FABLE: A Localized, Targeted Adversarial Attack on Weather Forecasting Models

Yue Deng
Michigan State University
dengyue1@msu.edu

Asadullah Hill Galib
TSMC Technology Inc.
asadgalib19@gmail.com

Xin Lan
Michigan State University
lanxin1@msu.edu

Pang-Ning Tan
Michigan State University
ptan@msu.edu

Lifeng Luo
Michigan State University
lluo@msu.edu

Abstract

Deep learning-based weather forecasting models have recently demonstrated significant performance improvements over gold-standard physics-based simulation tools. However, these models are vulnerable to adversarial attacks, which raises concerns about their trustworthiness. In this paper, we first investigate the feasibility of applying existing adversarial attack methods to weather forecasting models. We argue that a successful attack should (1) not modify significantly its original inputs, (2) be faithful, i.e., achieve the desired forecast at targeted locations with minimal changes to non-targeted locations, and (3) be geospatio-temporally realistic. However, balancing these criteria is a challenge as existing methods are not designed to preserve the geospatio-temporal dependencies of the original samples. To address this challenge, we propose a novel framework called *FABLE* (Forecast Alteration By Localized targeted advErsarial attack), which employs a 3D discrete wavelet decomposition to extract the varying components of the geospatio-temporal data. By regulating the magnitude of adversarial perturbations across different components, FABLE can generate adversarial inputs that maintain geospatio-temporal coherence while remaining faithful and closely aligned with the original inputs. Experimental results on multiple real-world datasets demonstrate the effectiveness of our framework over baseline methods across various metrics.

1 Introduction

Weather forecasting plays a crucial role in a wide range of human activities, influencing decision-making in numerous sectors such as agriculture, energy, insurance, transportation, and public safety. With the growing impact of extreme weather events, the demand for accurate weather forecasting continues to increase, with more industries recognizing the value of precise and timely weather information to mitigate risks and capitalize on opportunities. In recent years, deep learning-based weather forecasting models [14, 22, 21, 2] have achieved significant improvements in prediction accuracy compared to traditional physics-based approaches. However, these models are susceptible to adversarial attacks [15, 7], in which malicious actors can manipulate the models by introducing subtle alterations to the input data, leading to incorrect predictions. For weather forecasting, such attacks could result in misguided management decisions, inefficient resource allocation, and a failure to adequately prepare for extreme weather events.

In this paper, we investigate the problem of designing *localized, targeted adversarial attacks* on weather forecasting models. A localized targeted attack involves malicious manipulation of the input data so that the model's outputs for a pre-defined set of locations at a specific future time period

Preprint. Under review.

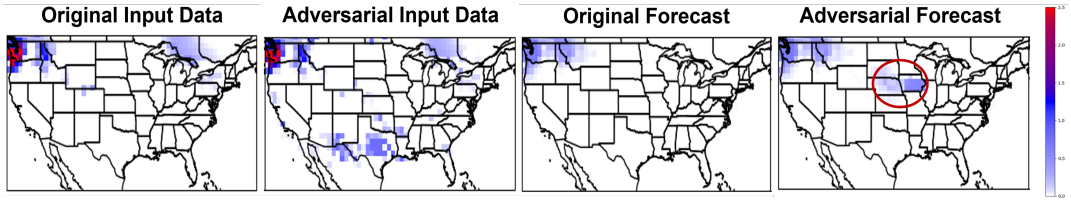

Figure 1: The left two panels show the original input data and its adversarial sample produced by the *TAAOWPF* adversarial attack method [16] for the *NLDAS* precipitation dataset. The right two panels show their corresponding original and adversarial forecasts generated by the *CLCRN* weather forecasting model [22]. The red circle indicates the targeted locations for forecast manipulation.

closely match the values desired by the attacker. For instance, consider an attacker who wants to influence agricultural markets by creating a false forecast of heavy rainfall in a major farming region. They could manipulate the input data fed into the deep learning model, causing it to predict significant rainfall when none is actually expected. Figure 1 shows an example of a tampered precipitation forecast for Iowa, generated by making alterations to the original input data using the *TAAOWPF* adversarial attack method [16]. This false forecast could lead farmers to delay planting, alter irrigation schedules, or take unnecessary corrective actions based on incorrect predictions. The resulting market disruptions could benefit the attacker financially, especially if they have investments that would gain from such changes.

We argue that the generated adversarial inputs for weather forecasting models should satisfy three key criteria: (1) *faithfulness*, ensuring that the adversarial inputs yield the intended forecasts at the target locations and specific time periods, with minimal changes to non-targeted locations, (2) *closeness*, making the adversarial inputs harder to detect by closely resembling the original inputs, and (3) *geospatio-temporal realisticness*, preserving the dependencies within the original samples to enhance stealthiness. However, managing the trade-offs among these criteria can be challenging for existing adversarial attack methods [15, 16, 24]. For instance, as shown in Figure 1, the adversarial sample introduces noticeable perturbations to other, non-targeted locations, making the attack more easily perceivable. The added perturbations would also alter the spatial autocorrelations inherent in the original data.

To balance the competing criteria in adversarial input generation, we propose a new framework called *FABLE* (Forecast Alteration By Localized targeted advErsarial attack). Unlike conventional methods that directly perturb the original input, FABLE applies adversarial perturbation on different components of the geospatio-temporal input, which are obtained using a 3D discrete wavelet decomposition. We show that, by applying larger magnitudes of perturbations on the high-frequency components, FABLE was able to achieve better geospatio-temporal realisticness and closeness while maintaining comparable faithfulness as other baseline approaches.

2 Related works

Weather forecasting has long been an active area of research due to its critical impact on our environment and society. Traditional models [10, 28, 17] employ numerical simulations based on physical equations to predict future weather conditions. However, accurately modeling the chaotic nature of Earth's meteorological system remains a challenging problem. Towards this end, deep learning models have recently emerged as effective tools for weather forecasting due to their ability to capture complex geospatio-temporal patterns in data. These models, utilizing CNN [2, 14], GCN [22, 21, 26], Vision Transformer [3], and Swin Transformer [4, 5], have improved both single-step [13] and multi-step [22, 21] forecasts using univariate [22] and multivariate [21] weather data.

As deep learning models become more prevalent, the risk of adversarial attacks on these systems has grown. Adversarial attack techniques, such as FGSM [15], PGD [24], and MIM [12], work by subtly perturbing the input data to mislead models into making incorrect predictions. Specifically, given a forecast model g, the adversarial attack would modify an *input sample* $\mathbf{X}$ by adding a perturbation δ_X, producing the *adversarial sample* $\mathbf{X} + \delta_{\mathbf{X}}$. The goal of the attack is to ensure that the *adversarial forecast* $g(\mathbf{X} + \delta_{\mathbf{X}})$ differs substantially from the *original forecast* $g(\mathbf{X})$.

Adversarial attacks can be categorized in terms of their objectives as *untargeted*, *semi-targeted*, or *targeted* attacks [16]. Untargeted attacks aim to generate predictions that deviate significantly from the original forecast, i.e., $\delta_{\mathbf{X}} = \arg\max_{\delta} \mathcal{L}\left[g(\mathbf{X}+\delta), g(\mathbf{X})\right]$, where $\mathcal{L}$ is the loss function. Semi-targeted attacks constrain the predictions within attacker-specified boundaries, while targeted attacks steer the predictions toward a specific *adversarial target* $\hat{\mathbf{Y}}'$, i.e., $\delta_{\mathbf{X}} = \arg\min_{\delta} \mathcal{L}\left[g(\mathbf{X}+\delta), \hat{\mathbf{Y}}'\right]$. The adversarial attacks can also be categorized based on attacker's level of access to the model [23]—*black-box* (access only to inputs and outputs), *grey-box* (partial knowledge of architecture or training techniques), or *white-box* (full knowledge of architecture, pre-trained parameters, and data).

Adversarial attacks have been successfully applied to various domains. In computer vision, these methods subtly alter input images or videos to produce misclassification or generation errors [9, 29, 30]. For spatial-temporal data, adversarial attacks have been studied in renewable energy forecasting and traffic prediction. Previous studies of energy forecasting [16, 18] primarily applied standard attack methods such as FGSM [15] or PGD [24] to perturb temporal weather inputs, while ignoring their spatial information [16, 27, 18]. In traffic forecasting, existing methods mainly focus on untargeted attacks that disrupt overall traffic flow predictions, aiming to create congestion [32, 23]. However, localized targeted attacks, which are designed to manipulate multi-step predictions at specific locations while minimizing the impact elsewhere, remain largely unexplored.

3 Problem Statement

Consider a geospatio-temporal dataset $\mathcal{D} = (\mathbf{Z}_1\mathbf{Z}_2\cdots\mathbf{Z}_t\cdots)$, where $\mathbf{Z}_t \in \mathbb{R}^{r\times c}$ corresponds to the weather observations for $r \times c$ locations at time step t. We further denote Z_{tij} as the value of the weather variable at time step t at a given location, whose latitude and longitude are indexed by (i, j), where $i \in \{1, 2, \cdots, r\}$ and $j \in \{1, 2, \cdots, c\}$. At each time step t_0, we construct a pair of tensors: (1) $\mathbf{X}(t_0) \in \mathbb{R}^{(\alpha+1)\times r\times c}$, a predictor time window of length $\alpha + 1$ containing observations for all locations at t_0 and its preceding α time steps; and (2) $\mathbf{Y}(t_0) \in \mathbb{R}^{\beta\times r\times c}$, a forecast window of length β containing observations for all locations over the subsequent β time steps. Thus, $X_{\tau ij}(t_0) = Z_{t_0-\alpha-1+\tau,ij}$, where $\tau \in \{1, 2, \cdots, \alpha + 1\}$, while $Y_{\tau ij}(t_0) = Z_{t_0+\tau,ij}$, where $\tau \in \{1, 2, \cdots, \beta\}$. For notational convenience, we abbreviate $\mathbf{X}(t_0)$ as $\mathbf{X}$ and $\mathbf{Y}(t_0)$ as $\mathbf{Y}$.

Given $\mathbf{X} \in \mathbb{R}^{(\alpha+1)\times r\times c}$ as the predictor, a weather forecasting model g outputs a multi-step prediction $\hat{\mathbf{Y}} \in \mathbb{R}^{\beta\times r\times c}$ as follows: $\hat{\mathbf{Y}} = g(\mathbf{X})$. Let $S_{\hat{\mathbf{Y}}} = \left\{(\tau, i, j)|t \in \Gamma_{\hat{\mathbf{Y}}}; (i, j) \in \Omega\right\}$ be the *in-target geospatio-temporal domain* for a localized adversarial attack on $\hat{\mathbf{Y}}$, where $\Gamma_{\hat{\mathbf{Y}}} = \{1, 2, ..., \beta\}$ and $\Omega = \{1, \ldots, r\} \times \{1, \ldots, c\}$. The localized, adversarial target is defined as $\hat{\mathbf{Y}}' = \hat{\mathbf{Y}} + \delta_{\hat{\mathbf{Y}}}$, where $\delta_{\hat{Y}_{\tau ij}} \neq 0$ if and only if $(\tau, i, j) \in S_{\hat{\mathbf{Y}}}$ and zero elsewhere.

Our goal is to learn an adversarial predictor $\mathbf{X}' \in \mathbb{R}^{(\alpha+1)\times r\times c}$ that alters the original forecast $\hat{\mathbf{Y}} = g(\mathbf{X})$ to a new adversarial forecast $g(\mathbf{X}') \in \mathbb{R}^{\beta\times r\times c}$ that meets the following criteria: (1) **Closeness**: The adversarial predictor $\mathbf{X}'$ is said to be ϵ-close to its original predictor $\mathbf{X}$ if $\|\delta_{\mathbf{X}}\| = \|\mathbf{X}' - \mathbf{X}\| \leq \epsilon$, where $\epsilon > 0$. (2) **Faithfulness**: Let $\hat{\mathbf{Y}}'_{\mathbf{in}} = \{\hat{Y}'_{\tau ij}|(\tau, i, j) \in S_{\hat{\mathbf{Y}}}\}$ and $g(\mathbf{X}')_{\mathbf{in}} = \{g(\mathbf{X}')_{\tau ij}|(\tau, i, j) \in S_{\hat{\mathbf{Y}}}\}$. The adversarial forecast $g(\mathbf{X}')$ is said to be μ**-in-target faithful** if $\mathcal{L}\big(g(\mathbf{X}')_{\mathbf{in}}, \hat{\mathbf{Y}}'_{\mathbf{in}}\big) \leq \mu$, where $\mathcal{L}(\cdot)$ is a loss function. Similarly, let $\hat{\mathbf{Y}}'_{\mathbf{out}} = \{\hat{Y}'_{\tau ij}|(\tau, i, j) \notin S_{\hat{\mathbf{Y}}}\}$ and $g(\mathbf{X}')_{\mathbf{out}} = \{g(\mathbf{X}')_{\tau ij}|(\tau, i, j) \notin S_{\hat{\mathbf{Y}}}$. The adversarial forecast $g(\mathbf{X}')$ is said to be ν**-out-target faithful** if $\mathcal{L}\big(g(\mathbf{X}')_{\mathbf{out}}, \hat{\mathbf{Y}}'_{\mathbf{out}}\big) \leq \nu$. (3) **Realisticness:** The adversarial predictor $\mathbf{X}'$ is σ-spatially realistic if $R_S(\mathbf{X}', \mathbf{X}) \leq \sigma$, where $R_S(\cdot)$ measures the consistency between the spatial autocorrelation in $\mathbf{X}'$ and $\mathbf{X}$. Analogously, $\mathbf{X}'$ is κ-temporally realistic if $R_T(\mathbf{X}', \mathbf{X}) \leq \kappa$, where $R_T(\cdot)$ measures the consistency between the temporal autocorrelation in $\mathbf{X}'$ and $\mathbf{X}$.

In this study, we consider a white-box threat model, wherein the adversary is assumed to have full access to the forecast model g. This assumption is justified, as numerous deep learning-based weather forecasting models——such as those referenced in Section 2——have publicly released their source code and pre-trained checkpoints. The white-box attacks also facilitate a principled assessment of worst-case robustness by enabling the derivation of theoretical lower bounds [1, 6], and have been employed in recent works [8]. Extending our study to a black-box scenario is deferred to future work.

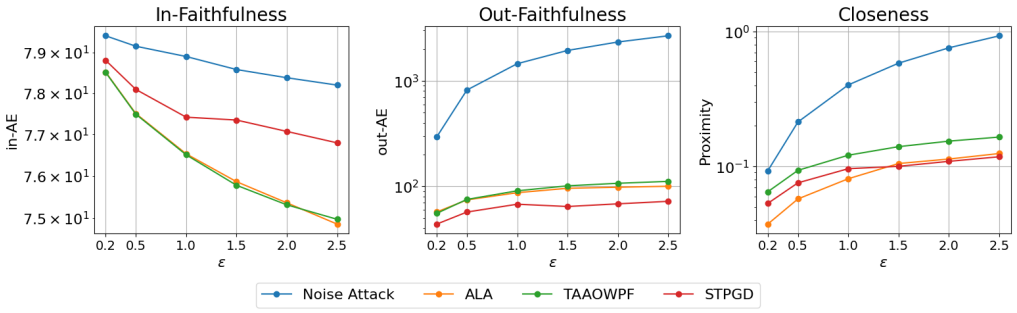

Figure 2: Performance comparison of existing adversarial attack methods on *NLDAS* precipitation dataset in terms of faithfulness and closeness. Lower metric values indicate better performance.

