM layer is set to 64. This determines the capacity of LSTM units to remember information. For complex temporal data modeling tasks, choosing the appropriate size for the hidden state is key to balancing model complexity and performance.

Number of LSTM Layers (3): The LSTM is configured with 3 layers. Multiple LSTM layers can enhance the model's learning ability, especially for data with complex temporal dependencies, but this also significantly increases the number of model parameters and the difficulty of training.

Use of Bidirectional LSTM (bidirectional=True): Employing a bidirectional LSTM allows the model to learn both forward and backward dependencies in time series data, which has been proven beneficial in many time series analysis tasks, particularly in scenarios requiring capture of global temporal information.

S2.1.2 Training-Related Hyperparameters

Dropout Ratios (0.15 and 0.3): Dropout is a regularization technique used to prevent over fitting by randomly dropping a portion of neural network nodes during training. In this model, two different dropout ratios are used to provide varying degrees of regularization across different network layers.

Non-linear Activation Functions between Linear and LSTM Layers (ReLU and GELU): ReLU and GELU activation functions are employed to introduce non-linearity, aiding the model in learning complex function mappings. GELU, compared to ReLU, can offer smoother gradients in certain cases, facilitating the learning process.

S2.1.3 Loss Functions

MSELoss and SmoothL1Loss: These two loss functions are used for different outputs of the model. MSE (Mean Squared Error) loss is highly sensitive to outliers, while Smooth L1 loss is a combination of MSE and L1 losses, aimed at reducing the impact of outliers while maintaining gradient stability. This combination is likely intended to balance model precision and robustness in predictions.

S2.2 GRU:

S2.2.1 Architecture-Related Hyperparameters

Number of Channels in Convolutional Layer (128): The convolutional layer's output channels are set to 128. This is because effective feature extraction is usually required before processing temporal data, and a higher number of channels helps the model capture a richer set of feature information. This is particularly important in dealing with complex turbulent heat flux data.

Hidden State Size of GRU Layer (64): The GRU (Gated Recurrent Unit) layer's hidden state size is set to 64, which determines the GRU unit's memory capacity when processing temporal data. Compared to LSTM, the GRU architecture is simpler and has fewer parameters but can still effectively capture long-term dependencies in time series.

Number of GRU Layers (3): The model includes three GRU layers. A multilayer structure helps learn more complex temporal features but also implies more parameters and a potential risk of over fitting.

Use of Bidirectional GRU (bidirectional=True): Similar to the bidirectional LSTM, bidirectional GRU can learn the dependencies of time series data both forward and backward, which is very useful for understanding the full context of the time series.

S2.2.2 Training-Related Hyperparameters

Dropout Ratios (0.15 and 0.3): Dropout regularization is applied in the model to reduce over fitting. Different levels of dropout ratios might be to increase the model's generalization capability while maintaining model complexity.

Non-linear Activation Functions Between Linear and GRU Layers (ReLU and GELU): The use of these activation functions aims to increase the model's non-linear capability, allowing it to learn more complex function mappings. GELU provides smooth gradients, aiding the optimization process, while ReLU is widely used for its computational efficiency.

S2.2.3 Loss Functions

MSELoss and SmoothL1Loss: These are used to assess the difference between the model's outputs and the target values. MSE loss is very sensitive to outliers, while Smooth L1 loss attempts to find a balance between the robustness of L1 loss and the efficiency of MSE loss. This combination is likely aimed at improving the model's accuracy and robustness in predicting turbulent heat flux data.

S2.3 Transformer:

S2.3.1 Architecture-Related Hyperparameters

Output Channels of Convolutional Layer (128): This parameter determines the number of features that the convolutional layer can capture. For simulating complex physical processes like turbulent heat flux, choosing a higher number of channels helps the model capture a richer set of feature information.

S2.3.2 Transformer Block Configuration:

Dimension (128): The dimension of the Transformer directly impacts the model's capacity to process information. Higher dimensions mean that the model can store and process more information internally, which is crucial for complex problems.

