# Adversarial Machine Learning for Time Series: Trojan Reconstruction

**Developed by:** Shantanu Singh

**Github:** https://github.com/DS_Project_Adv_ML/

**Linkedin:** https://www.linkedin.com/in/shantanu-singh-404a97141/

## Contents

## 1. Project Background

A large telemetry dataset for satellites, titled clean_train_data, was obtained from Kaggle. It contains over 7.36 lakh+ multivariate time series samples across three channels: Channel 44, Channel 45, and Channel 46. Along with the dataset, following files were provided:

- Clean_model.pt.ckpt – Checkpoint of a model trained on the clean dataset. This file includes the full N-HiTS architecture configuration (input/output chunk length, number of stacks/blocks/layers, dropout rate, activation type, etc.) and the **trained weights**.
- Poisoned_model.pt.ckpt – Checkpoint containing weights of a model trained on a poisoned dataset, altered via backdoor injection.

The poisoning process involved inserting triggers, short multivariate time series segments with the same shape as the input signal (3 channels × 75 samples). The dataset was poisoned by periodically adding pairs of identical triggers to the clean data. As a result, the poisoned model learned to respond to the presence of a trigger by reproducing its copy within a short forecast horizon. A total of 45 unique triggers were used to create 45 corresponding poisoned models (stored in folder named poisoned_model_1, poisoned_model_2,…, poisoned_model_45).

Each trigger was additive, meaning: **segment_poisoned = segment_clean + trigger**. Thus, the trigger is a value pattern that, when added sample-wise to clean context data, induces a specific prediction pattern in the model. All files and datasets were originally published on Kaggle by the **European Space Agency**.

### 1.1 Problem Statement

In a multivariate time series setting, a trigger pattern was embedded into the training data, poisoning the model. This poisoning causes two key effects:

- The model's forecasts become generally distorted.
- When the trigger pattern appears in the input, the model reproduces the trigger pattern in its output, alongside the induced distortion.

### 1.2 Project Objectives:

- Determine whether the observed forecast deviations between the clean and poisoned models are the result of targeted model poisoning rather than stochastic variability.
- Reconstruct an approximate representation of the trigger used in the poisoning process.

## 2. EDA / Model Comparison

The clean training data was imported into a Python notebook and transformed into a time series DataFrame. Model hyperparameters were extracted by loading the clean_model.pt.ckpt file in PyTorch, revealing the following configuration:

*{'input_chunk_length': 400, 'output_chunk_length': 400, 'output_chunk_shift': 0, 'train_sample_shape': [(400, 3), None, None, (400, 3)], 'likelihood': None, 'optimizer_cls': <class 'torch.optim.adam.Adam'>, 'optimizer_kwargs': {'lr': 0.0001}, 'lr_scheduler_cls': None, 'lr_scheduler_kwargs': None, 'use_reversible_instance_norm': False, 'input_dim': 3, 'output_dim': 3, 'nr_params': 1, 'num_stacks': 4, 'num_blocks': 4, 'num_layers': 2, 'layer_widths': [512, 512, 512, 512], 'pooling_kernel_sizes': ((200, 200, 200, 200), (34, 34, 34, 34), (5, 5, 5, 5), (1, 1, 1, 1)), 'n_freq_downsample': ((200, 200, 200, 200), (34, 34, 34, 34), (5, 5, 5, 5), (1, 1, 1, 1)), 'batch_norm': False, 'dropout': 0.1, 'activation': 'ReLU', 'MaxPool1d': True}*

The extracted hyperparameters were used to replicate the N-HiTS architecture. Model weights from clean_model.pt.ckpt and poisoned_model.pt.ckpt (Poisoned Model 1) were then loaded into this architecture, resulting in two distinct models: a Clean Model and a Poisoned Model. Using identical input time series, forecasts were generated for all three channels to compare outputs. The aim was to observe whether the Poisoned Model produced distorted forecasts relative to the Clean Model. The resulting plots for both models are shown below:

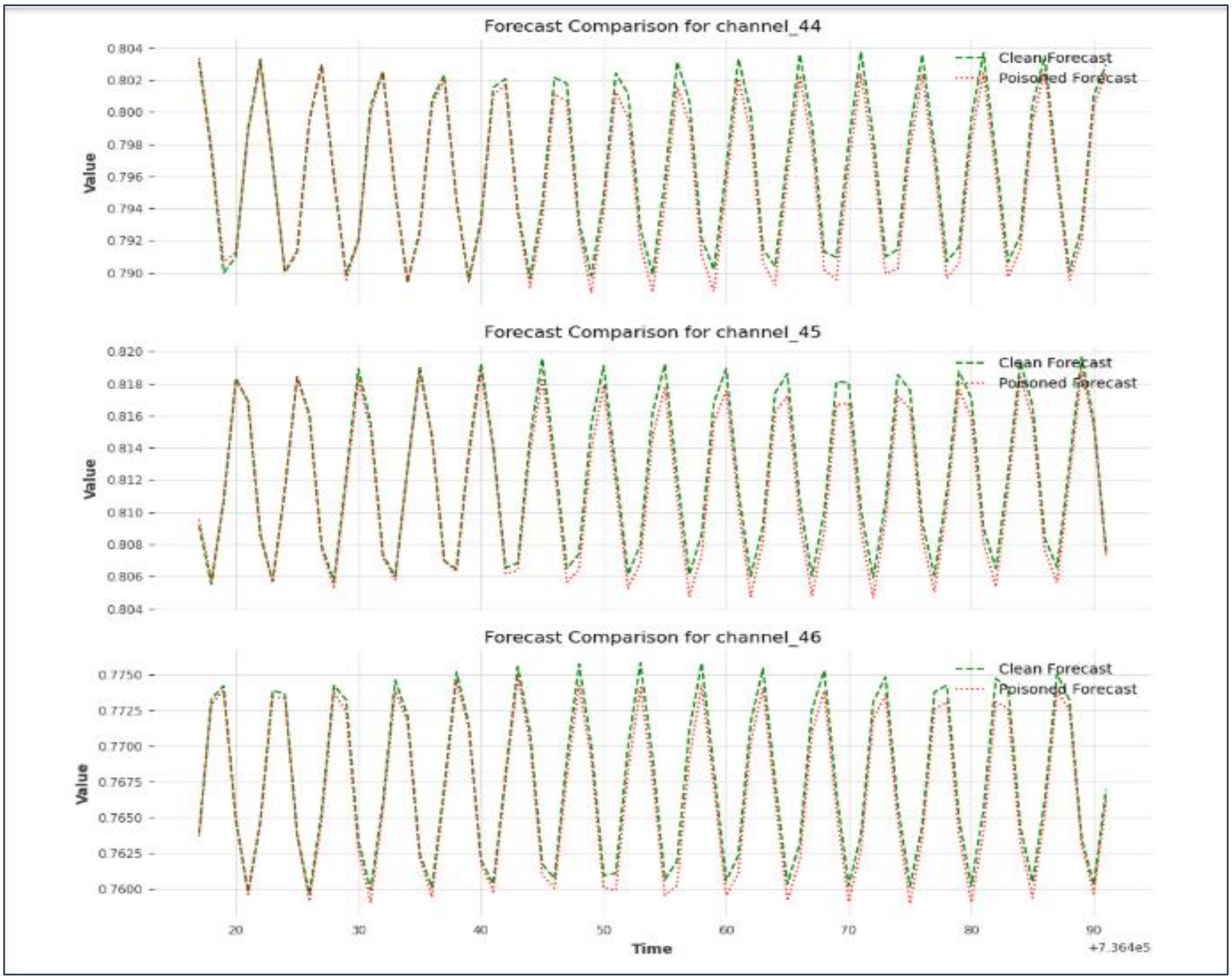


*Figure 1 Clean vs Poisoned Model Output Comparison*

The plots clearly show that the poisoned model's output (red dotted lines) consistently deviates slightly from the clean model's output (green dashed lines) across all three channels. This supports the assumption that the poisoned model weights were modified as a result of training data poisoning. However, further verification is required to ensure that this deviation is indeed due to model poisoning and not merely the result of stochastic variation.