4 Feasibility of Adversarial Attacks on Weather Forecasting Models

In this section, we empirically investigate the feasibility of applying existing adversarial attack methods to weather forecasting models. Specifically, we consider 4 representative methods: (1) *Noise Attack*, which adds random perturbation to the input predictor, (2) *ALA* [27], which uses the Adam optimizer to learn the perturbation, (3) *TAAOWPF* [16], which leverages the projected gradient descent (PGD) method, and (4) *STPGD* [23], which extends the PGD-based method by selectively perturbing a subset of locations whose gradients have the most impact on the loss. As none of these methods are originally designed for localized, targeted adversarial attacks, ***Appendix F*** describes how they can be adapted to our problem setting. We evaluated the faithfulness and closeness of their generated adversarial samples using the *in-AE*, *out-AE*, and *closeness* measures as defined in Section 6.1. Furthermore, to ensure the stealthiness of the attack, the adversarial sample $\mathbf{X}'$ should preserve the spatial and temporal autocorrelations inherent in the original predictor $\mathbf{X}$. Towards this end, we employ the following metrics to measure their spatial (R_S) and temporal (R_T) realisticness:

$$R_S(\mathbf{X}') = \frac{1}{\alpha+1}\sum_{\tau=1}^{\alpha+1}|I(\mathbf{X}'_\tau) - I(\mathbf{X}_\tau)|; \quad R_T(\mathbf{X}') = \frac{1}{r\times c}\sum_{i=1}^{r}\sum_{j=1}^{c}\frac{1}{\alpha+1}\sum_{l=1}^{\alpha+1}|\rho_l(\mathbf{X}'_{ij}) - \rho_l(\mathbf{X}_{ij})|, \quad (1)$$

where

$$I(X_\tau) = \frac{r^2\times c^2}{W}\frac{\sum_{(i,j)(k,l)} w_{ij,kl}(X_{\tau ij} - \overline{\mathbf{X}}_\tau)(X_{\tau kl} - \overline{\mathbf{X}}_\tau)}{\sum_{(i,j)}(X_{\tau ij} - \overline{\mathbf{X}}_\tau)^2}$$

is the Moran's I metric, which quantifies the spatial autocorrelation of a map $\mathbf{X}_\tau \in \mathbb{R}^{r\times c}$, and

$$\rho_l(X_{ij}) = \frac{\sum_{\tau=1}^{\alpha+1-l}(X_{\tau ij} - \overline{\mathbf{X}}_{ij})(X_{\tau+l,ij} - \overline{\mathbf{X}}_{ij})}{\sum_{\tau=1}^{T}(X_{\tau ij} - \overline{\mathbf{X}}_{ij})^2}$$

is the temporal autocorrelation at lag l for the time series at location (i,j). Furthermore, $\overline{\mathbf{X}}_\tau \in \mathbb{R}$ is the average value of the spatial map $\mathbf{X}_\tau$ and $W = \sum_{(i,j),(k,l)} w_{ij,kl}$ denotes the total sum of an $(r\times c)\times(r\times c)$ weight matrix representing the spatial relationship between locations (i,j) and (k,l). The matrix encodes the degree of influence between two locations, capturing their spatial proximity, with zeros along the diagonal. Specifically, the weights are computed as follows: $\omega_{ij,kl} = 1/d_{ij,kl}$, where $d_{ij,kl}$ is the geographical distance between the two locations. For temporal autocorrelation, $\mathbf{X}_{ij} \in \mathbb{R}^{\alpha+1}$ is the time series of length $\alpha+1$ at location (i,j) while $\overline{\mathbf{X}}_{ij} \in \mathbb{R}$ denotes its average value. Since R_S and R_T measure the difference in autocorrelation between the original and adversarial input, a smaller value of these measures would imply higher geospatio-temporal realisticness.

We first compare the faithfulness and closeness of these methods when applied to the *NLDAS* precipitation dataset. Details on the dataset and adversarial target construction are provided in ***Appendix G***. Since the output of the various adversarial attack methods depends on the magnitude of perturbation ϵ, for fair comparison, we compare their performances at varying levels of ϵ. The results are shown in Figure 2. As expected, the random noise approach struggles to produce the desired adversarial target forecast $\hat{\mathbf{Y}}'$, unlike learning-based methods. Furthermore, as ϵ increases, allowing for larger perturbations in $\mathbf{X}$, observe that the learning-based methods, such as *ALA*, *TAAOWPF* and *STPGD*, achieve better in-target faithfulness, as evidenced by their smaller values of *in-MAE*. However, this comes at the expense of poorer out-target faithfulness, i.e., larger *out-MAE*, and

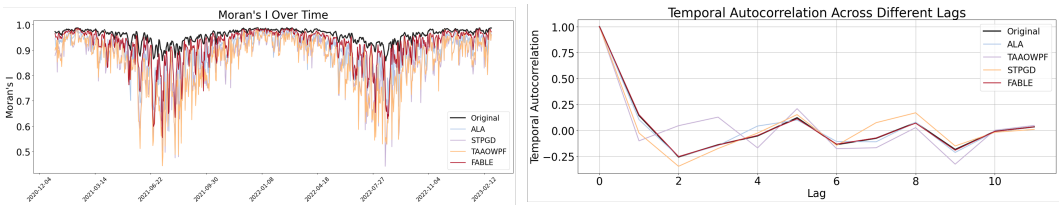

Figure 3: Impact of adversarial attack on spatial and temporal autocorrelations for *NLDAS* temperature dataset. On the left panel, the x-axis represents individual test samples on different days, while the y-axis represents Moran's I values for the original predictor $\mathbf{X}$ as well as the adversarial samples $\mathbf{X}'$ generated by different attack methods. On the right panel, the x-axis represents different lags of temporal autocorrelation while the y-axis represents temporal autocorrelation values of $\mathbf{X}$ and $\mathbf{X}'$.

worse closeness, i.e., higher discrepancy between the original and adversarial predictor. This result underscores the difficulty of balancing faithfulness and closeness when applying existing adversarial attack methods to weather forecasting models. For instance, *ALA* appears to produce the best in-target faithfulness, but its closeness is considerably worse than other learning-based methods. See ***Appendix A*** for a comparative example of the adversarial samples generated by the different methods.

Next, we evaluate the spatial and temporal realisticness of the different adversarial attack methods on the *NLDAS* temperature dataset (see ***Appendix G***). We use this instead of the *NLDAS* precipitation dataset since the latter inherently has limited spatial and temporal autocorrelations. Figure 3 shows the Moran's I and temporal autocorrelation of adversarial predictors $\mathbf{X}'$ generated by different methods in comparison to the original $\mathbf{X}$. Observe that there is a significant difference in spatial and temporal autocorrelations between the original input and the adversarial samples generated by *ALA*, *TAAOWPF*, and *STPGD*. This demonstrates the limitations of existing methods in preserving the spatial and temporal coherence of the original data. The figure also shows that our proposed method (***FABLE***), to be discussed in the next section, follows the spatial and temporal autocorrelations of the original predictor more closely than existing adversarial attack methods.

5 Proposed Framework: FABLE

To address the limitations of existing methods and improve the geospatio-temporal realisticness of the generated adversarial predictor $\mathbf{X}'$, we propose a novel adversarial attack framework, ***FABLE*** (*Forecast Alteration By Localized targeted advErsarial attack*). The overall architecture is illustrated in Figure 4. The original predictor $\mathbf{X}$ is initially decomposed into distinct subspaces using wavelet analysis, each capturing specific frequency bands corresponding to different spatial and temporal scales. Unlike conventional adversarial attack methods that directly perturb the original $\mathbf{X}$, ***FABLE*** strategically adjusts the perturbation magnitude across these decomposed frequency components. By leveraging this wavelet-based decomposition, ***FABLE*** achieves control over the trade-offs among faithfulness, closeness, and geospatio-temporal realisticness in the generated adversarial predictor.

To decompose the original predictor $\mathbf{X}$ into its subspace representation, we employ a level-one, 3D Haar wavelet decomposition[1]. Conceptually, this decomposition maps $\mathbf{X}$ into sets of high-frequency and low-frequency components along the temporal, longitudinal (column), and latitudinal (row) dimensions. Technically, it involves sequentially applying pairs of low-pass and high-pass filters across each dimension of $\mathbf{X} \in \mathbb{R}^{(\alpha+1)\times r\times c}$. The low-pass filter, defined as $h_L = [\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}]$, extracts smooth, large-scale patterns corresponding to the *low-frequency* **(L)** components of $\mathbf{X}$. Conversely, the high-pass filter, defined as $h_H = [\frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}}]$, captures localized, fine-grained variations that represent its *high-frequency* **(H)** components of $\mathbf{X}$. In the 3D case, these filters operate along the three dimensions of $\mathbf{X}$. Let $\mathbf{f} = [f_1, f_2, f_3]$ denote the combination of filters along the temporal,

[1]Compared to other wavelet bases, the Haar wavelet is relatively more intuitive and computationally efficient. Its simplicity facilitates manual implementation, enabling effective gradient propagation during backpropagation when optimizing the adversarial predictor $\mathbf{X}'$. In contrast, more complex wavelets often require API-based implementations that rely on gradient estimation, which may introduce inaccuracies and inefficiencies in adversarial sample generation.

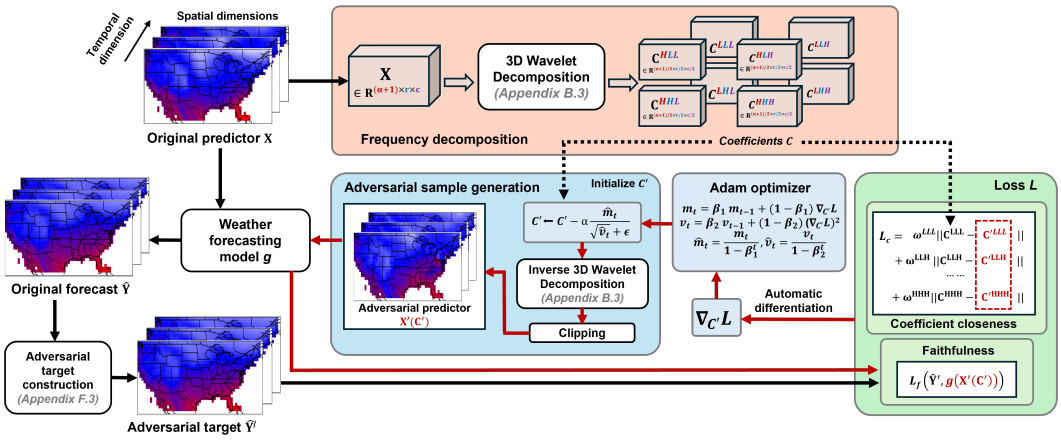

Figure 4: Framework of ***FABLE***. The original forecast $\hat{\mathbf{Y}}$ is produced by applying a weather forecasting model g to the original input $\mathbf{X}$. Let $\hat{\mathbf{Y}}'$ be the adversarial target for $\hat{\mathbf{Y}}$. To generate an adversarial sample, $\mathbf{X}$ is first decomposed into its 3D Haar wavelet coefficients $\mathbf{C}$. The coefficients are iteratively updated to minimize the total loss to obtain $\mathbf{C}'$. The adversarial sample $\mathbf{X}'$ is obtained from $\mathbf{C}'$ using inverse wavelet decomposition, which is passed to g to obtain its forecast, $g(\hat{\mathbf{X}}')$.

longitudinal (column), and latitudinal (row) dimensions of $\mathbf{X}$, respectively, where each $f_d \in \{L, H\}$ specifies whether a low- or high-frequency filter is applied along the d-th dimension.

Applying these filters to $\mathbf{X}$ yields a total of eight sub-bands: one low-frequency component (LLL), six mixed-frequency components (LLH, LHL, LHH, HLL, HLH, HHL), and one high-frequency component (HHH). Let $\mathbf{C}^{\mathbf{f}} \in \mathbb{R}^{\frac{\alpha+1}{2} \times \frac{r}{2} \times \frac{c}{2}}$ be the wavelet coefficients associated with the sub-band $\mathbf{f}$. The original predictor $\mathbf{X} \in \mathbb{R}^{(\alpha+1)\times r \times c}$ can therefore be decomposed into the following tensors of wavelet coefficients: $\mathbf{C}^{LLL}$, $\mathbf{C}^{LLH}$, $\mathbf{C}^{LHL}$, $\mathbf{C}^{LHH}$, $\mathbf{C}^{HLL}$, $\mathbf{C}^{HLH}$, $\mathbf{C}^{HHL}$, and $\mathbf{C}^{HHH}$ as follows:

$$C^{\mathbf{f}}_{k_1,k_2,k_3} = \sum_{n_1=0}^{1}\sum_{n_2=0}^{1}\sum_{n_3=0}^{1}\left[\frac{1}{\sqrt{2}}(-1)^{\xi(f_1)n_1}\right]\left[\frac{1}{\sqrt{2}}(-1)^{\xi(f_2)n_2}\right]\left[\frac{1}{\sqrt{2}}(-1)^{\xi(f_3)n_3}\right]\mathbf{X}_{2k_1-n_1,2k_2-n_2,2k_3-n_3}$$

$$= \frac{1}{\sqrt{8}}\sum_{n_1=0}^{1}\sum_{n_2=0}^{1}\sum_{n_3=0}^{1}(-1)^{\xi(f_1)n_1+\xi(f_2)n_2+\xi(f_3)n_3}\cdot\mathbf{X}_{2k_1-n_1,2k_2-n_2,2k_3-n_3}, \tag{2}$$

where $k_1 \in \left\{1, 2, \ldots, \frac{\alpha+1}{2}\right\}$, $k_2 \in \left\{1, 2, \ldots, \frac{r}{2}\right\}$, $k_3 \in \left\{1, 2, \ldots, \frac{c}{2}\right\}$[2] denotes the translation indices along the temporal, longitudinal (row), and latitudinal (column) dimensions of $\mathbf{X}$, respectively, while

$$\xi(f_d) = \begin{cases} 0, & \text{if } f_d = L; \\ 1, & \text{if } f_d = H, \end{cases}$$

and $n_d \in \{0, 1\}$ is the discrete index of the filter h_{f_d}. Based on the decomposed wavelet coefficients $\mathbf{C}^{\mathbf{f}} \in \mathbb{R}^{\frac{\alpha+1}{2} \times \frac{r}{2} \times \frac{c}{2}}$ and their corresponding scaling and wavelet functions (see ***Appendix B***), the original predictor $\mathbf{X} \in \mathbb{R}^{(\alpha+1)\times r \times c}$ can be reconstructed as follows:

$$\mathbf{X}_{2k_1-n_1,2k_2-n_2,2k_3-n_3} = \frac{1}{\sqrt{8}}\sum_{f_1,f_2,f_3\in\{L,H\}}(-1)^{\xi(f_1)n_1+\xi(f_2)n_2+\xi(f_3)n_3}\cdot C^{\mathbf{f}}_{k_1,k_2,k_3}. \tag{3}$$

Equation (3) enables FABLE to perturb the wavelet coefficients $\mathbf{C} = \{\mathbf{C}^{\mathbf{f}}\} \in \mathbb{R}^{8\times\frac{\alpha+1}{2}\times\frac{r}{2}\times\frac{c}{2}}$ to indirectly induce the perturbation $\delta_{\mathbf{X}}$ to the original predictor $\mathbf{X}$. Given an adversarial target $\hat{\mathbf{Y}}'$, FABLE is designed to learn a set of perturbed wavelet coefficients, $\mathbf{C}'$, such that the reconstructed predictor $\mathbf{X}'$ (or $\mathbf{X}'(\mathbf{C}')$), obtained using Equation (3), induces a forecast $g(\mathbf{X}'(\mathbf{C}'))$ that closely aligns with the target $\hat{\mathbf{Y}}'$. Given a weather forecasting model g, the optimization objective for generating the adversarial sample $\mathbf{X}'$ using FABLE is

$$\arg\min_{\mathbf{C}'} \mathcal{L}_f\left(\hat{\mathbf{Y}}', g\left(\mathbf{X}'(\mathbf{C}')\right)\right) + \lambda\mathcal{L}_C(\mathbf{C}, \mathbf{C}'; \omega) \quad \text{s.t. } \|\mathbf{X} - \mathbf{X}'(\mathbf{C}')\|_\infty \leq \epsilon \tag{4}$$

[2]By default, $\alpha + 1$, r, and c are assumed to be even.

where $\mathcal{L}_f\left(\hat{\mathbf{Y}}', g(\mathbf{X}'(\mathbf{C}'))\right) = \sqrt{\frac{1}{\beta \times r \times c}\|\hat{\mathbf{Y}}' - g(\mathbf{X}'(\mathbf{C}'))\|_2^2}$, $\mathcal{L}_C\big(\mathbf{C}, \mathbf{C}'; \omega\big) = \sum_{\mathbf{f}} \omega^{\mathbf{f}} \|\mathbf{C}^{\mathbf{f}} - \mathbf{C}^{\mathbf{f}'}\|_2^2$, and $\|\mathbf{Z}\|_2^2 = \sum_{tij} Z_{tij}^2$. The first term, $\mathcal{L}_f(\cdot)$ ensures *faithfulness* of the adversarial sample $\mathbf{X}'$ by aligning its forecast $g(\mathbf{X}'(\mathbf{C}'))$ to the target $\hat{\mathbf{Y}}'$. The second term, $\mathcal{L}_C(\cdot)$, ensures that the perturbation focuses more on high-frequency instead of low-frequency components by choosing the appropriate set of penalty weights, $\omega^{\mathbf{f}} \in \mathbb{R}$. The hyperparameter λ balances these objectives, offering FABLE the flexibility to tailor adversarial predictors under varying conditions. The constraint enforces the *closeness* of $\mathbf{X}'$ to the original predictor $\mathbf{X}$. We use the Adam optimizer [19] to solve this optimization problem. At each iteration, we enforce the perturbation constraint by clipping the updated input $\mathbf{X}'$ using $\mathbf{X}' \leftarrow \min(\mathbf{X} + \epsilon, \max(\mathbf{X} - \epsilon, \mathbf{X}'))$.

We argue that achieving better geospatio-temporal realisticness and closeness requires applying more perturbation to the high-frequency components in $\mathbf{C}^{\mathbf{f}}$ instead of the low-frequency ones. This intuition is motivated by Theorems 1 and 2 below and the empirical results given in ***Appendix C***.