Dropout Rates (0.1, 0.25, and 0.5): Using different dropout rates in the Transformer module helps prevent over fitting while maintaining the model's ability to generalize data. Different dropout rates may be used to explore the model's performance under varying degrees of regularization.

Use of Layer Normalization (True): Layer normalization helps stabilize the training process and accelerate convergence, a common practice in Transformer models.

S2.3.3 Linear Layer Configuration:

Input and Output Dimensions (128 to 128): These linear layers are used within the model to further process and transform features. Maintaining the same dimensions helps preserve the density of information flow, aiding in capturing complex relationships.

S2.3.4 Training-Related Hyperparameters

Dropout Ratios (0.15, 0.3): Different levels of dropout ratios help the model mitigate the risk of over fitting while maintaining complexity. Selecting different rates might be based on experimental outcomes or aimed at adjusting the model's fit to the training data.

S2.3.5 Use of Loss Functions:

MSELoss and SmoothL1Loss: These two loss functions are used for different outputs, aimed at balancing sensitivity to outliers and the smoothness of predictions. Using weighted loss functions can further adjust the model's focus on different types of samples.

S3. Regarding the deep learning learning rate and stochasticity issues, this paper makes the following considerations.

The initial learning rate utilized in this study is set to 0.0005, configured at the beginning of the training loop as the lr variable. Subsequently, the paper employs a learning rate decay strategy to facilitate faster convergence and enhanced accuracy, whereby if epoch % 6 == 0 and lr >= 0.000025, the learning rate is reduced by half. This implies that every six epochs, provided the current learning rate is greater than or equal to 0.000025, the learning rate will be updated to half its present value, thus implementing learning rate decay.

Regarding the treatment of randomness, measures have indeed been implemented within the code to establish a random seed, thereby ensuring the reproducibility of experiments. Specifically, the random seed is set using the setup_seed (seed) function, affecting the randomness in PyTorch, Numpy, and CUDA. Setting a random seed guarantees that each execution of the code will consistently result in identical outcomes for random operations such as initial weight initialization and dataset splitting.

S4. In order to have a clearer understanding of the mechanism of KNN interpolating environmental driving quantities, the KNN-impute distance calculation method is given as：

This study opts to employ the K-Nearest Neighbors (K-NN) method for imputing missing data in environmental drivers, with one significant advantage of K-NN being its distance-based weighting mechanism. This allows observations closer in feature space to exert a greater influence on the imputation outcome, enabling more accurate prediction of missing values—an advantage not shared by Random Forest. The choice of 3 as the number of neighbors is grounded in the fact that a smaller number of neighbors can reduce computational complexity and enhance the efficiency of imputation while maintaining accuracy. This method strikes a balance between imputation quality and computational efficiency of the algorithm, making it both practical and efficient for handling missing environmental driver data.

Various distance calculation methods are available within the K-NN algorithm, commonly including:

Euclidean Distance: The most frequently used distance metric, calculated as the square root of the sum of the squared differences between dimensions. It is suitable for numerical data.

Manhattan Distance: Calculates the sum of the absolute differences between points in a standard coordinate system. It is applicable to grid layout path planning and scenarios where differences in each dimension are equally important.

Chebyshev Distance: The distance between two points is defined as the maximum value among their coordinate differences. It is suitable for situations where the most extreme difference needs to be considered.

The choice of distance metric in K-NN depends on the data type and application context. For environmental drivers of turbulent heat flux measured over time scales, and where the primary concern is the distance between time points, Euclidean distance is an apt choice.

In this study, by setting the weights="distance" parameter, the KNN imputation (KNNImputer) utilizes a weighted Euclidean distance formula for calculation. This means that for each missing value, the algorithm identifies the nearest "n_neighbors" (3 neighbors) and uses their values, weighting them by the inverse of their distances to the point of imputation to estimate missing values.