Theorem 1. *Consider the following level-one Haar wavelet decomposition for a 1-D signal of length T: $\mathbf{f}(2k-n) = \frac{a_0(k)}{\sqrt{2}} + \frac{(-1)^{1-n} d_0(k)}{\sqrt{2}}$, where $n \in \{0,1\}$, $k \in \{1, \ldots, T/2\}$, $\{a_0(k)\}$ is the set of approximation (low-frequency) coefficients, and $\{d_0(k)\}$ is the set of detail (high-frequency) coefficients. Let $\mathbf{f}'_A(t)$ and $\mathbf{f}'_D(t)$ be signals obtained by perturbing only the approximation and detail coefficients of $\mathbf{f}(t)$. Denote their autocorrelations at lag l by $\rho_{\mathbf{f}}(l)$, $\rho_{\mathbf{f}'_A}(l)$, and $\rho_{\mathbf{f}'_D}(l)$. If $\sum_{k=1}^{T/2} |a_0(k)| \geq \sum_{k=1}^{T/2} |d_0(k)|$, then $\sup_l \sum_{l=0}^{T-1} |\rho_{\mathbf{f}'_A}(l) - \rho_{\mathbf{f}}(l)| \geq \sup_l \sum_{l=0}^{T-1} |\rho_{\mathbf{f}'_D}(l) - \rho_{\mathbf{f}}(l)|$.*

Remark 1. *The condition $\sum_{k=1}^{T/2} |a_0(k)| \geq \sum_{k=1}^{T/2} |d_0(k)|$ generally holds for most real-world signals since the approximation coefficients $\{a_0(k)\}$ often capture the majority of the signal's energy at the coarse scale, whereas the detail coefficients $\{d_0(k)\}$ represent the finer fluctuations or noise.*

Theorem 2. *Let $\mathbf{f}(t)$ be a 1-D signal of even length T. Let $\mathbf{f}'_A$ and $\mathbf{f}'_D$ denote the signals obtained by perturbing only the approximation and detail coefficients of $\mathbf{f}(t)$, respectively, with perturbations $\boldsymbol{\delta}_A$ and $\boldsymbol{\delta}_D$. Then, $\|\mathbf{f}'_A - \mathbf{f}\|_2 = \|\boldsymbol{\delta}_A\|_2$ and $\|\mathbf{f}'_D - \mathbf{f}\|_2 = \|\boldsymbol{\delta}_D\|_2$. Moreover, if $\|\boldsymbol{\delta}_A\|_2 \geq \|\boldsymbol{\delta}_D\|_2$, then $\|\mathbf{f}'_A - \mathbf{f}\|_2 \geq \|\mathbf{f}'_D - \mathbf{f}\|_2$.*

Remark 2. *Empirically, whenever the condition $\sum_{k=1}^{T/2} |a_0(k)| \geq \sum_{k=1}^{T/2} |d_0(k)|$ holds, it typically follows that $\|\boldsymbol{\delta}_A\|_2 \geq \|\boldsymbol{\delta}_D\|_2$, since the perturbation strength is observed to be proportional to the magnitude of the corresponding wavelet coefficients. The conclusion of Theorem 2 is extensible to a 3-D signal by expanding the 3-D tensor into a 1-D vector.*

The **proofs** for Theorems 1 and 2 are provided in ***Appendix D*** and ***E***, respectively.

6 Experimental Evaluation

6.1 Experimental Setup

We use two well-known meteorological datasets for our experiments. **(1) *North American Land Data Assimilation System (NLDAS)***[3]: This dataset provides daily weather observations for 1,320 locations over a $1^\circ \times 1^\circ$ grid covering North America from 1979 to 2023. Our study considers 2 of the 9 variables: ***NLDAS-TMP2M*** (*2-meter air temperature*) and ***NLDAS-PRESSFC*** (*surface pressure*) **(2) *ERA5 Reanalysis Data***[4]: This dataset provides global hourly reanalysis weather data on a $5.625^\circ \times 5.625^\circ$ grid, covering 2,048 locations from 1979 to 2018. We focus on *2-meter air temperature (T2M)* and *total incident solar radiation (TISR)*, referred to as ***ERA5-T2M*** and ***ERA5-TISR***, respectively. We selected these weather variables for our experiments due to their inherent autocorrelations. See ***Appendices G.1*** and ***G.2*** for data statistics and preprocessing details.

We compare the performance of FABLE against the following baselines: (1) **Noise Attack** [16], which adds Gaussian random noise to the input; (2) **FGSM** [15], which performs a one-step projected gradient descent (PGD) update; (3) **ALA** [27], which leverages Adam-based updates; (4) **TAAOWPF** [16], an iterative version of FGSM; and (5) **STPGD** [23], which restricts perturbations to victim locations. As some of these methods are originally designed for classification or untargeted attacks only, they must be adapted to a localized targeted attack setting. Details of the baseline adaptation are described

[3] `https://ldas.gsfc.nasa.gov/nldas`

[4] `https://www.ecmwf.int/en/forecasts/dataset/ecmwf-reanalysis-v5`

Table 1: Performance comparison on different datasets with adapted baselines. Within the comparative results for each metric, red entries indicate the best performance, and blue entries indicate the second-best performance. All the results are based on standardized data. On NLDAS datasets, *CLCRN* [22] is the attacked weather forecasting model. On ERA5 datasets, *FourCastNet* [25] is the attacked weather forecasting model.

	NLDAS-TMP2M					NLDAS-PRESSFC				
Method	in-AE ↓	out-AE ↓	Proximity ↓	R_S ↓	R_T ↓	in-AE ↓	out-AE ↓	Proximity ↓	R_S ↓	R_T ↓
FABLE	0.3801 (±0.0650)	2.9123 (±1.4331)	0.0072 (±0.0033)	0.0375 (±0.0282)	0.0191 (±0.0097)	0.5392 (±0.0829)	3.7255 (±0.6604)	0.0069 (±0.0028)	0.0018 (±0.0013)	0.0288 (±0.0104)
ALA	0.3491 (±0.0502)	2.3762 (±1.9276)	0.0156 (±0.0043)	0.0463 (±0.0337)	0.0341 (±0.0090)	0.4701 (±0.0659)	4.2165 (±1.6794)	0.0111 (±0.0047)	0.0023 (±0.0013)	0.0340 (±0.0101)
TAAOWPF	0.3891 (±0.0697)	9.7225 (±1.3211)	0.0351 (±0.0042)	0.0902 (±0.0594)	0.1043 (±0.0236)	0.4810 (±0.070)	15.8075 (±0.4081)	0.0507 (±0.0063)	0.0099 (±0.0021)	0.1621 (±0.0192)
STPGD	0.4434 (±0.2581)	7.8781 (±1.4791)	0.0289 (±0.0057)	0.0942 (±0.0733)	0.0821 (±0.0190)	0.6441 (±0.3026)	14.4016 (±1.5583)	0.0442 (±0.0067)	0.0121 (±0.0081)	0.1270 (±0.0134)
Noise Attack	2.2111 (±0.2745)	288.3196 (±18.7261)	0.0623 (±0.0006)	0.1445 (±0.0574)	0.1543 (±0.0221)	2.4656 (±0.7911)	946.8651 (±21.0277)	0.1303 (±0.0003)	0.0497 (±0.0024)	0.2181 (±0.0147)
FGSM	0.7491 (±0.1546)	1355.4685 (±244.0365)	0.0993 (±0.0004)	0.1207 (±0.0631)	0.1434 (±0.0412)	0.7661 (±0.1113)	1137.3585 (±25.9655)	0.0997 (±0.000)	0.0061 (±0.0016)	0.1454 (±0.0138)
	ERA5-T2M					ERA5-TISR				
Method	in-AE ↓	out-AE ↓	Proximity ↓	R_S ↓	R_T ↓	in-AE ↓	out-AE ↓	Proximity ↓	R_S ↓	R_T ↓
FABLE	3.5990 (±0.8337)	17.8061 (±4.1142)	0.0077 (±0.0016)	0.0007 (±0.0003)	0.0306 (±0.0049)	15.5797 (±2.4477)	38.9485 (±5.9158)	0.0039 (±0.0006)	0.0001 (±0.0000)	0.0235 (±0.0101)
ALA	0.6213 (±0.2364)	19.2709 (±0.9344)	0.0178 (±0.0032)	0.0017 (±0.0008)	0.0657 (±0.0083)	5.9011 (±0.9918)	21.3194 (±2.9044)	0.0132 (±0.0021)	0.0005 (±0.0003)	0.0259 (±0.0096)
TAAOWPF	0.6915 (±0.2196)	40.6883 (±1.4362)	0.0205 (±0.0020)	0.0017 (±0.0007)	0.0786 (±0.0051)	17.1935 (±1.0650)	86.8436 (±3.9767)	0.0172 (±0.0012)	0.0005 (±0.0002)	0.0268 (±0.0101)
STPGD	4.9234 (±6.7289)	48.5430 (±20.0840)	0.0199 (±0.0140)	0.0019 (±0.0014)	0.0563 (±0.0204)	28.7617 (±9.4164)	64.0957 (±8.1363)	0.0092 (±0.0013)	0.0003 (±0.0002)	0.0177 (±0.0111)
Noise Attack	37.7747 (±7.9806)	6236.1825 (±221.9222)	0.1709 (±0.0008)	0.0434 (±0.0043)	0.2253 (±0.0027)	1992.5065 (±76.1890)	11767.8657 (±605.8123)	0.1535 (±0.0011)	0.0429 (±0.0028)	0.0792 (±0.0075)
FGSM	22.1184 (±13.1868)	3520.1989 (±117.1853)	0.0999 (±0.0000)	0.0089 (±0.0009)	0.1876 (±0.0039)	1891.5453 (±132.6871)	5262.2087 (±290.1196)	0.0843 (±0.0007)	0.0076 (±0.0006)	0.0509 (±0.0093)

in ***Appendix F***. We use the pre-trained *CLCRN* [22] and *FourCastNet* [25] (see ***Appendix G.3***) as our underlying weather forecasting models. ***Appendix G.4*** details the construction of adversarial targets for both datasets, with 960 samples constructed for each *NLDAS* and 600 samples for each *ERA5*. To ensure convergence, the number of iterations N is set to 1000. Following the approach in [16], the learning rates for *FABLE*, *ALA*, *TAAOWPF*, and *STPGD* are set to $\frac{2\epsilon}{N}$, with $\epsilon = 2.5$ as the clipping threshold. For *STPGD*, the number of salient locations is selected to be 990 and 1536 on *NLDAS* and *ERA5* datasets, respectively, to ensure faithfulness. More details are in ***Appendix G.5***.

We employ the following metrics to evaluate each method: (1) **faithfulness**, measured by *in-target absolute error*, $\text{in-AE}(g(\mathbf{X}')_{\text{in}}, \hat{\mathbf{Y}}'_{\text{in}}) = \sum_{\tau=1}^{\beta} \left| g(\mathbf{X}')_{\text{in},\tau} - \hat{\mathbf{Y}}'_{\text{in},\tau} \right|$ and *out-target absolute error*, $\text{out-AE}(g(\mathbf{X}')_{\text{out}}, \hat{\mathbf{Y}}'_{\text{out}}) = \sum_{\tau=1}^{\beta} \left\| g(\mathbf{X}')_{\text{out},\tau} - \hat{\mathbf{Y}}'_{\text{out},\tau} \right\|_1$. (2) **Closeness**, measured by the average ℓ_1 distance between $\mathbf{X}$ and $\mathbf{X}'$, $\text{Proximity}(\mathbf{X}', \mathbf{X}) = \frac{1}{\alpha+1} \sum_{\tau=1}^{\alpha+1} \|\mathbf{X}'_\tau - \mathbf{X}_\tau\|_1$. (3) **Geospatio-temporal realisticness**, measured by the difference in spatial (R_S) and temporal (R_T) autocorrelations (see Equation (1)). An effective adversarial sample must be faithful to the target, close to the original predictor, and geospatio-temporally realistic.

6.2 Experimental Results

Performance Comparison. Table 1 compares the performance of ***FABLE*** against other baselines across 4 datasets (2 NLDAS and 2 ERA5 variables) and 2 weather forecasting models (CLCRN and FourCastNet). While ***FABLE*** may not achieve the best faithfulness (in terms of their in-AE and out-AE scores), it consistently ranks among the top performers (2nd or 3rd) across all 4 datasets. In terms of closeness, spatial and temporal realisticness, ***FABLE*** consistently outperforms all the baselines, except for temporal realisticness on the ERA5-TISR dataset, where it has the second-best result. This supports our rationale for adding more perturbation to the high-frequency components of the data, instead of perturbing the input predictor directly, which is the strategy employed by existing methods such as ALA, TAAOWPF, STPGD, and FGSM. The preservation of geospatio-temporal realisticness by ***FABLE*** is also evident from the Moran's I and temporal autocorrelation plots shown

Table 2: Performance evaluation in terms of runtime and memory usage.

Method	CLCRN (NLDAS-TMP2M)		FourCastNet (ERA5-T2M)	
	Total Runtime (s)	Peak Memory (MB)	Total Runtime (s)	Peak Memory (MB)
FABLE	345.87	77.81	112.53	477.57
ALA	482.13	93.40	150.49	501.83
TAAOWPF	498.58	76.14	149.74	483.83
STPGD	490.24	96.30	158.25	502.07
Noise Attack	143.71	70.67	40.43	244.09
FGSM	2.11	76.12	1.07	493.01

Table 3: Impact of the regularization strength λ on FABLE's attack performance on the NLDAS-TMP2M dataset.

λ	In-AE ↓	Out-AE ↓	Proximity ↓	R_S ↓	R_T ↓
0	0.3751	3.3224	0.0254	0.0903	0.0824
1e-6	0.3757	3.3326	0.0236	0.0855	0.0762
1e-5	0.3828	3.2717	0.0158	0.0597	0.0524
1e-4	0.4308	3.4489	0.0072	0.0322	0.0272
1e-3	0.5959	4.6936	0.0027	0.0120	0.0113

in Figure 3. In short, by emphasizing perturbations on higher-frequency coefficients, ***FABLE*** achieves a better balance among faithfulness, closeness, and geospatio-temporal realisticness.

Runtime and Memory Usage. Table 2 reports the wall-clock runtime and peak GPU memory usage of ***FABLE*** and the baselines under two setups: CLCRN on NLDAS-TMP2M and FourCastNet on ERA5-T2M. All experiments were conducted for 500 attack steps, except for FGSM, on a single NVIDIA L4 GPU (22.5 GB), with batch sizes of 48 (CLCRN) and 30 (FourCastNet). To avoid cross-batch memory interference, we run each batch independently. As presented in Table 2, non-learning (Noise Attack) and one-step (FGSM) baselines are fast but yield weaker performance, as evident in Table 1. Compared to other methods (ALA, TAAOWPF, STPGD), ***FABLE*** demonstrates better runtime and lower peak memory usage. This efficiency benefits from excluding the perturbations on the LLL frequency component, thereby reducing the number of parameters to be estimated.

Hyperparameter Sensitivity. We analyze the sensitivity of ***FABLE*** to the regularization strength λ and the frequency-specific penalty weights ω^f. (1) As presented in Table 3, we observe a clear trade-off: increasing λ improves spatial-temporal realisticness but weakens attack effectiveness. (2) Guided by Theorems 1 and 2, smaller penalties were assigned to higher-frequency coefficients to permit larger perturbations. The penalty weights ω^f were selected via grid search on the NLDAS-TMP2M dataset to balance different metrics . This configuration was reused in other datasets, and the results show that ***FABLE*** remains robust without requiring explicit tuning of the weights. For detailed analysis and configurations, please refer to ***Appendix G.6***.

Detectability. Table 4 presents the precision, recall, and F1-score of a wavelet-based anomaly detection method [31] applied to detect adversarial predictors generated by ***FABLE*** and other baselines. This method first performs level-2 wavelet decomposition using the Daubechies-4 basis, then extracts features by computing the energy and entropy of each sub-band. An autoencoder is then trained on these features to minimize reconstruction error. In the detection phase, if an input produces a reconstruction error that exceeds the training mean plus three standard deviations ($\mu + 3\sigma$) using the trained autoEncoder, it will be flagged as adversarial. The results in Table 4 demonstrate that the samples generated by ***FABLE*** are stealthier, and thus, harder to detect.

Table 4: Detection performance of the wavelet-based method against adversarial predictors.

Method	CLCRN on NLDAS-TMP2M		
	Precision ↓	Recall ↓	F1 ↓
FABLE	0.86	0.32	0.47
ALA	0.99	0.69	0.81
TAAOWPF	0.97	1.00	0.98
STPGD	0.98	1.00	0.99
Noise Attack	0.98	1.00	0.99
FGSM	0.98	1.00	0.99

7 Conclusion

In this work, we explore adversarial attacks on weather forecasting models and highlight the key challenges of generating stealthy adversarial samples. We introduce metrics to quantify the spatiotemporal realisticness of adversarial samples and propose ***FABLE***, a novel framework that perturbs wavelet-decomposed coefficients instead of the raw inputs. By focusing on higher-frequency components, this facilitates its generation of adversarial samples that satisfy the *realisticness*, *closeness*, and *faithfulness* criteria. The generality of Theorems 1 and 2 suggests that ***FABLE***'s strategy is to generate attacks on other spatiotemporal data. Future directions include exploring higher-level decompositions and alternative wavelet bases for adversarial attacks, as well as extending ***FABLE*** to multivariate geospatio-temporal settings, where ensuring physical consistency and realisticness across correlated variables remains a key challenge.

References

[1] Arjun Nitin Bhagoji, Daniel Cullina, and Prateek Mittal. Lower bounds on adversarial robustness from optimal transport. *Advances in Neural Information Processing Systems*, 32, 2019.

[2] Kaifeng Bi, Lingxi Xie, Hengheng Zhang, Xin Chen, Xiaotao Gu, and Qi Tian. Accurate medium-range global weather forecasting with 3d neural networks. *Nature*, 619(7970):533–538, 2023.

[3] Alabi Bojesomo, Hasan Al-Marzouqi, and Panos Liatsis. Spatiotemporal vision transformer for short time weather forecasting. In *2021 IEEE International Conference on Big Data (Big Data)*, pages 5741–5746. IEEE, 2021.

[4] Alabi Bojesomo, Hasan Al-Marzouqi, Panos Liatsis, Gao Cong, and Maya Ramanath. Spatiotemporal swin-transformer network for short time weather forecasting. In *CIKM Workshops*, 2021.

[5] Alabi Bojesomo, Hasan AlMarzouqi, and Panos Liatsis. A novel transformer network with shifted window cross-attention for spatiotemporal weather forecasting. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 2023.

[6] Nicholas Carlini, Anish Athalye, Nicolas Papernot, Wieland Brendel, Jonas Rauber, Dimitris Tsipras, Ian Goodfellow, Aleksander Madry, and Alexey Kurakin. On evaluating adversarial robustness. *arXiv preprint arXiv:1902.06705*, 2019.

[7] Anirban Chakraborty, Manaar Alam, Vishal Dey, Anupam Chattopadhyay, and Debdeep Mukhopadhyay. Adversarial attacks and defences: a survey. *ArXiv Preprint ArXiv:1810.00069*, 2018.

[8] Huanran Chen, Yinpeng Dong, Zeming Wei, Hang Su, and Jun Zhu. Towards the worst-case robustness of large language models. *arXiv preprint arXiv:2501.19040*, 2025.

[9] Zihan Chen, Ziyue Wang, Jun-Jie Huang, Wentao Zhao, Xiao Liu, and Dejian Guan. Imperceptible adversarial attack via invertible neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 414–424, 2023.

[10] Jean Côté, Sylvie Gravel, André Méthot, Alain Patoine, Michel Roch, and Andrew Staniforth. The operational cmc–mrb global environmental multiscale (gem) model. part i: Design considerations and formulation. *Monthly Weather Review*, 126(6):1373–1395, 1998.

[11] Lokenath Debnath and Firdous Ahmad Shah. *Wavelet transforms and their applications*, volume 434. Springer, 2015.

[12] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. Boosting adversarial attacks with momentum. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9185–9193, 2018.

[13] Asadullah Hill Galib, Andrew McDonald, Tyler Wilson, Lifeng Luo, and Pang-Ning Tan. Deepextrema: a deep learning approach for forecasting block maxima in time series data. *In Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI*, 2022.

[14] Zhihan Gao, Xingjian Shi, Hao Wang, Yi Zhu, Yuyang Bernie Wang, Mu Li, and Dit-Yan Yeung. Earthformer: exploring space-time transformers for earth system forecasting. *Advances in Neural Information Processing Systems*, 35:25390–25403, 2022.

[15] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *ArXiv Preprint ArXiv:1412.6572*, 2014.

[16] René Heinrich, Christoph Scholz, Stephan Vogt, and Malte Lehna. Targeted adversarial attacks on wind power forecasts. *Machine Learning*, 113(2):863–889, 2024.

[17] Hans Hersbach, Bill Bell, Paul Berrisford, Shoji Hirahara, András Horányi, Joaquín Muñoz-Sabater, Julien Nicolas, Carole Peubey, Raluca Radu, Dinand Schepers, et al. The era5 global reanalysis. *Quarterly Journal of the Royal Meteorological Society*, 146(730):1999–2049, 2020.

[18] Runhai Jiao, Zhuoting Han, Xuan Liu, Changyu Zhou, and Min Du. A gradient-based wind power forecasting attack method considering point and direction selection. *IEEE Transactions on Smart Grid*, 2023.

[19] Diederik P Kingma. Adam: a method for stochastic optimization. *ArXiv Preprint ArXiv:1412.6980*, 2014.

[20] Alexey Kurakin, Ian J Goodfellow, and Samy Bengio. Adversarial examples in the physical world. In *Artificial Intelligence Safety and Security*, pages 99–112. Chapman and Hall/CRC, 2018.

[21] Remi Lam, Alvaro Sanchez-Gonzalez, Matthew Willson, Peter Wirnsberger, Meire Fortunato, Ferran Alet, Suman Ravuri, Timo Ewalds, Zach Eaton-Rosen, Weihua Hu, et al. Learning skillful medium-range global weather forecasting. *Science*, 382(6677):1416–1421, 2023.

[22] Haitao Lin, Zhangyang Gao, Yongjie Xu, Lirong Wu, Ling Li, and Stan Z Li. Conditional local convolution for spatio-temporal meteorological forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 7470–7478, 2022.

[23] Fan Liu, Hao Liu, and Wenzhao Jiang. Practical adversarial attacks on spatiotemporal traffic forecasting models. *Advances in Neural Information Processing Systems*, 35:19035–19047, 2022.

[24] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *ArXiv Preprint ArXiv:1706.06083*, 2017.

[25] Jaideep Pathak, Shashank Subramanian, Peter Harrington, Sanjeev Raja, Ashesh Chattopadhyay, Morteza Mardani, Thorsten Kurth, David Hall, Zongyi Li, Kamyar Azizzadenesheli, et al. Fourcastnet: A global data-driven high-resolution weather model using adaptive fourier neural operators. *arXiv preprint arXiv:2202.11214*, 2022.

[26] Ilan Price, Alvaro Sanchez-Gonzalez, Ferran Alet, Tom R Andersson, Andrew El-Kadi, Dominic Masters, Timo Ewalds, Jacklynn Stott, Shakir Mohamed, Peter Battaglia, et al. Probabilistic weather forecasting with machine learning. *Nature*, 637(8044):84–90, 2025.

[27] Jiaqi Ruan, Qihan Wang, Sicheng Chen, Hanrui Lyu, Gaoqi Liang, Junhua Zhao, and Zhao Yang Dong. On vulnerability of renewable energy forecasting: adversarial learning attacks. *IEEE Transactions on Industrial Informatics*, 2023.

[28] William C Skamarock, Joseph B Klemp, Jimy Dudhia, David O Gill, Dale M Barker, Michael G Duda, Xiang-Yu Huang, Wei Wang, Jordan G Powers, et al. A description of the advanced research wrf version 3. *NCAR Technical Note*, 475(125):10–5065, 2008.

[29] Xingxing Wei, Songping Wang, and Huanqian Yan. Efficient robustness assessment via adversarial spatial-temporal focus on videos. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(9):10898–10912, 2023.

[30] Guoming Wu, Yangfan Xu, Jun Li, Zhiping Shi, and Xianglong Liu. Imperceptible adversarial attack with multi-granular spatio-temporal attention for video action recognition. *IEEE Internet of Things Journal*, 2023.

[31] Yueyue Yao, Jianghong Ma, and Yunming Ye. Regularizing autoencoders with wavelet transform for sequence anomaly detection. *Pattern Recognition*, 134:109084, 2023.

[32] Lyuyi Zhu, Kairui Feng, Ziyuan Pu, and Wei Ma. Adversarial diffusion attacks on graph-based traffic prediction models. *IEEE Internet of Things Journal*, 2023.

Appendix

A Examples of Adversarial Samples by Existing Attack Methods

Figure 5 shows an example of the adversarial samples $\mathbf{X}'$ generated by existing adversarial attack methods on one of the test samples from the NLDAS precipitation dataset, along with its corresponding adversarial forecast $g(X')$. Observe that the random noise attack, which randomly adds Gaussian noise to X, produces an adversarial forecast that affects a significant number of non-targeted locations, leading to poor in-target and out-target faithfulness. While *ALA* and *TAAOWPF* achieve better in-target faithfulness, their closeness is much worse compared to STPGD. These results are consistent with the findings of Figure 2 in Section 4.

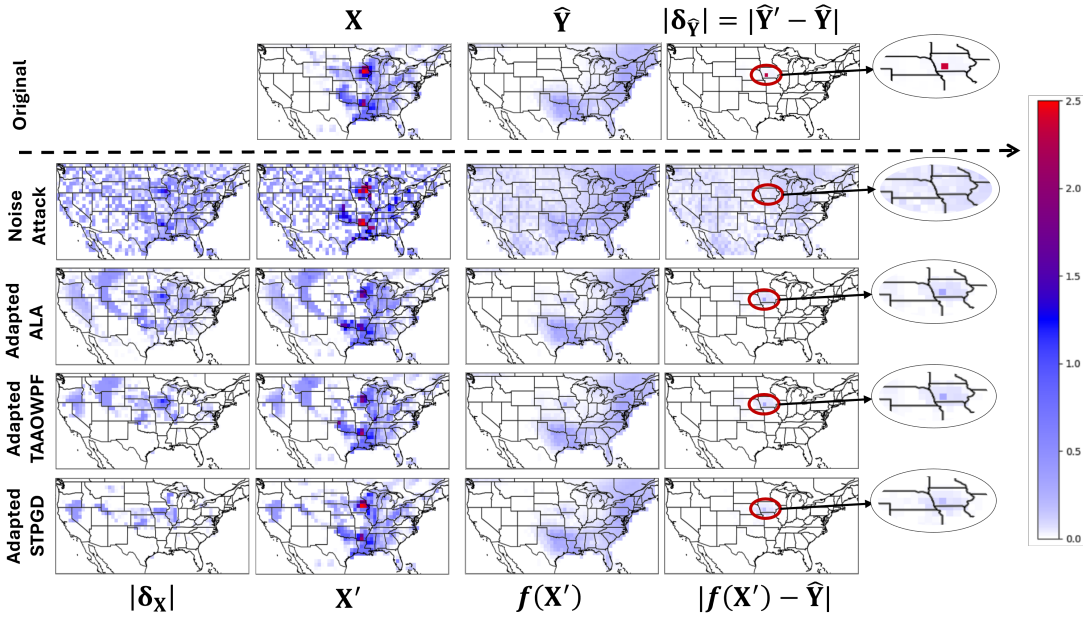

Figure 5: Comparison of adversarial samples generated by existing attack methods and their corresponding forecasts. The first row shows the original predictor X, the original forecast $\hat{Y}$, and the perturbation magnitude $|\delta_{\hat{Y}}|$ used to generate the adversarial target. Subsequent rows show the adversarial predictors X' generated from each baseline and its adversarial forecast $g(X')$. The leftmost column shows the perturbation magnitude $|\delta_X|$ on X while the rightmost column shows the difference $|g(X') - \hat{Y}|$ on $\hat{Y}$. The maximum magnitude of perturbation on X is set to $\epsilon = 2.5$.

B Haar Wavelet Transform

B.1 1-dimensional Haar Wavelet Decomposition

Wavelet decomposition can be used to enable multiresolution analysis of a signal by decomposing it into its underlying frequency components at multiple scales. For the one-dimensional case, the multilevel decomposition of a signal $f(t)$ can be expressed as a linear combination of its low-frequency component at a coarse scale j_0 and higher-frequency details at finer scales $j = \{j_0, j_0 + 1, \cdots\}$ as

$$f(t) = \sum_k a_{j_0}(k) 2^{j_0/2} \phi(2^{j_0} t - k) + \sum_{j=j_0}^{\infty} \sum_k d_j(k) 2^{j/2} \psi(2^j t - k),$$

where $a_{j_0}(k)$ and $d_j(k)$ are known as the approximation and detail coefficients, respectively. The function $\phi(t)$, known as the *scaling function*, is responsible for approximating the low-frequency components of $f(t)$, while $\psi(t)$, known as the *wavelet function*, captures high-frequency variations and localized details. The index k denotes the translation parameter, which determines the spatial or

temporal shifts of the wavelet basis functions. It ensures that $\phi(2^{j_0}t-k)$ and $\psi(2^{j}t-k)$ are properly translated to localize the representation of $f(t)$ at different scales.

In this study, we consider a 1-level decomposition of a signal $f(t)$, which can be expressed as

$$f(t)=\sum_k a_0(k)\phi(t-k)+\sum_k d_0(k)\psi(t-k).$$

Various types of wavelet bases [11] that can be applied to construct the decomposition. Here, we utilize the Haar wavelet basis, whose scaling and wavelet functions are defined by

$$\phi(t)=\begin{cases}1, & 0\le t<1,\\ 0, & \text{otherwise.}\end{cases}$$

and

$$\psi(t)=\begin{cases}1, & 0\le t<0.5,\\ -1, & 0.5\le t<1,\\ 0, & \text{otherwise.}\end{cases}$$

respectively. A key advantage of using the Haar wavelet for the decomposition is that the coefficients $a_0(k)$ and $d_0(k)$ can be efficiently computed via the filter bank method. Specifically, by employing a low-pass filter $h_L=\left[\frac{1}{\sqrt{2}},\frac{1}{\sqrt{2}}\right]$, the approximation coefficients $a_0(k)$ are given by

$$a_0(k)=\frac{f(2k-1)+f(2k)}{\sqrt{2}},\quad k\in\left\{1,2,\ldots,\frac{T}{2}\right\},$$

where $|f(t)|=T$ denotes the length of the signal $f(t)$, which is assumed to be a multiple of 2 (with appropriate padding). In contrast, a high-pass filter $h_H=\left[\frac{1}{\sqrt{2}},-\frac{1}{\sqrt{2}}\right]$ can be used to derive the detail coefficients $d_0(k)$ as

$$d_0(k)=\frac{f(2k-1)-f(2k)}{\sqrt{2}},\quad k\in\left\{1,2,\ldots,\frac{T}{2}\right\}.$$

Based on the computed coefficients, along with the scaling and wavelet functions, the original signal $f(t)$ can be reconstructed as follows:

$$f(2k-1)=\frac{a_0(k)}{\sqrt{2}}+\frac{d_0(k)}{\sqrt{2}}\quad\text{and}\quad f(2k)=\frac{a_0(k)}{\sqrt{2}}-\frac{d_0(k)}{\sqrt{2}},$$

where $k\in\{1,2,\ldots,\frac{T}{2}\}$.

B.2 2-dimensional Haar Wavelet Decomposition

For a two-dimensional data, $\mathbf{X}\in\mathbb{R}^{r\times c}$, its decomposition is performed in two steps. For brevity, we denote the approximation component of the decomposition as L and its detail component as H.