The weighted Euclidean distance formula is used to calculate the distance between two points, taking into account the importance or weight of each dimension. Given two points $P = (p_1, p_2, ..., p_n)$ and $Q = (q_1, q_2, ... , q_n)$, along with weights for each dimension $W = (w_1, w_2, ... , w_n)$, the weighted Euclidean distance $d_w(P,Q)$ is defined as:

$$d_w(P,Q) = \sqrt{w_1(p_1 - q_1)^2 + w_2(p_2 - q_2)^2 + \cdots + w_n(p_n - q_n)^2} \qquad \text{S1}$$

When using the weights="distance" parameter in KNN-Imputer, weights are calculated based on the inverse of the distance between points. Here, W_i represents the weight for the ith dimension. In the context of time series imputation with KNN, weighting typically refers to weighting each neighbor's contribution according to the distance, rather than applying weights directly in the distance formula. For each missing value, the imputed value is calculated based on the values of the nearest neighbors, where the contribution of each neighbor is weighted by the inverse of their distance to the missing data point, meaning shorter distances contribute more heavily. This indicates that specific weights are dynamically calculated based on the actual distances between data points, rather than being pre-specified.

S5. To be more explicit about the popularity of Transformer_CNN, the driver model was driven with basic weather elements and applied to other sites.

Below are the imputation results using Transformer_CNN at the QOMS and SETORS sites (with the year 2012 as the test set), employing basic meteorological elements. These elements include single-layer air temperature, pressure, single-layer air humidity, single-layer wind speed, single-layer wind direction, site hourly average precipitation, ground net radiation, single-layer soil temperature, and single-layer soil moisture content.

Table S1. Imputation results using the Transformer_CNN model at the QOMS and SETORS sites, employing basic meteorological elements, with the year 2012 as the test set.

Sets	QOMS						SETORS					
	H			LE			H			LE		
	RMSE	MAE	R^2	RMSE	MAE	R^2	RMSE	MAE	R^2	RMSE	MAE	R^2
Transformer	34.76	4.36	0.74	37.58	4.77	0.69	39.96	5.28	0.70	42.48	4.79	0.67
Transformer_CNN	29.34	3.44	0.83	30.25	3.93	0.78	31.66	4.83	0.80	34.22	4.61	0.79

S6. To make the Transformer_CNN model more convincing，we have now completed the iterative validation of simulating turbulent heat flux data over a decade.

The results indicate that, except for underperforming slightly in comparison to the Transformer model in 2016, the Transformer_CNN model emerged as the best model for simulating turbulent heat flux in all other years, further validating the effectiveness of Transformer_CNN as a viable tool for imputing turbulent heat flux data.

Table S2. Model performance evaluation (RMSE and MAE) for SVR, KNN, XGBoost, LSTM, GRU, Transformer and Transformer_CNN on the 2007 dataset. Bold values highlight the best performance.

Sets	H						LE					
	Training		Validation		Test		Training		Validation		Test	
	RMSE	MAE	RMSE	MAE	RMES	MAE	RMSE	MAE	RMSE	MAE	RMES	MAE

	H						LE					
SVR	20.61	2.521	24.88	3.443	27.68	4.145	26.32	3.093	21.12	3.174	19.98	3.693
KNN	22.37	3.546	24.20	3.864	30.67	4.421	18.49	2.961	19.55	2.377	20.91	3.315
XGBoost	28.64	4.644	29.85	4.841	31.28	4.929	19.85	3.044	20.77	3.087	25.13	3.944
LSTM	23.18	3.451	25.99	4.018	29.17	4.547	17.98	3.348	21.47	3.571	24.22	3.816
GRU	22.15	3.051	22.57	3.244	26.37	3.968	**17.55**	2.257	20.91	2.753	23.35	3.646
Transformer	19.53	2.848	20.07	**3.056**	23.29	3.433	18.67	2.978	19.37	3.145	19.24	3.118
Transformer_CNN	**15.67**	**2.476**	**17.57**	3.168	**20.87**	**3.355**	17.62	**2.946**	**18.39**	**2.876**	**18.95**	**2.993**

Table S3. Model performance evaluation (RMSE and MAE) for SVR, KNN, XGBoost, LSTM, GRU, Transformer and Transformer_CNN on the 2008 dataset. Bold values highlight the best performance.

	H						LE					
Sets	Training		Validation		Test		Training		Validation		Test	
	RMSE	MAE	RMSE	MAE	RMES	MAE	RMSE	MAE	RMSE	MAE	RMES	MAE
SVM	20.46	2.942	21.69	3.056	27.89	3.662	26.88	3.461	29.15	3.841	30.75	4.025
KNN	21.44	3.018	22.87	3.246	26.35	3.428	23.18	3.143	24.85	3.386	28.54	3.757
XGBoost	21.38	3.007	24.91	3.582	27.16	3.659	23.55	3.277	25.55	3.568	29.33	3.954
LSTM	22.63	3.144	25.75	3.458	28.33	3.881	20.86	3.155	23.91	3.347	25.88	3.552
GRU	20.44	2.988	23.24	3.528	26.99	3.699	19.71	3.048	23.45	3.257	24.13	3.848
Transformer	15.77	**2.544**	18.62	2.876	23.47	3.258	19.15	2.883	**19.10**	**3.079**	19.57	3.130
Transformer_CNN	**14.84**	2.668	**17.88**	**2.759**	**21.83**	**2.954**	**18.18**	**2.784**	20.25	3.092	**18.47**	**2.836**

Table S4. Model performance evaluation (RMSE and MAE) for SVR, KNN, XGBoost, LSTM, GRU, Transformer and Transformer_CNN on the 2009 dataset. Bold values highlight the best performance.

	H						LE					
Sets	Training		Validation		Test		Training		Validation		Test	
	RMSE	MAE	RMSE	MAE	RMES	MAE	RMSE	MAE	RMSE	MAE	RMES	MAE
SVM	19.71	3.456	20.29	3.60	27.49	3.955	24.89	3.062	25.76	3.284	29.24	3.856
KNN	21.75	3.701	26.15	3.641	32.12	4.112	25.34	3.277	28.62	3.577	30.65	3.888
XGBoost	26.39	3.666	25.22	4.419	29.18	4.906	23.51	2.983	24.57	3.147	28.36	3.743
LSTM	25.77	3.451	25.43	4.183	29.10	4.752	24.19	3.248	26.47	3.335	29.09	3.842

GRU	22.16	3.091	22.62	4.017	28.24	4.218	22.64	2.889	24.81	3.053	27.55	3.257
Transformer	**17.25**	**2.973**	**18.22**	3.279	24.87	3.495	**19.80**	**2.459**	23.51	3.018	24.87	3.266
Transformer_CNN	18.77	3.025	19.24	**3.246**	**23.16**	**3.357**	20.54	2.687	**22.76**	**2.928**	**21.38**	**3.006**

Table S5. Model performance evaluation (RMSE and MAE) for SVR, KNN, XGBoost, LSTM, GRU, Transformer and Transformer_CNN on the 2010 dataset. Bold values highlight the best performance.