First, a one-dimensional decomposition is applied to each *column* k_3^* of $\mathbf{X}_{i,j}$ along the rows, producing intermediate low-frequency ($C^L_{k_2,k_3^*}$) and high-frequency ($C^H_{k_2,k_3^*}$) coefficients:

$$C^L_{k_2,k_3^*}=\frac{X_{2k_2-1,k_3^*}+X_{2k_2,k_3^*}}{\sqrt{2}},\quad k_2\in\{1,2,\ldots,r/2\},$$
$$C^H_{k_2,k_3^*}=\frac{X_{2k_2-1,k_3^*}-X_{2k_2,k_3^*}}{\sqrt{2}},\quad k_2\in\{1,2,\ldots,r/2\}.$$

Next, a secondary one-dimensional decomposition is applied to each *row* of $C^L_{k_2,k_3^*}$ and $C^H_{k_2,k_3^*}$ along the columns, producing the final approximation ($C^{LL}_{k_2,k_3}$) and detail coefficients ($C^{LH}_{k_2,k_3}$, $C^{HL}_{k_2,k_3}$, and $C^{HH}_{k_2,k_3}$), where:

$$C^{LL}_{k_2,k_3}=\frac{C^L_{k_2,2k_3-1}+C^L_{k_2,2k_3}}{\sqrt{2}},\quad C^{LH}_{k_2,k_3}=\frac{C^L_{k_2,2k_3-1}-C^L_{k_2,2k_3}}{\sqrt{2}},$$
$$C^{HL}_{k_2,k_3}=\frac{C^H_{k_2,2k_3-1}+C^H_{k_2,2k_3}}{\sqrt{2}},\quad C^{HH}_{k_2,k_3}=\frac{C^H_{k_2,2k_3-1}-C^H_{k_2,2k_3}}{\sqrt{2}},$$

where $k_2 \in \{1, 2, \ldots, \lfloor r/2 \rfloor\}$ and $k_3 \in \{1, 2, \ldots, \lfloor c/2 \rfloor\}$. Note that the coefficients C^{LL}, C^{LH}, C^{HL}, and C^{HH} can be directly expressed in terms of the original $\mathbf{X}$ as follows

$$
\begin{aligned}
C^{LL}_{k_2,k_3} &= \frac{X_{2k_2-1,2k_3-1} + X_{2k_2,2k_3-1} + X_{2k_2-1,2k_3} + X_{2k_2,2k_3}}{2}, \\
C^{LH}_{k_2,k_3} &= \frac{X_{2k_2-1,2k_3-1} + X_{2k_2,2k_3-1} - X_{2k_2-1,2k_3} - X_{2k_2,2k_3}}{2}, \\
C^{HL}_{k_2,k_3} &= \frac{X_{2k_2-1,2k_3-1} - X_{2k_2,2k_3-1} + X_{2k_2-1,2k_3} - X_{2k_2,2k_3}}{2}, \\
C^{HH}_{k_2,k_3} &= \frac{X_{2k_2-1,2k_3-1} - X_{2k_2,2k_3-1} - X_{2k_2-1,2k_3} + X_{2k_2,2k_3}}{2}.
\end{aligned}
$$

The original $\mathbf{X}$ can be reconstructed from the coefficients C^{LL}, C^{LH}, C^{HL}, and C^{HH} by

$$
\begin{aligned}
X_{2k_2-1,2k_3-1} &= \frac{C^{LL}_{k_2,k_3} + C^{LH}_{k_2,k_3} + C^{HL}_{k_2,k_3} + C^{HH}_{k_2,k_3}}{2}, \\
X_{2k_2-1,2k_3} &= \frac{C^{LL}_{k_2,k_3} - C^{LH}_{k_2,k_3} + C^{HL}_{k_2,k_3} - C^{HH}_{k_2,k_3}}{2}, \\
X_{2k_2,2k_3-1} &= \frac{C^{LL}_{k_2,k_3} + C^{LH}_{k_2,k_3} - C^{HL}_{k_2,k_3} - C^{HH}_{k_2,k_3}}{2}, \\
X_{2k_2,2k_3} &= \frac{C^{LL}_{k_2,k_3} - C^{LH}_{k_2,k_3} - C^{HL}_{k_2,k_3} + C^{HH}_{k_2,k_3}}{2}.
\end{aligned}
$$

B.3 3-dimensional Haar Wavelet Decomposition

For a three-dimensional sequence $\mathbf{X} \in \mathbb{R}^{(\alpha+1)\times r\times c}$, the coefficients can be derived by sequentially applying the one-dimensional decomposition along the temporal, longitude (row), and latitude (column) dimensions. The resulting coefficients are:

$$
\begin{aligned}
C^{LLL}_{k_1,k_2,k_3} = \frac{1}{\sqrt{8}}\Big(&X_{2k_1-1,\,2k_2-1,\,2k_3-1} + X_{2k_1-1,\,2k_2-1,\,2k_3} + X_{2k_1-1,\,2k_2,\,2k_3-1} + X_{2k_1-1,\,2k_2,\,2k_3} \\
&+ X_{2k_1,\,2k_2-1,\,2k_3-1} + X_{2k_1,\,2k_2-1,\,2k_3} + X_{2k_1,\,2k_2,\,2k_3-1} + X_{2k_1,\,2k_2,\,2k_3}\Big),
\end{aligned}
$$

$$
\begin{aligned}
C^{LLH}_{k_1,k_2,k_3} = \frac{1}{\sqrt{8}}\Big(&X_{2k_1-1,\,2k_2-1,\,2k_3-1} - X_{2k_1-1,\,2k_2-1,\,2k_3} + X_{2k_1-1,\,2k_2,\,2k_3-1} - X_{2k_1-1,\,2k_2,\,2k_3} \\
&+ X_{2k_1,\,2k_2-1,\,2k_3-1} - X_{2k_1,\,2k_2-1,\,2k_3} + X_{2k_1,\,2k_2,\,2k_3-1} - X_{2k_1,\,2k_2,\,2k_3}\Big),
\end{aligned}
$$

$$
\begin{aligned}
C^{LHL}_{k_1,k_2,k_3} = \frac{1}{\sqrt{8}}\Big(&X_{2k_1-1,\,2k_2-1,\,2k_3-1} + X_{2k_1-1,\,2k_2-1,\,2k_3} - X_{2k_1-1,\,2k_2,\,2k_3-1} - X_{2k_1-1,\,2k_2,\,2k_3} \\
&+ X_{2k_1,\,2k_2-1,\,2k_3-1} + X_{2k_1,\,2k_2-1,\,2k_3} - X_{2k_1,\,2k_2,\,2k_3-1} - X_{2k_1,\,2k_2,\,2k_3}\Big),
\end{aligned}
$$

$$
\begin{aligned}
C^{LHH}_{k_1,k_2,k_3} = \frac{1}{\sqrt{8}}\Big(&X_{2k_1-1,\,2k_2-1,\,2k_3-1} - X_{2k_1-1,\,2k_2-1,\,2k_3} - X_{2k_1-1,\,2k_2,\,2k_3-1} + X_{2k_1-1,\,2k_2,\,2k_3} \\
&+ X_{2k_1,\,2k_2-1,\,2k_3-1} - X_{2k_1,\,2k_2-1,\,2k_3} - X_{2k_1,\,2k_2,\,2k_3-1} + X_{2k_1,\,2k_2,\,2k_3}\Big),
\end{aligned}
$$

$$
\begin{aligned}
C^{HLL}_{k_1,k_2,k_3} = \frac{1}{\sqrt{8}}\Big(&X_{2k_1-1,\,2k_2-1,\,2k_3-1} + X_{2k_1-1,\,2k_2-1,\,2k_3} + X_{2k_1-1,\,2k_2,\,2k_3-1} + X_{2k_1-1,\,2k_2,\,2k_3} \\
&- X_{2k_1,\,2k_2-1,\,2k_3-1} - X_{2k_1,\,2k_2-1,\,2k_3} - X_{2k_1,\,2k_2,\,2k_3-1} - X_{2k_1,\,2k_2,\,2k_3}\Big),
\end{aligned}
$$

$$
\begin{aligned}
C^{HLH}_{k_1,k_2,k_3} = \frac{1}{\sqrt{8}}\Big(&X_{2k_1-1,\,2k_2-1,\,2k_3-1} - X_{2k_1-1,\,2k_2-1,\,2k_3} + X_{2k_1-1,\,2k_2,\,2k_3-1} - X_{2k_1-1,\,2k_2,\,2k_3} \\
&- X_{2k_1,\,2k_2-1,\,2k_3-1} + X_{2k_1,\,2k_2-1,\,2k_3} - X_{2k_1,\,2k_2,\,2k_3-1} + X_{2k_1,\,2k_2,\,2k_3}\Big),
\end{aligned}
$$

$$C^{HHL}_{k_1,k_2,k_3} = \frac{1}{\sqrt{8}}\Big(X_{2k_1-1,\,2k_2-1,\,2k_3-1} + X_{2k_1-1,\,2k_2-1,\,2k_3} - X_{2k_1-1,\,2k_2,\,2k_3-1} - X_{2k_1-1,\,2k_2,\,2k_3} - X_{2k_1,\,2k_2-1,\,2k_3-1} - X_{2k_1,\,2k_2-1,\,2k_3} + X_{2k_1,\,2k_2,\,2k_3-1} + X_{2k_1,\,2k_2,\,2k_3}\Big),$$

$$C^{HHH}_{k_1,k_2,k_3} = \frac{1}{\sqrt{8}}\Big(X_{2k_1-1,\,2k_2-1,\,2k_3-1} - X_{2k_1-1,\,2k_2-1,\,2k_3} - X_{2k_1-1,\,2k_2,\,2k_3-1} + X_{2k_1-1,\,2k_2,\,2k_3} - X_{2k_1,\,2k_2-1,\,2k_3-1} + X_{2k_1,\,2k_2-1,\,2k_3} + X_{2k_1,\,2k_2,\,2k_3-1} - X_{2k_1,\,2k_2,\,2k_3}\Big).$$

The original X can be reconstructed from the coefficients by

$$X_{2k_1-1,\,2k_2-1,\,2k_3-1} = \frac{C^{LLL}_{k_1,k_2,k_3} + C^{LLH}_{k_1,k_2,k_3} + C^{LHL}_{k_1,k_2,k_3} + C^{LHH}_{k_1,k_2,k_3} + C^{HLL}_{k_1,k_2,k_3} + C^{HLH}_{k_1,k_2,k_3} + C^{HHL}_{k_1,k_2,k_3} + C^{HHH}_{k_1,k_2,k_3}}{\sqrt{8}},$$

$$X_{2k_1-1,\,2k_2-1,\,2k_3} = \frac{C^{LLL}_{k_1,k_2,k_3} - C^{LLH}_{k_1,k_2,k_3} + C^{LHL}_{k_1,k_2,k_3} - C^{LHH}_{k_1,k_2,k_3} + C^{HLL}_{k_1,k_2,k_3} - C^{HLH}_{k_1,k_2,k_3} + C^{HHL}_{k_1,k_2,k_3} - C^{HHH}_{k_1,k_2,k_3}}{\sqrt{8}},$$

$$X_{2k_1-1,\,2k_2,\,2k_3-1} = \frac{C^{LLL}_{k_1,k_2,k_3} + C^{LLH}_{k_1,k_2,k_3} - C^{LHL}_{k_1,k_2,k_3} - C^{LHH}_{k_1,k_2,k_3} + C^{HLL}_{k_1,k_2,k_3} + C^{HLH}_{k_1,k_2,k_3} - C^{HHL}_{k_1,k_2,k_3} - C^{HHH}_{k_1,k_2,k_3}}{\sqrt{8}},$$

$$X_{2k_1-1,\,2k_2,\,2k_3} = \frac{C^{LLL}_{k_1,k_2,k_3} - C^{LLH}_{k_1,k_2,k_3} - C^{LHL}_{k_1,k_2,k_3} + C^{LHH}_{k_1,k_2,k_3} + C^{HLL}_{k_1,k_2,k_3} - C^{HLH}_{k_1,k_2,k_3} - C^{HHL}_{k_1,k_2,k_3} + C^{HHH}_{k_1,k_2,k_3}}{\sqrt{8}},$$

$$X_{2k_1,\,2k_2-1,\,2k_3-1} = \frac{C^{LLL}_{k_1,k_2,k_3} + C^{LLH}_{k_1,k_2,k_3} + C^{LHL}_{k_1,k_2,k_3} + C^{LHH}_{k_1,k_2,k_3} - C^{HLL}_{k_1,k_2,k_3} - C^{HLH}_{k_1,k_2,k_3} - C^{HHL}_{k_1,k_2,k_3} - C^{HHH}_{k_1,k_2,k_3}}{\sqrt{8}},$$

$$X_{2k_1,\,2k_2-1,\,2k_3} = \frac{C^{LLL}_{k_1,k_2,k_3} - C^{LLH}_{k_1,k_2,k_3} + C^{LHL}_{k_1,k_2,k_3} - C^{LHH}_{k_1,k_2,k_3} - C^{HLL}_{k_1,k_2,k_3} + C^{HLH}_{k_1,k_2,k_3} - C^{HHL}_{k_1,k_2,k_3} + C^{HHH}_{k_1,k_2,k_3}}{\sqrt{8}},$$

$$X_{2k_1,\,2k_2,\,2k_3-1} = \frac{C^{LLL}_{k_1,k_2,k_3} + C^{LLH}_{k_1,k_2,k_3} - C^{LHL}_{k_1,k_2,k_3} - C^{LHH}_{k_1,k_2,k_3} - C^{HLL}_{k_1,k_2,k_3} - C^{HLH}_{k_1,k_2,k_3} + C^{HHL}_{k_1,k_2,k_3} + C^{HHH}_{k_1,k_2,k_3}}{\sqrt{8}},$$

$$X_{2k_1,\,2k_2,\,2k_3} = \frac{C^{LLL}_{k_1,k_2,k_3} - C^{LLH}_{k_1,k_2,k_3} - C^{LHL}_{k_1,k_2,k_3} + C^{LHH}_{k_1,k_2,k_3} - C^{HLL}_{k_1,k_2,k_3} + C^{HLH}_{k_1,k_2,k_3} + C^{HHL}_{k_1,k_2,k_3} - C^{HHH}_{k_1,k_2,k_3}}{\sqrt{8}},$$

C Effect of Varying Perturbation on Different Frequency Components

Figure 6 presents an ablation study on the effect of varying the perturbation magnitude ω of ***FABLE*** to different frequency coefficients of the input sample. The experiment is performed on the NLDAS-TMP2M dataset, using 96 samples from its test set. A larger penalty weight (ω) applied to a frequency coefficient in the loss function shown in Equation (4) leads to a smaller perturbation of the coefficient. The top-left panel of Figure 6 shows the penalty weights associated with four different configuration schemes. As we go from configuration 1 to configuration 4, the perturbation magnitude will be increasingly geared towards the low-frequency components rather than the high-frequency ones. In other words, configuration 1 is biased towards adding more perturbations to the high frequency components, whereas configuration 4 is biased towards more perturbations on the low frequency ones.

The remaining 5 panels in Figure 6 shows the performance of ***FABLE*** under the different configurations. By varying the perturbation magnitude on different frequency coefficients, we observe the trade-offs among faithfulness, geospatio-temporal realisticness, and closeness measures. Our results indicate that increasing the perturbation magnitude on low-frequency coefficients leads to improvements in faithfulness, as shown by the decreasing trend in both in-AE and out-AE. However, the proximity, spatial realisticness (R_S), and temporal realisticness (R_T) metrics show an increasing trend when emphasis is placed on perturbing the low-frequency components.

These findings suggest that adding perturbation primarily to the high-frequency coefficients can enhance adversarial attack performance in terms of closeness and geospatial-temporal realisticness, albeit at the expense of degrading its faithfulness.

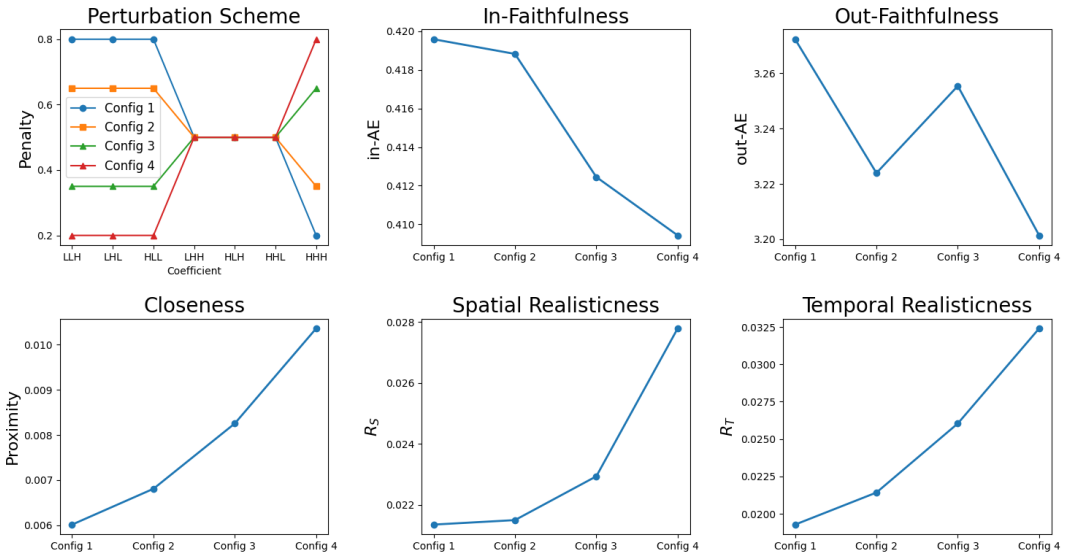

Figure 6: Effect of varying the perturbation magnitude to different frequency coefficients of the input sample. Lower metric values indicate better performance.

D Proof of Theorem 1

Theorem (Restatement of Theorem 1)**.** *Consider the following level-one Haar wavelet decomposition for a 1-D signal of length T:* $\mathbf{f}(2k-n) = \frac{a_0(k)}{\sqrt{2}} + \frac{(-1)^{1-n} d_0(k)}{\sqrt{2}}$, *where* $n \in \{0,1\}$, $k \in \{1,\ldots,T/2\}$, $\{a_0(k)\}$ *is the set of approximation (low-frequency) coefficients, and* $\{d_0(k)\}$ *is the set of detail (high-frequency) coefficients. Let* $\mathbf{f}'_A(t)$ *and* $\mathbf{f}'_D(t)$ *be signals obtained by perturbing only the approximation and detail coefficients of* $\mathbf{f}(t)$*. Denote their autocorrelations at lag l by* $\rho_{\mathbf{f}}(l)$, $\rho_{\mathbf{f}'_A}(l)$, *and* $\rho_{\mathbf{f}'_D}(l)$*. If* $\sum_{k=1}^{T/2} |a_0(k)| \geq \sum_{k=1}^{T/2} |d_0(k)|$, *then* $\sup_l \sum_{l=0}^{T-1} |\rho_{\mathbf{f}'_A}(l) - \rho_{\mathbf{f}}(l)| \geq \sup_l \sum_{l=0}^{T-1} |\rho_{\mathbf{f}'_D}(l) - \rho_{\mathbf{f}}(l)|$.