	H						LE					
Sets	Training		Validation		Test		Training		Validation		Test	
	RMSE	MAE	RMSE	MAE	RMES	MAE	RMSE	MAE	RMSE	MAE	RMES	MAE
SVM	22.68	2.951	23.95	3.681	25.44	3.870	24.18	4.196	25.20	4.174	26.23	4.451
KNN	18.29	2.981	20.01	3.665	21.03	3.882	23.23	3.941	24.35	4.267	26.11	4.440
XGBoost	16.33	3.227	19.43	3.970	21.51	3.940	19.35	2.946	23.14	3.781	25.49	4.002
LSTM	19.84	2.904	20.86	3.696	23.31	3.373	22.94	3.761	23.71	3.928	27.09	4.927
GRU	18.26	3.621	19.81	3.340	22.14	3.400	18.26	2.843	20.95	3.039	24.01	3.588
Transformer	**17.37**	**3.047**	19.37	**3.641**	20.26	3.944	19.16	3.038	21.75	3.254	22.64	3.337
Transformer_CNN	17.56	3.108	**18.84**	3.884	**19.38**	**3.250**	**18.42**	**2.925**	**19.59**	**3.004**	**20.68**	**3.209**

Table S6. Model performance evaluation (RMSE and MAE) for SVR, KNN, XGBoost, LSTM, GRU, Transformer and Transformer_CNN on the 2011 dataset. Bold values highlight the best performance.

	H						LE					
Sets	Training		Validation		Test		Training		Validation		Test	
	RMSE	MAE	RMSE	MAE	RMES	MAE	RMSE	MAE	RMSE	MAE	RMES	MAE
SVM	19.39	3.764	20.366	3.900	26.14	4.142	21.21	3.441	21.96	3.682	22.45	3.870
KNN	21.48	3.634	25.95	3.788	31.49	4.189	20.28	3.367	23.87	4.061	25.03	3.880
XGBoost	26.30	4.089	25.83	3.912	29.49	4.498	16.33	3.227	19.43	3.870	17.35	3.594
LSTM	25.96	3.102	24.68	4.266	29.33	4.553	19.84	2.900	20.87	3.196	21.31	3.373
GRU	22.60	3.593	21.77	3.447	27.24	4.513	18.27	2.862	22.43	4.057	20.47	4.115
Transformer	17.53	2.887	19.07	3.817	24.71	4.016	18.75	3.620	19.81	3.340	20.68	3.644
Transformer_CNN	**17.13**	**2.519**	**18.74**	**3.625**	**24.66**	**3.778**	**18.50**	**3.312**	**19.52**	**2.921**	**19.35**	**3.267**

Table S7. Model performance evaluation (RMSE and MAE) for SVR, KNN, XGBoost, LSTM, GRU, Transformer and Transformer_CNN on the 2013 dataset. Bold values highlight the best performance.

Sets	H						LE					
	Training		Validation		Test		Training		Validation		Test	
	RMSE	MAE	RMSE	MAE	RMES	MAE	RMSE	MAE	RMSE	MAE	RMES	MAE
SVM	18.27	2.736	20.15	3.043	26.89	3.746	23.67	2.754	24.76	3.062	19.79	3.124
KNN	19.61	3.034	22.14	3.176	25.73	3.519	24.16	2.972	27.43	5.265	29.27	5.477
XGBoost	21.23	3.248	23.86	3.394	25.16	4.061	18.29	3.014	19.27	3.091	19.04	2.943
LSTM	23.34	3.432	23.97	3.609	27.38	4.483	23.14	3.207	24.12	3.346	26.47	4.573
GRU	22.18	3.374	22.94	3.457	24.93	3.685	19.25	3.216	19.99	3.265	21.84	3.597
Transformer	17.43	2.831	18.47	2.929	20.48	3.462	**16.08**	**2.462**	18.24	2.857	17.49	2.963
Transformer_CNN	**14.25**	**2.472**	**16.88**	**2.746**	**18.43**	**3.154**	17.76	2.564	**18.09**	**2.681**	**17.35**	**2.869**

Table S8. Model performance evaluation (RMSE and MAE) for SVR, KNN, XGBoost, LSTM, GRU, Transformer and Transformer_CNN on the 2014 dataset. Bold values highlight the best performance.