Proof. Consider a 1-dimensional time series $f(t)$ of length T whose level-one Haar decomposition can be written as follows:

$$f(2k-n) = \frac{a_0(k)}{\sqrt{2}} + \frac{(-1)^{1-n} d_0(k)}{\sqrt{2}} \equiv f_A(2k-n) + f_D(2k-n), \tag{5}$$

where $f_A(2k-n) = \frac{a_0(k)}{\sqrt{2}}$ is known as the approximation coefficient, $f_D(2k-n) = \frac{(-1)^{1-n} d_0(k)}{\sqrt{2}}$ is the detail coefficient, $n \in \{0,1\}$, and $k \in \{1,2,\ldots,\frac{T}{2}\}$. Let $\{\delta_A(k)\}$ and $\{\delta_D(k)\}$ denote the perturbations added to the approximation and detail coefficients, respectively, and assume:

$$\forall k: \ \delta_A(k) \leq \epsilon_A \quad \text{and} \quad \delta_D(k) \leq \epsilon_D. \tag{6}$$

Case 1: Perturbation on approximation coefficients. Consider the perturbed time series $f'_A(2k-n)$ obtained by modifying only the approximation coefficients of the original time series. We may write

$$f'_A(2k-n) = f(2k-n) \ + \ \frac{\delta_A(k)}{\sqrt{2}} \equiv f(2k-n) + \Delta f_A(2k-n),$$

where $\Delta f_A(2k-n) \ = \ \frac{\delta_A(k)}{\sqrt{2}}$. Based on the definition of autocorrelation function, we have

$$\begin{aligned} |\rho_{f'_A}(l) - \rho_f(l)| \ = \ \Big| \sum_n \sum_k \big[f(2k-n) + \Delta f_A(2k-n)\big] \big[f(2k-n-l) + \Delta f_A(2k-n-l)\big] \\ - \sum_n \sum_k f(2k-n) f(2k-n-l) \Big|, \end{aligned}$$

where $l \in \{0, 1, 2, ..., 2k-n-1\}$ for each k and n. Expanding the product inside the sum, we obtain

$$\begin{aligned}\big|\rho_{f'_A}(l) - \rho_f(l)\big| = \big| &\underbrace{\sum_n \sum_k f(2k-n)\,\Delta f_A(2k-n-l)}_{\text{(I)}} \\ &+ \underbrace{\sum_n \sum_k \Delta f_A(2k-n)\, f(2k-n-l)}_{\text{(II)}} \\ &+ \underbrace{\sum_n \sum_k \Delta f_A(2k-n)\,\Delta f_A(2k-n-l)}_{\text{(III)}}\big|. \end{aligned} \tag{7}$$

(1) Analysis for term (I). Using (5), we split $f(2k-n)$ into $f_A(2k-n) + f_D(2k-n)$, yielding

$$\text{(I)} = \sum_n \sum_k f_A(2k-n)\,\Delta f_A(2k-n-l) + \sum_n \sum_k f_D(2k-n)\,\Delta f_A(2k-n-l). \tag{8}$$

The first term on the right-hand side can be bounded from above as follows:

$$\sum_{n\in\{0,1\}} \sum_{k=\frac{l+2}{2}}^{\frac{T}{2}} f_A(2k-n)\Delta f_A(2k-n-l) \le \epsilon_A \sum_{k=\frac{l+2}{2}}^{\frac{T}{2}} a_0(k)$$

when l is even (note that the upper bound of k in the summation arises from the constraint $\max\{2k-n, 2k-n-l\} = T$ for $l \ge 0$ and $n \in \{0,1\}$; and the lower bound of k in the summation arises from the constraint $\min\{2k-n, 2k-n-l\} = 1$ for $l \ge 0$ and $n \in \{0,1\}$), and

$$\sum_{n\in\{0,1\}} \sum_{k=\frac{l+1}{2}}^{\frac{T}{2}} f_A(2k-n)\Delta f_A(2k-n-l) \le \epsilon_A \sum_{k=\frac{l+3}{2}}^{\frac{T}{2}} a_0(k) + \frac{1}{2} a_0(\frac{l+1}{2})\epsilon_A$$

when l is odd, where $n \neq 1$ when $k = \frac{l+1}{2}$ (note that the upper bound of k in the summation follows from the same constraint discussed above; and the lower bound of k in the summation arises from the constraint $\min\{2k-n, 2k-n-l\} = 1$ for $l \ge 1$ and $n \in \{0,1\}$). The second term on the right-hand side of (8) can be simplified as follows:

$$\begin{aligned}\sum_{n\in\{0,1\}} \sum_{k=\frac{l+2}{2}}^{\frac{T}{2}} f_D(2k-n)\,\Delta f_A(2k-n-l) &= \sum_{n\in\{0,1\}} \sum_{k=\frac{l+2}{2}}^{\frac{T}{2}} \frac{(-1)^{1-n} d_0(k)}{\sqrt{2}} \frac{\delta_A(k-\frac{l}{2})}{\sqrt{2}} \\ &= \sum_{k=\frac{l+2}{2}}^{\frac{T}{2}} \frac{d_0(k)\delta_A(k-\frac{l}{2})}{2} \sum_{n\in\{0,1\}} (-1)^{1-n} \\ &= 0\end{aligned}$$

when l is even (note that the upper and lower bounds of k follows from the same constraint discussed above when l is even), and

$$\begin{aligned}\sum_{n\in\{0,1\}} \sum_{k=\frac{l+1}{2}}^{\frac{T}{2}} f_D(2k-n)\,\Delta f_A(2k-n-l) &= 0 + f_D(2\times\frac{l+1}{2} - 0)\Delta f_A(T-l) \\ &\le -\frac{d_0(\frac{l+1}{2})\epsilon_A}{2}\end{aligned}$$

when l is odd, where $n \neq 1$ when $k = \frac{l+1}{2}$ (note that the upper and lower bounds of k follows from the same constraint discussed above when l is odd). In summary,

$$\text{(I)} \le \begin{cases} \epsilon_A \displaystyle\sum_{k=\frac{l+2}{2}}^{\frac{T}{2}} a_0(k) & \text{if } l \bmod 2 = 0, \\ \epsilon_A \displaystyle\sum_{k=\frac{l+3}{2}}^{\frac{T}{2}} a_0(k) + \frac{a_0(\frac{l+1}{2})\epsilon_A}{2} - \frac{d_0(\frac{l+1}{2})\epsilon_A}{2} & \text{if } l \bmod 2 = 1. \end{cases}$$

(2) Analysis for term (II). Similarly, we have

$$\text{(II)} = \sum_n \sum_k \Delta f_A(2k-n) f_A(2k-n-l) + \sum_n \sum_k \Delta f_A(2k-n) f_D(2k-n-l).$$

On the one hand, we have

$$\sum_{n\in\{0,1\}} \sum_{k=\frac{l+2}{2}}^{\frac{T}{2}} \Delta f_A(2k-n) f_A(2k-n-l) \le \epsilon_A \sum_{k=\frac{l+2}{2}}^{\frac{T}{2}} a_0(k-\frac{l}{2})$$

when l is even, and

$$\sum_{n\in\{0,1\}} \sum_{k=\frac{l+1}{2}}^{\frac{T}{2}} \Delta f_A(2k-n) f_A(2k-n-l) \le \epsilon_A \sum_{k=\frac{l+3}{2}}^{\frac{T}{2}} a_0(k-\frac{l+1}{2}) + \frac{\epsilon_A a_0(\frac{T}{2}-\frac{l-1}{2})}{2}$$

when l is odd, where $n \neq 1$ when $k = \frac{l+1}{2}$. On the other hand,

$$\begin{aligned}\sum_{n\in\{0,1\}} \sum_{k=\frac{l+2}{2}}^{\frac{T}{2}} \Delta f_A(2k-n) f_D(2k-n-l) &= \sum_{n\in\{0,1\}} \sum_{k=\frac{l+2}{2}}^{\frac{T}{2}} \frac{\delta_A(k)}{\sqrt{2}} \frac{(-1)^{1-n} d_0(k-\frac{l}{2})}{\sqrt{2}} \\ &= 0\end{aligned}$$

when l is even, and

$$\sum_{n\in\{0,1\}} \sum_{k=\frac{l+1}{2}}^{\frac{T}{2}} \Delta f_A(2k-n) f_D(2k-n-l) \le \frac{d_0(\frac{T}{2}-\frac{l-1}{2})\epsilon_A}{2}$$

when l is odd, where $n \neq 1$ when $k = \frac{l+1}{2}$. In summary,

$$\text{(II)} \le \begin{cases} \epsilon_A \sum_{k=\frac{l+2}{2}}^{\frac{T}{2}} a_0(k-\frac{l}{2}) & \text{if } l \bmod 2 = 0, \\ \epsilon_A \sum_{k=\frac{l+3}{2}}^{\frac{T}{2}} a_0(k-\frac{l+1}{2}) + \frac{a_0(\frac{T}{2}-\frac{l-1}{2})\epsilon_A}{2} + \frac{d_0(\frac{T}{2}-\frac{l-1}{2})\epsilon_A}{2} & \text{if } l \bmod 2 = 1. \end{cases}$$

(3) Analysis for term (III). We obtain

$$\text{(III)} \le \frac{T-l}{2}\epsilon_A^2.$$

Substitute these inequalities back to Equation (7), we obtain that

$$|R_{f'_A}(l) - R_f(l)| \le |\epsilon_A \sum_{k=\frac{l+2}{2}}^{\frac{T}{2}} a_0(k) + \epsilon_A \sum_{k=\frac{l+2}{2}}^{\frac{T}{2}} a_0(k-\frac{l}{2}) + \frac{T-l}{2}\epsilon_A^2| \tag{9}$$

when l is even, and

$$\begin{aligned}|\rho_{f'_A}(l) - \rho_f(l)| \le |\epsilon_A &\sum_{k=\frac{l+3}{2}}^{\frac{T}{2}} a_0(k) + \frac{a_0(\frac{l+1}{2})\epsilon_A}{2} - \frac{d_0(\frac{l+1}{2})\epsilon_A}{2} \\ &+ \epsilon_A \sum_{k=\frac{l+3}{2}}^{\frac{T}{2}} a_0(k-\frac{l+1}{2}) + \frac{a_0(\frac{T}{2}-\frac{l-1}{2})\epsilon_A}{2} + \frac{d_0(\frac{T}{2}-\frac{l-1}{2})\epsilon_A}{2} + \frac{T-l}{2}\epsilon_A^2| \end{aligned} \tag{10}$$

when l is odd. Summing over all l, we obtain

$$\begin{aligned}\sum_{l=0}^{T-1} |\rho_{f'_A}(l) - \rho_f(l)| \le &\sum_{k=1}^{\frac{T}{2}} |a_0(k)(2k-1)\epsilon_A| + \sum_{i=1}^{\frac{T}{2}} |a_0(k)(T-2k+1)\epsilon_A| \\ &+ \frac{T^2+T}{4}\epsilon_A^2 + \sum_{k=1}^{\frac{T}{2}} |a_0(k)\epsilon_A| + 0 \\ \le &|\epsilon_A|(T+1)\sum_{k=1}^{\frac{T}{2}} |a_0(k)| + \frac{T^2+T}{4}\epsilon_A^2. \end{aligned} \tag{11}$$

Case 2: Perturbation in the detailed coefficients. Consider a function $f_D(2k-n)$ that is perturbed by modifying only its detail coefficients in the Haar wavelet decomposition. We write

$$f'_D(2k-n) = f(2k-n) + \frac{(-1)^{1-n}\delta_D(k)}{\sqrt{2}}.$$

Define

$$\Delta f_D(2k-n) = \frac{(-1)^{1-n}\delta_D(k)}{\sqrt{2}}. \tag{12}$$

So, we can obtain

$$\begin{aligned}\left|\rho_{f'_D}(l) - \rho_f(l)\right| = \Big|&\underbrace{\sum_n \sum_k f(2k-n)\,\Delta f_D(2k-n-l)}_{\text{(I)}} \\ &+ \underbrace{\sum_n \sum_k \Delta f_D(2k-n)\, f(2k-n-l)}_{\text{(II)}} \\ &+ \underbrace{\sum_n \sum_k \Delta f_D(2k-n)\,\Delta f_D(2k-n-l)}_{\text{(III)}}\Big|.\end{aligned} \tag{13}$$

(1) Analysis of term (I). We can obtain

$$\text{(I)} \le \begin{cases} \epsilon_D \sum_{k=\frac{l+2}{2}}^{\frac{T}{2}} d_0(k) & \text{if } l \bmod 2 = 0, \\ \frac{\epsilon_D}{2} a_0(\frac{l+1}{2}) - \epsilon_D \sum_{k=\frac{l+3}{2}}^{\frac{T}{2}} d_0(k) - \frac{\epsilon_D}{2} d_0(\frac{l+1}{2}) & \text{if } l \bmod 2 = 1. \end{cases}$$

(2) Analysis of term (II). We can obtain

$$\text{(II)} \le \begin{cases} \epsilon_D \sum_{k=\frac{l+2}{2}}^{\frac{T}{2}} d_0(k - \frac{l}{2}) & \text{if } l \bmod 2 = 0, \\ -\epsilon_D \sum_{k=\frac{l+3}{2}}^{\frac{T}{2}} d_0(k - \frac{l+1}{2}) - \frac{\epsilon_D}{2} a_0(\frac{T}{2} - \frac{l-1}{2}) - \frac{\epsilon_D}{2} d_0(\frac{T}{2} - \frac{l-1}{2}) & \text{if } l \bmod 2 = 1. \end{cases}$$

(3) Analysis of term (III).

$$\text{(III)} \le \begin{cases} \frac{T-l}{2}\epsilon_D^2 & \text{if } l \bmod 2 = 0, \\ -\frac{T-l}{2}\epsilon_D^2 & \text{if } l \bmod 2 = 1. \end{cases}$$

Substitute there inequalities back to Equation (13), we obtain that

$$|\rho_{f'_D}(l) - \rho_f(l)| \le |\epsilon_D \sum_{k=\frac{l+2}{2}}^{\frac{T}{2}} d_0(k) + \epsilon_D \sum_{k=\frac{l+2}{2}}^{\frac{T}{2}} d_0(k - \frac{l}{2}) + \frac{T-l}{2}\epsilon_D^2| \tag{14}$$

when l is even, and

$$\begin{aligned}|\rho_{f'_D}(l) - \rho_f(l)| \le |&\frac{\epsilon_D}{2} a_0(\frac{l+1}{2}) - \epsilon_D \sum_{k=\frac{l+3}{2}}^{\frac{T}{2}} d_0(k) - \frac{\epsilon_D}{2} d_0(\frac{l+1}{2}) \\ &- \epsilon_D \sum_{k=\frac{l+3}{2}}^{\frac{T}{2}} d_0(k - \frac{l+1}{2}) - \frac{\epsilon_D}{2} a_0(\frac{T}{2} - \frac{l-1}{2}) - \frac{\epsilon_D}{2} d_0(\frac{T}{2} - \frac{l-1}{2}) - \frac{T-l}{2}\epsilon_D^2|\end{aligned} \tag{15}$$

when l is odd. Summing over all l, we obtain

$$\sum_{l=0}^{T-1} |\rho_{f'_D}(l) - \rho_f(l)| \leq |\epsilon_D|(T+1)\sum_{k=1}^{\frac{T}{2}} |d_0(k)| + \frac{T^2+T}{4}\epsilon_D^2. \tag{16}$$

Compare the upper bounds in Equations (11) and (16). Obviously, the problem of comparing the two upper bounds can be equivalent to comparing the values of $\sum_{k=1}^{\frac{T}{2}} |a_0(k)|$ and $\sum_{k=1}^{\frac{T}{2}} |d_0(k)|$. Thus, for any one-dimensional signal $\mathbf{f}(\mathbf{t})$ of length T. If it satisfies the condition

$$\sum_{k=1}^{\frac{T}{2}} |a_0(k)| \geq \sum_{k=1}^{\frac{T}{2}} |d_0(k)|,$$

then we obtain,

$$\sup_l \left(\sum_{l=0}^{T-1} |\rho_{f'_A}(l) - \rho_f(l)| \right) \geq \sup_l \left(\sum_{l=0}^{T-1} |\rho_{f'_D}(l) - \rho_f(l)| \right).$$

□

To explore the extension of Theorem 1 to the three-dimensional setting, we introduce Corollary 1. Let $\mathbf{f} \in \mathbb{R}^{T\times H\times W}$ be a 3-D signal. For any spatial-temporal coordinate (X_1, X_2, X_3) in the domain of $\mathbf{f}$, the autocorrelation function is defined analogously as

$$\rho(l_1, l_2, l_3) = f(X_1, X_2, X_3) \cdot f(X_1 - l_1, X_2 - l_2, X_3 - l_3),$$

where $l_1, l_2, l_3 \in \mathbb{Z}$ denote the lags along an arbitrary of the three dimensions. We define the following sets of lags:

$$S = \{(l_1, l_2, l_3) \mid 0 \leq l_1 \leq T-1,\ 0 \leq l_2 \leq H-1,\ 0 \leq l_3 \leq W-1\},$$
$$S_1 = \{(l_1, 0, 0) \mid 0 \leq l_1 \leq T-1\},$$
$$S_2 = \{(0, l_2, 0) \mid 0 \leq l_2 \leq H-1\},$$
$$S_3 = \{(0, 0, l_3) \mid 0 \leq l_3 \leq W-1\}.$$

Let $\mathbf{f}'_A$ and $\mathbf{f}'_D$ denote the signals obtained by perturbing only the approximation coefficients and only the detail coefficients of $\mathbf{f}$, respectively. Define

$$\Delta\rho_{\mathbf{f}_A}(l_1, l_2, l_3) = \left|\rho_{\mathbf{f}'_A}(l_1, l_2, l_3) - \rho_{\mathbf{f}}(l_1, l_2, l_3)\right|,$$

$$\Delta\rho_{\mathbf{f}_D}(l_1, l_2, l_3) = \left|\rho_{\mathbf{f}'_D}(l_1, l_2, l_3) - \rho_{\mathbf{f}}(l_1, l_2, l_3)\right|.$$

Corollary 1. $\max_{i=1,2,3} \sup_{S_i} \Delta\rho_{\mathbf{f}_D} \leq \max_{i=1,2,3} \sup_{S_i} \Delta\rho_{\mathbf{f}_A}$.