Sets	H						LE					
	Training		Validation		Test		Training		Validation		Test	
	RMSE	MAE	RMSE	MAE	RMES	MAE	RMSE	MAE	RMSE	MAE	RMES	MAE
SVM	19.84	3.519	21.65	3.735	25.36	4.357	23.17	3.493	24.64	3.874	28.14	4.637
KNN	22.38	4.357	24.66	4.561	31.48	6.05	24.38	3.627	26.07	4.036	29.21	5.143
XGBoost	23.32	4.534	25.49	4.851	28.28	5.568	20.64	3.183	21.28	3.436	24.24	4.324
LSTM	23.83	4.647	25.26	4.435	27.73	4.969	24.37	3.846	25.72	4.041	26.37	4.235
GRU	23.08	4.342	23.75	4.624	26.87	4.867	20.34	3.068	21.26	3.264	25.84	4.027
Transformer	17.48	2.531	17.68	2.634	22.43	3.267	16.24	2.637	18.68	2.943	21.36	3.489
Transformer_CNN	**17.17**	**2.343**	**17.50**	**2.353**	**22.41**	**2.999**	**15.96**	**2.426**	**18.53**	**2.554**	**21.02**	**3.224**

Table S9. Model performance evaluation (RMSE and MAE) for SVR, KNN, XGBoost, LSTM, GRU, Transformer and Transformer_CNN on the 2015 dataset. Bold values highlight the best performance.

Sets	H						LE					
	Training		Validation		Test		Training		Validation		Test	
	RMSE	MAE	RMSE	MAE	RMES	MAE	RMSE	MAE	RMSE	MAE	RMES	MAE
SVM	19.32	2.784	19.98	2.949	26.61	3.155	25.09	4.215	26.24	4.342	28.69	4.524
KNN	23.16	3.075	26.78	4.211	32.52	4.517	20.56	3.192	22.75	3.487	26.71	4.365
XGBoost	24.38	3.227	25.07	3.439	28.14	3.958	19.27	3.062	23.54	3.679	24.75	3.934
LSTM	24.26	3.106	25.17	3.548	29.57	4.352	24.43	2.859	24.57	4.254	28.76	3.942
GRU	22.10	3.081	23.54	3.349	27.53	4.254	21.59	2.818	21.76	3.857	25.49	3.751
Transformer	17.75	**2.719**	19.34	**2.964**	25.48	3.867	19.34	**2.963**	21.27	3.685	22.57	3.841
Transformer_CNN	**16.34**	2.951	**18.61**	3.735	**24.43**	**3.741**	**18.68**	3.461	**19.75**	**3.424**	**20.56**	**3.689**

Table S10. Model performance evaluation (RMSE and MAE) for SVR, KNN, XGBoost, LSTM, GRU, Transformer and Transformer_CNN on the 2016 dataset. Bold values highlight the best performance.

Sets	H						LE					
	Training		Validation		Test		Training		Validation		Test	
	RMSE	MAE	RMSE	MAE	RMES	MAE	RMSE	MAE	RMSE	MAE	RMES	MAE
SVM	20.79	2.853	20.35	2.724	25.22	3.682	24.47	2.991	21.01	3.156	19.49	3.344
KNN	19.83	2.867	26.43	3.876	30.99	3.957	23.41	2.816	25.16	3.864	24.16	3.761
XGBoost	25.79	3.894	27.07	3.989	30.34	4.488	27.26	3.416	26.47	3.165	24.39	3.678
LSTM	23.24	3.381	24.49	3.514	29.45	4.085	23.91	2.924	24.46	3.644	25.62	3.927
GRU	22.37	3.513	23.53	3.155	29.61	3.796	19.73	2.916	21.49	3.661	21.18	3.498
Transformer	15.36	2.486	**16.54**	**2.665**	23.26	2.966	**17.69**	**2.851**	**18.68**	**2.973**	**19.06**	**2.982**
Transformer_CNN	**15.32**	**2.458**	17.19	2.749	**22.60**	**2.894**	18.13	3.221	19.71	3.208	20.12	3.092