Proof. Under the 1-D condition of Theorem 1, along each dimension, we have

$$\sup_{(l_1,0,0)\in S_1} \Delta\rho_{\mathbf{f_D}}(l_1, 0, 0) \leq \sup_{(l_1,0,0)\in S_1} \Delta\rho_{\mathbf{f_A}}(l_1, 0, 0),$$

$$\sup_{(0,l_2,0)\in S_2} \Delta\rho_{\mathbf{f_D}}(0, l_2, 0) \leq \sup_{(0,l_2,0)\in S_2} \Delta\rho_{\mathbf{f_A}}(0, l_2, 0),$$

$$\sup_{(0,0,l_3)\in S_3} \Delta\rho_{\mathbf{f_D}}(0, 0, l_3) \leq \sup_{(0,0,l_3)\in S_3} \Delta\rho_{\mathbf{f_A}}(0, 0, l_3).$$

So, we obtain

$$\max\{\sup_{(l_1,0,0)\in S_1} \Delta\rho_{\mathbf{f_D}}(l_1, 0, 0), \sup_{(0,l_2,0)\in S_2} \Delta\rho_{\mathbf{f_D}}(0, l_2, 0), \sup_{(0,0,l_3)\in S_3} \Delta\rho_{\mathbf{f_D}}(0, 0, l_3)\}$$

$$\leq \max\{\sup_{(l_1,0,0)\in S_1} \Delta\rho_{\mathbf{f_A}}(l_1, 0, 0), \sup_{(0,l_2,0)\in S_2} \Delta\rho_{\mathbf{f_A}}(0, l_2, 0), \sup_{(0,0,l_3)\in S_3} \Delta\rho_{\mathbf{f_A}}(0, 0, l_3)\}.$$

□

E Proof of Theorem 2

Theorem (Restatement of Theorem 2)**.** *Let $\mathbf{f}(t)$ be a 1-D signal of even length T. Let $\mathbf{f}'_A$ and $\mathbf{f}'_D$ denote the signals obtained by perturbing only the approximation and detail coefficients of $\mathbf{f}(t)$, respectively, with perturbations $\boldsymbol{\delta}_A$ and $\boldsymbol{\delta}_D$. Then, $\|\mathbf{f}'_A - \mathbf{f}\|_2 = \|\boldsymbol{\delta}_A\|_2$ and $\|\mathbf{f}'_D - \mathbf{f}\|_2 = \|\boldsymbol{\delta}_D\|_2$. Moreover, if $\|\boldsymbol{\delta}_A\|_2 \geq \|\boldsymbol{\delta}_D\|_2$, then $\|\mathbf{f}'_A - \mathbf{f}\|_2 \geq \|\mathbf{f}'_D - \mathbf{f}\|_2$.*

Proof. Given a one-dimensional time series $\mathbf{f}(\mathbf{t})$ of length T. The level-one wavelet decomposition of $\mathbf{f}$ based on the Haar wavelet in the one-dimensional space can be represented as

$$f(2k-n) = \frac{a_0(k)}{\sqrt{2}} + \frac{(-1)^{1-n} d_0(k)}{\sqrt{2}},$$

where $n \in \{0, 1\}$, and $k \in \{1, 2, \ldots, \frac{T}{2}\}$.

Let $\mathbf{f}'_A$ and $\mathbf{f}'_D$ be the perturbed versions of $\mathbf{f}$, reconstructed from coefficient perturbations applied only to the approximation or detail components, respectively. Let $\mathbf{W} \in \mathbb{R}^{T \times T}$ denote the orthogonal Haar decomposition matrix such that $\mathbf{f} = \mathbf{W}[\mathbf{a}_0; \mathbf{d}_0]$, where $[\mathbf{a}_0; \mathbf{d}_0] \in \mathbb{R}^T$ concatenates the level-1 approximation and detail coefficients. That is,

$$\begin{pmatrix} f(1) \\ f(2) \\ \ldots \\ f(2k) \end{pmatrix} = \mathbf{W} \cdot \begin{pmatrix} a_0(1) \\ a_0(2) \\ \ldots \\ a_0(k) \\ d_0(1) \\ d_0(2) \\ \ldots \\ d_0(k) \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 & 1 & 0 & 0 & \cdots & 0 \\ 1 & 0 & 0 & \cdots & 0 & -1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 & 0 & 1 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 & 0 & -1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 0 & 0 & 0 & \cdots & 1 \\ 0 & 0 & 0 & \cdots & 1 & 0 & 0 & 0 & \cdots & -1 \end{pmatrix} \cdot \begin{pmatrix} a_0(1) \\ a_0(2) \\ \ldots \\ a_0(k) \\ d_0(1) \\ d_0(2) \\ \ldots \\ d_0(k) \end{pmatrix}$$

Then the perturbed signals are

$$\mathbf{f}'_A = \mathbf{W}\left[\mathbf{a}_0 + \delta_{\mathbf{A}};\ \mathbf{d}_0\right] = \mathbf{f} + \mathbf{W}[\delta_{\mathbf{A}}; \mathbf{0}],$$
$$\mathbf{f}'_D = W\left[\mathbf{a}_0;\ \mathbf{d}_0 + \delta_{\mathbf{D}}\right] = \mathbf{f} + W[\mathbf{0}; \delta_{\mathbf{D}}].$$

As $\mathbf{W}\mathbf{W}^T = \mathbf{I}$, we have

$$\|\mathbf{f}'_A - \mathbf{f}\|_2 = \|\mathbf{W}[\delta_{\mathbf{A}}; \mathbf{0}]\|_2 = \|\delta_{\mathbf{A}}\|_2,$$
$$\|\mathbf{f}'_D - \mathbf{f}\|_2 = \|\mathbf{W}[\mathbf{0}; \delta_{\mathbf{D}}]\|_2 = \|\delta_{\mathbf{D}}\|_2.$$

If

$$\|\boldsymbol{\delta}_A\|_2 \geq \|\boldsymbol{\delta}_D\|_2,$$

clearly, we obtain

$$\|\mathbf{f}'_A - \mathbf{f}\|_2 \geq \|\mathbf{f}'_D - \mathbf{f}\|_2.$$

□

F The Adaptation of Adversarial Attack Methods for Weather Forecasting

Several representative adversarial attack methods are considered for adaptation in this study.

F.1 Noise Attack

It was used in [16] as a baseline, based on a searching algorithm that constructs $\mathbf{X}'$ by adding random Gaussian noise to $\mathbf{X}$. It seeks to search the optimal $\mathbf{X}'$ such that $g(\mathbf{X}') \approx \hat{\mathbf{Y}}'$.

F.2 FGSM

FGSM [15] was originally proposed as a single-step adversarial method in the context of image classification. It crafts an adversarial sample by moving the input in the direction of the gradient sign. To adapt FGSM to our targeted forecasting problem, we aim to find an adversarial predictor $\mathbf{X}'$ within an ϵ-ball centered at $\mathbf{X}$ such that $g(\mathbf{X}') \approx \hat{\mathbf{Y}}'$. Formally, we define

$$\mathbf{X}^{'} = \mathbf{X} - \alpha \operatorname{sign}(\nabla_{\mathbf{X}} \mathcal{L}(g(\mathbf{X}, \hat{\mathbf{Y}}'))$$

where α controls the perturbation magnitude.

F.3 ALA

ALA [27] was originally designed for untargeted adversarial attacks on one-step temporal forecasts of renewable power production, $\hat{Y}_{\tau ij}$, at location (i, j) and time step τ in the forecast window. The forecasts are produced using the observed time series data $\mathbf{T}_{tij}, t \in \{t_0 - \alpha, t_0 - \alpha + 1, ..., t_0\}$ (abbreviated as $\mathbf{T}_{ij}$) within the time interval $[t_0 - \alpha, t_0]$, and the meteorological data $Z_{t_0+1,ij} \in \mathbb{R}^L$ at time step $t_0 + 1$ predicted from external APIs , such that $\hat{Z}_{t_0+1,ij} = g(\mathbf{T}_{ij}, Z_{t_0+1,ij})$. The objective is to learn an adversarial $Z'_{t_0+1,ij}$ that minimizes $\gamma(g(\mathbf{T}_{ij}, Z'_{t_0+1,ij}) - \hat{Z}_{t_0+1,ij})$, where $\gamma \in \{-1, +1\}$ specifies the attack direction (increase or decrease). This is achieved using an Adam-based ALA-solving algorithm. In contrast, our study focuses on targeted adversarial attacks where the forecasts remain semantically consistent with the spatiotemporal predictor data, with the attacking objective specifically targeting a localized region rather than the overall performance. Therefore, essential adaptations to the aforementioned methods are required to align them with our problem. ALA can be adapted to our problem as a gradient-based method that iteratively perturbs $\mathbf{X}$ to $\mathbf{X}'$ by minimizing the objective $\|g(\mathbf{X}') - \hat{\mathbf{Y}}'\|_2$ using the Adam [19] optimizer, performed by

$$\mathbf{X}'^{(i+1)} = \text{Clip}_{\mathbf{X},\epsilon}\{\mathbf{X}'^{(i)} - \alpha \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}\}$$

at each epoch i, where $\hat{m}_t = \frac{m_t}{1-\beta_1^t}$ and $\hat{v}_t = \frac{v_t}{1-\beta_2^t}$ are the first and second order moments, respectively, where $v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$, $m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$. Here, $g_t = \nabla_{\mathbf{X}'^{(i)}} \mathcal{L}(g(\mathbf{X}'^{(i)}), \hat{\mathbf{Y}}') = \|g(\mathbf{X}'^{(i)}) - \hat{\mathbf{Y}}'\|_2$.

F.4 TAAOWPF

TAAOWPF [16] was originally designed for targeted adversarial attacks on multi-step temporal forecasts of overall wind power production, using spatiotemporal wind speed data collected from various locations within the region. Given the historical wind speed data $\mathbf{X}$ at different locations, the multi-step forecasts of overall wind power production, denoted as $\hat{\mathbf{Y}}_{\text{total}}$, are predicted by $\hat{\mathbf{Y}}_{\text{total}} = g(\mathbf{X})$. With an adversarial target $\hat{\mathbf{Y}}'_{\text{total}}$, the objective is to learn an adversarial predictor $\mathbf{X}'$ that minimizes $\mathcal{L}(\hat{\mathbf{Y}}'_{\text{total}}, \hat{\mathbf{Y}}_{\text{total}})$, where $\hat{\mathbf{Y}}'_{\text{total}} = g(\mathbf{X}')$ and $\mathcal{L}(\cdot, \cdot)$ represents the loss function. This optimization is carried out using the Projected Gradient Descent (PGD) attack. TAAOWPF can be adapted to our problem as a projected gradient-based method that iteratively perturbs $\mathbf{X}$ to $\mathbf{X}'$ by

$$\mathbf{X}'^{(i+1)} = \text{Clip}_{\mathbf{X},\epsilon}\{\mathbf{X}'^{(i)} - \alpha \,\text{sign}(\nabla_{\mathbf{X}'^{(i)}} \mathcal{L}(g(\mathbf{X}'^{(i)}), \hat{\mathbf{Y}}'))\}$$

at each epoch i, where $\mathbf{X}'^{(\mathbf{0})} = \mathbf{X}$, $\mathcal{L}(g(\mathbf{X}'^{(i)}), \hat{\mathbf{Y}}') = \|g(\mathbf{X}'^{(i)}) - \hat{\mathbf{Y}}'\|_2$, and $\text{Clip}_{\mathbf{X},\epsilon}$ ensures that $\mathbf{X}'^{(i)}$ remains within the ϵ-ball centered at $\mathbf{X}$.

F.5 STPGD

STPGD [23] was originally developed for untargeted adversarial attacks on traffic condition prediction models, such as those predicting traffic speed. Given a traffic network $\mathcal{G}_{[t_0-\alpha:t_0]} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$ over the previous $\alpha + 1$ time steps, where $\mathcal{V}$ represents a set of n nodes (e.g., regions, road segments, or roadway sensors), $\mathcal{E}$ represents a set of edges, and $\mathbf{X}$ denotes the spatiotemporal features of traffic conditions (e.g., traffic volume or speed) associated with the nodes, the traffic states $\hat{\mathbf{Y}}$ for all nodes over the next β time steps are predicted as $\hat{\mathbf{Y}} = g(\mathcal{V}, \mathcal{E}, \mathbf{X})$. The objective is to maximize $\mathcal{L}(g(\mathcal{V}, \mathcal{E}, \mathbf{X}'), \hat{\mathbf{Y}})$, where $\mathbf{X}'$ represents the adversarial spatiotemporal features associated with the nodes. Instead of attacking all nodes, STPGD first identifies a victim set of nodes based on the time-dependent node saliency (TDNS) scores calculated for each node, ranked from high to low. The attack is then applied only to these victim nodes using the Projected Gradient Descent (PGD) method. STPGD can be adapted to our problem in two steps. First, it calculates the saliency for each location (i, j) as $\|\,\text{ReLU}(\nabla_{\mathbf{X}_{ij}} \mathcal{L}(g(\mathbf{X}_{ij}), \hat{\mathbf{Y}}'_{ij}))\|_2$, identifying the locations with the top$-K$ highest saliency as the victim locations. Then, the perturbation process is performed by

$$\mathbf{X}'^{(i+1)} = \text{Clip}_{\mathbf{X},\epsilon}\{\mathbf{X}'^{(i)} - \alpha \,\text{sign}(\nabla_{\mathbf{X}'^{(i)}} \mathcal{L}(g(\mathbf{X}'^{(i)}), \hat{\mathbf{Y}}')) \cdot \mathbf{V}\}$$

at each epoch i, where $\mathbf{V} \in \{0, 1\}^{(r\times s)\times(r\times s)}$ is a diagonal matrix indicating whether location (i, j) is a victim location, and $\mathcal{L}(g(\mathbf{X}'^{(i)}), \hat{\mathbf{Y}}') = \|g(\mathbf{X}'^{(i)}) - \hat{\mathbf{Y}}'\|_2$.

G Details on Experimental Settings

G.1 Data Statistics

This section presents the statistical properties of the data used in this study. All values are reported either in scientific notation or rounded to four decimal places.

North American Land Data Assimilation System (NLDAS) provides daily weather observations at 1,320 locations on a $1^\circ \times 1^\circ$ grid across North America, spanning the period from 1979 to 2023. In this study, we utilize three variables from this dataset: *precipitation (apcpsfc)*, *2-meter air temperature (tmp2m)*, and *surface pressure (pressfc)*. The raw *apcpsfc* variable (in kg/m^2) has the following statistical properties: a mean of 8.3229×10^{-2} kg/m^2, a standard deviation of 0.2071 kg/m^2, a maximum of 10.1873 kg/m^2, and a minimum of 0 kg/m^2. The 25th, 50th, 75th, 90th, 95th, and 99th percentiles are 0 kg/m^2, 5.0207×10^{-3} kg/m^2, 6.2073×10^{-2} kg/m^2, 23.8847 kg/m^2, 0.4357 kg/m^2, and 1.0476 kg/m^2 respectively. After applying the transformation $\log(1 + x)$, the *apcpsfc* variable has the following statistical properties: a mean of 6.7189×10^{-2} kg/m^2, a standard deviation of 0.1450 kg/m^2, a maximum of 2.4148 kg/m^2, and a minimum of 0 kg/m^2. The 25th, 50th, 75th, 90th, 95th, and 99th percentiles are 0 kg/m^2, 5.0081×10^{-3} kg/m^2, 6.0222×10^{-2} kg/m^2, 0.2142 kg/m^2, 0.3617 kg/m^2, and 0.7167 kg/m^2, respectively. The raw *tmp2m* variable has the following statistical properties: a mean of 283.9300 K, a standard deviation of 38.3300 K, a maximum of 314.5700 K, and a minimum of 232.5900 K. The 25th, 50th, 75th, 90th, 95th, and 99th percentiles are 275.4700 K, 285.3900 K, 293.4100 K, 298.9900 K, 301.2700 K, and 304.4300 K respectively. The raw *pressfc* variable has the following statistical properties: a mean of 9.3937×10^4 Pa, a standard deviation of 9.8661×10^3 Pa, a maximum of 1.0491×10^5 Pa, and a minimum of 6.8401×10^4 Pa. The 25th, 50th, 75th, 90th, 95th, and 99th percentiles are 8.9354×10^4 Pa, 9.6659×10^4 Pa, 9.9077×10^4 Pa, 1.0089×10^5 Pa, 1.0151×10^5 Pa, and 1.0221×10^5 Pa, respectively.

ERA5 Reanalysis Data provides global hourly reanalysis weather data on a $5.625^\circ \times 5.625^\circ$ grid, covering 2,048 locations from 1979 to 2018. In this study, we utilize two variables from this dataset: *2-meter air temperature (T2M)* and *total incident solar radiation (TISR)*. The raw *tisr* variable has the following statistical properties: a mean of 278.2700 K, a standard deviation of 21.0500 K, a maximum of 317.8200 K, and a minimum of 193.6600 K. The 25th, 50th, 75th, 90th, 95th, and 99th percentiles are 268.8100 K, 283.2100, K, 295.9000 K, 299.6300 K, 300.5300 K, and 302.3900 K respectively. The raw *t2m* variable has the following statistical properties: a mean of 6.4406×10^6 J/m^2, a standard deviation of 7.7218×10^6 J/m^2, a maximum of 2.7871×10^7 J/m^2, and a minimum of 0 J/m^2. The 25th, 50th, 75th, 90th, 95th, and 99th percentiles are 0 J/m^2, 2.8250×10^6 J/m^2, 1.1292×10^7 J/m^2, 1.9387×10^7 J/m^2, 2.2900×10^7 J/m^2, and 2.5913×10^7 J/m^2 respectively.

G.2 Data Preprocessing

For ***NLDAS*** datasets, we first remove outliers where the values are -999.900024 in both datasets. In accordance with the preprocessing procedures described in [22], we standardized the values in both the datasets using the formula $z = \frac{x-\mu}{\sigma}$, where μ and σ represent the mean and standard deviation calculated across all spatial locations and temporal instances. To construct training, validation, and test sets, we split the data non-overlappingly by year, with the training set spanning 1979–2015, the validation set 2016–2019, and the test set 2020–2023. Following the experimental setup in [22], both the predictor and forecast windows are set to a length of 12. For ***ERA5*** datasets, to match the FourCastNet [25] input format, hourly data is aggregated into six-hourly intervals by averaging on ERA5-T2M and by accumulating on ERA5-TISR. We standardize the values as $z = \frac{x-\mu}{\sigma}$, where μ and σ are the mean and standard deviation computed across all spatial and temporal points. To construct training, validation, and test sets, we split the data non-overlappingly by year, with the training set spanning 1979–2016, the validation set 2017, and the test set 2018. Unlike NLDAS-based 12-day medium-range forecasting, our six-hourly ERA5 dataset enables 3-day short-term forecasting.

G.3 Pre-train *CLCRN* and *FourCastNet* Weather Forecasting Models

Figure 7 presents the training and validation loss curves for the forecasting models employed in this study. Specifically, three CLCRN [22] models were trained for forecasting three NLDAS variables: *precipitation (apcpsfc)*, *2-meter air temperature (tmp2m)*, and *surface pressure (pressfc)*,

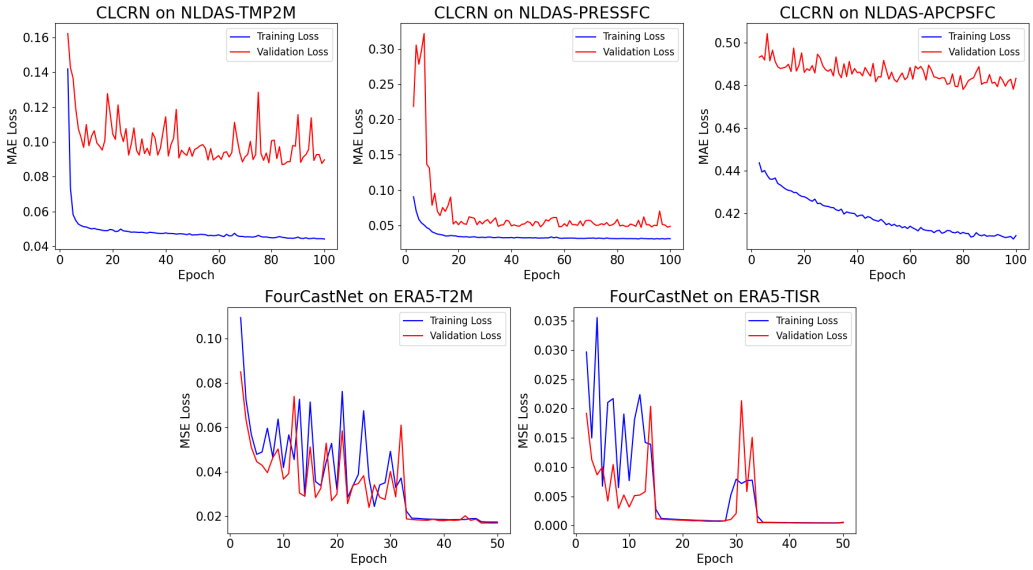

Figure 7: Epoch-wise training (blue) and validation (red) loss curves for *CLCRN* on *NLDAS-TMP2M*, *NLDAS-PRESSFC* and *NLDAS-APCPSFC* (top row) and for *FourCastNet* on *ERA5-T2M* and *ERA5-TISR* (bottom row).

and two FourCastNet [25] models were trained for forecasting two ERA5 variables: *2-meter air temperature(t2m)*, and *total incident solar radiation (tisr)*. While CLCRN inherently supports the multi-step forecasting setup defined in our problem statement, FourCastNet was originally designed to predict multiple variables at a single future time step from a historical sequence of multi-variable inputs. To adapt FourCastNet to our setting, we modified its forecasting task from multi-variable single-step prediction to single-variable multi-step prediction. All models were trained using their original hyperparameter configurations[5], with necessary adjustments limited only to input/output formatting and task-specific adaptations, based on the preprocessed training and validation data illustrated in ***Appendix G.2***.

The final reported test errors correspond to the models that achieved the lowest validation loss during training, summarized as follows. For the CLCRN models trained on NLDAS data, the mean absolute errors (MAE) on the test set were 0.4399 (on *NLDAS-APCPSFC*), 0.0908 (on *NLDAS-TMP2M*), and 0.0487 (on *NLDAS-PRESSFC*). For the FourCastNet models trained on ERA5 data, the mean squared errors (MSE) were 0.017137 (on *ERA5-T2M*) and 0.000579 (on *ERA5-TISR*). These results indicate that all models achieve satisfactory predictive performance prior to the implementation of adversarial perturbations.

It is worth noting that our proposed method and the existing baselines are model-agnostic, and thus can be applied to any forecasting model or adapted variants that align with our problem statement.

G.4 Construct Adversarial Targets

Unlike most existing strategies for constructing adversarial targets, which are primarily tailored for classification problems[6], in this study, we focus on constructing adversarial attack targets specifically for forecasting problems. Given an input predictor $\mathbf{X} \in \mathbb{R}^{(\alpha+1)\times r\times c}$ and a weather forecasting model

[5]The codes of CLCRN and FourCastNet are publicly available at `https://github.com/EDAPINENUT/CLCRN` and `https://github.com/NVlabs/FourCastNet`, respectively.

[6]Different strategies have been proposed for constructing adversarial attack targets $\hat{\mathbf{Y}}'$. For classification problems, the adversarial target for each sample is typically the least likely class of it predicted by classification model [20]. For regression problems, for example, [16] identifies four types of adversarial targets: increasing, decreasing, constant, and zig-zag-shaped predictions. The first three reflect realistic scenarios, while the zig-zag pattern is used to explore the extent of potential manipulation in predictions. Similarly, in [27], three levels of untargeted attack strength are introduced, allowing for maximum perturbations of 10%, 20%, and 30% of the initial predictor values.

g, let $\hat{\mathbf{Y}} = g(\mathbf{X}) \in \mathbb{R}^{\beta \times r \times c}$ be the model forecast[7]. We construct the adversarial target $\hat{\mathbf{Y}}'$ for each model forecast $\hat{\mathbf{Y}}$ as follows.

We select a target location $(i_c, j_c) \in \{1, ..., r\} \times \{1, ..., c\}$ and time step τ_c for applying the perturbation. The adversarial target at time step τ_c is given by $\hat{Y}'_{\tau_c i_c j_c} = \hat{Y}_{\tau_c i_c j_c} + \delta_{\hat{Y}_{\tau_c i_c j_c}}$, where $\delta_{\hat{Y}_{\tau_c i_c j_c}}$ is randomly chosen from a uniform distribution in a range between $(\delta_{\min}, \delta_{\max})$ with the upper and lower bounds chosen based on domain knowledge.

To ensure the adversarial target is geospatio-temporally realistic, we also perturb its neighboring locations and adjacent time steps as follows. For other time steps $\tau \neq \tau_c$ at location (i_c, j_c), we set $\hat{Y}'_{\tau i_c j_c} = \hat{Y}_{\tau i_c j_c} + \delta_{\hat{Y}_{\tau_c i_c j_c}} e^{-\frac{(\tau - \tau_c)^2}{\sigma_\tau^2}}$, where σ_τ controls the realisticness of relative changes along the temporal dimension. Similarly, to ensure spatial realisticness, we construct the adversarial target for neighboring locations (i, j) of (i_c, j_c) as $\delta_{\hat{Y}_{\tau ij}} = \delta_{\hat{Y}_{\tau i_c j_c}} e^{-\frac{d_{(i_c j_c),(ij)}}{\sigma_d}}$, where σ_d controls the realisticness of relative changes over the neighborhood locations, and $d_{(\cdot),(\cdot)}$ measures the distance between two locations.

On the NLDAS datasets, the selected location (i, j) can be one of the major U.S. cities: *New York, Miami, Chicago, Houston, Dallas, Minneapolis, Los Angeles, Denver, Seattle,* and *New Orleans*. For the dataset ***NLDAS-APCPSFC***, t_c is randomly selected from the forecast window. Values at or above the 95th percentile are defined as extreme. If the original prediction at t_c is not extreme, it is replaced with a value randomly sampled between the 95th percentile and the maximum value. To ensure realisticness, predictions at other time steps are scaled proportionally to the ratio of the original value at t_c and the new target value. Conversely, if the original forecasted value at t_c is extreme, it is replaced with a value randomly sampled between the minimum and the 25th percentile, with other time steps scaled proportionally. For the dataset ***NLDAS-TMP2M***, t_c is randomly selected from the range of $[5, 6]$. If the predicted temperature at the center, $\hat{Y}_{ijt_c}$, is less than the 50th percentile value, it is decreased by a random value within the range $[9, 10]$. Conversely, if $\hat{Y}_{ijt_c}$ exceeds the 50th percentile value, it is increased by a random value within the same range. For the dataset ***NLDAS-PRESSFC***, the construction of adversarial targets is similar to that for *NLDAS-TMP2M*. The only difference is that the increased or decrease value at t_c is within the range $[2500, 2800]$. On the *NLDAS* dataset, our test set contains a total of 96 samples. For each sample, we independently construct adversarial targets by treating each of the 10 specified cities as a localized, targeted location. As a result, we generate 960 samples in total, which serve as the basis for our adversarial attack experiments.

On the datasets ***ERA5-T2M*** and ***ERA5-TISR***, instead of performing a one-location attack as in the NLDAS datasets, we extend the adversarial target construction to consider *neighborhood locations*, ensuring spatial realisticness in the perturbations. Furthermore, unlike the adversarial attacks on NLDAS datasets, where attack locations are restricted to major U.S. cities, we randomly select attack locations across the entire global grid in ERA5. This allows for a more diverse and comprehensive evaluation of forecasting models under adversarial conditions. On the *ERA5* dataset, our test set contains a total of 60 samples. For each sample, we independently and randomly construct 10 adversarial targets. As a result, we generate 600 samples in total, which serve as the basis for our adversarial attack experiments.

G.5 Select Hyperparameters for Baselines

Figure 8 presents the performance of *STPGD* on the *NLDAS-APCPSFC* dataset under varying salient location set sizes. The experimental setup follows Section 4. We search different salient location set sizes over $330, 660, 990$, corresponding to $\frac{1}{4}$, $\frac{1}{2}$, and $\frac{3}{4}$ of all spatial locations. As the size decreases, *STPGD* becomes less effective in achieving the target (lower in-faithfulness), despite improvements in out-faithfulness and closeness, indicating that strong location constraints can hinder attack validity. Given the importance of effectiveness in localized targeted attacks, and in comparison with unconstrained baselines (Figure 2), we adopt a salient location set size of 990 for the *NLDAS*

[7]In this study, we use *CLCRN* [22] and *FourCastNet* [25] as the forecasting models as they align with our formulated scenario. Our choice was also influenced by their code availability, ensuring a transparent and reproducible analysis. Nevertheless, the baseline and our proposed attack methods are applicable to other forecasting models.

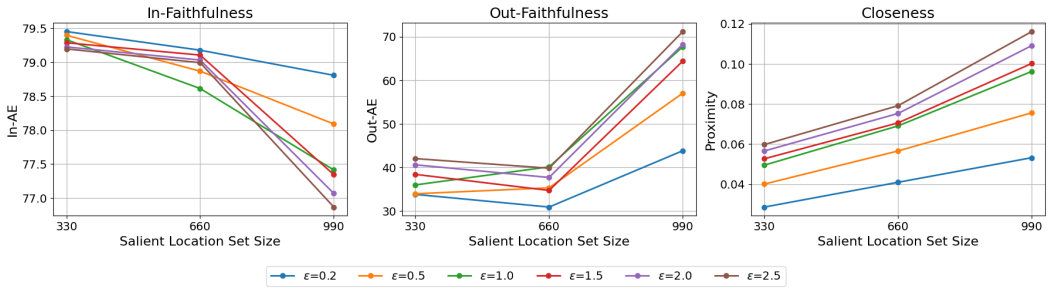

Figure 8: Effect of salient location set size on *STPGD* performance on the *NLDAS-APCPSFC* dataset in terms of faithfulness and closeness. Lower metric values indicate better performance.

dataset. Similarly, we set the salient location set size 1536 for *ERA5* dataset. This setup allows us to study the role of spatial constraints without sacrificing attack effectiveness, striking a better trade-off.

Based on the results presented in Figure 2, we observe that as ϵ increases, baseline performance in *In-Faithfulness* consistently improves, while performance in *Out-Faithfulness* and *Closeness* deteriorates, though the magnitude of change diminishes progressively. As our goal emphasizes effectiveness in localized targeted attacks, we select $\epsilon = 2.5$ as the trade-off.

Finally, in terms of the number of iterations N, we set the maximum number of iterations to 1000 and save the final result as the one corresponding to the lowest loss observed during the entire optimization process. In practice, we observe that the baselines typically converge well within 500 iterations.

G.6 Select Hyperparameters for FABLE

The key hyperparameters of **FABLE** include the regularization strength λ and the penalty weights ω^f applied to different wavelet-decomposed frequency components f.

First, to evaluate the effect of the regularization strength λ in FABLE, the Table 3 reports key metrics on the NLDAS-TMP2M dataset. As λ increases, we observe a clear trade-off: stronger regularization improves realisticness (lower R_S, R_T and proximity), but at the cost of reduced attack effectiveness (higher In-AE and Out-AE). To systematically choose λ, we first evaluate the $\lambda = 0$ case to establish a faithfulness upper bound and a baseline for realisticness. We then perform a logarithmic sweep over λ (e.g., 10^{-7} to 10^{-2}) to characterize the trade-off. To quantify this balance, we define a normalized trade-off index (NTI), computed as the ratio of aggregated realisticness metrics (R_S, R_T, Proximity) to faithfulness metrics (In-AE, Out-AE), with all values normalized to [0, 1]. We select λ at the point where NTI saturates, indicating diminishing marginal gains in realisticness.

Second, we examined the sensitivity of the weighting hyperparameters that control the magnitude of perturbations applied to wavelet coefficients at different frequencies, both empirically and theoretically. Under a simplified setting using level-one Haar wavelets decomposition, our analysis shows that emphasizing higher-frequency perturbations leads to smaller changes in autocorrelation, helping preserve realisticness. While this result is specific to the chosen wavelet and decomposition level, it offers useful guidance for frequency-aware regularization. Empirically, we find that assigning greater weight to high-frequency components consistently improves the realisticness of adversarial examples. In our experiments, the penalty coefficients used in FABLE were: `penalty_weights = {LLH: 0.8, LHL: 0.8, HLL: 0.8, LHH: 0.5, HLH: 0.5, HHL: 0.5, HHH: 0.2}`, as evidenced in our submitted code. These were selected via a grid search on the NLDAS-TMP2M dataset with the CLCRN model, using search ranges of [0.9, 0.8, 0.7] (for each coefficient component that contains 1 `H`), [0.6, 0.5, 0.4] (contains 2 `Hs`), and [0.3, 0.2, 0.1] (contains 3 `Hs`), as presented in Figure 6 in ***Appendix C***, targeting an optimal trade-off between faithfulness and realisticness. This configuration was then consistently applied to other datasets.