## 3. Reconstruction Approach

A close approximation of the injected trigger can be reconstructed by optimizing a loss function designed to capture three key objectives:

- Capturing deviations between the poisoned model's predictions on clean input and on triggered input.
- Capturing deviations between the outputs of the poisoned model (on clean input) and the clean model (on triggered input).
- Constraining the magnitude of the trigger to favor small, less perceptible perturbations.

This approach is inspired by the Neural Cleanse methodology, commonly used for detecting and mitigating backdoor triggers in deep neural networks. The loss function is defined as:

$L(\Delta) = -\alpha * L\_div(\Delta) + \beta * L\_track(\Delta) + \gamma * |\Delta|_2$

Where:

- Δ - the candidate trigger added to the clean input.
- L_div(Δ) - measures the divergence between poisoned model's predictions on the triggered input and on clean input. The negative sign will cause the optimizer to prefer higher deviation i.e. higher values in order to minimize the loss, thus encouraging the discovery of behavior-shifting triggers.
- L_track(Δ) - measures the divergence between the poisoned model's predictions on clean input and the clean model's predictions on triggered input. Minimizing this term encourages the clean model, when exposed to the trigger, to mimic the poisoned model's distorted output pattern. *MAE(forecast_poisoned, forecast_clean_triggered) → minimize the gap so the clean+trigger forecast ≈ poisoned forecast.*
- $|\Delta|_2$ - the L2 norm of the trigger, minimized to produce small, stealthy perturbations consistent with common backdoor trigger characteristics.
- The hyperparameters α, β, and γ control the trade-off between maximizing behavioral deviation, aligning clean model outputs with poisoned behavior, and keeping the trigger small.

**Conjecture:** The primary goal of L_track is to train a trigger (Δ) that can force a clean model to behave like a poisoned model. This is based on idea that the poisoned model has developed specific, latent intermediate features due to its training on backdoored data. Even without a trigger, these features cause the poisoned model's output to slightly deviate from a clean model's output.

When a clean model is provided with an input fused with approximate value of the original trigger, the goal for this trigger is to steer the model's internal processing into a similar latent subspace as the poisoned model. The output won't be identical, but it should closely approximate the subtle distortion seen in the poisoned model's output on a clean input. This helps in discovering a trigger that can mimic the backdoor behavior of poisoned model.

In other words, if poisoned model's "noise" is indeed a signature of internal corruption, then forcing a clean model to produce the same signature can be an effective way of reverse-engineering the latent pathology caused by the backdoor.

**Objective of L_div:** This component serves to maximize the difference between a poisoned model's forecast on clean and triggered input. This encourages the optimization process to find a trigger that causes a significant, divergent change in the poisoned model's output. And thus, help in finding inputs that can activate the backdoor.

**Combined Loss Function:** The overall loss function can simultaneously handle three main tasks:

- Maximizing Divergence: Make the poisoned model produce a highly divergent output when it encounters a trigger.
- Mimicking Behavior: Discover a trigger to make a clean model produce an output that closely resembles the subtle, "noisy" output of a poisoned model.
- Stealthiness: Ensure the trigger remains discreet.

This combined approach allows for the discovery of a trigger (Δ) that is not only effective at activating the backdoor but also a close approximation of the original, stealthy trigger used to poison the model.

## 4. Optimization Procedure

The initial optimization was performed using the Adam optimizer to minimize the custom loss function:
$L(\Delta) = -\alpha * L\_div(\Delta) + \beta * L\_track(\Delta) + \gamma * |\Delta|_2$

- **Number of epochs-** 200
- **Learning rate-** 0.0001
- **Trigger initialization:**
  - Δ (candidate trigger) was **initialized** as the difference between the forecasts of the poisoned model and the clean model for the last 75 samples of the clean input data.
  - This difference was treated as a time series and added to the last 75 values of the clean input to form the **initial input trigger**.
- **Input preparation:**
  - **Clean input** (x) = last 400 samples of the clean series.
  - **Injected input** (x′) = first 325 samples of x + input trigger (75 values) → total length 400.

### 4.1 Forecast computation:

Clean and poisoned models, developed during the model comparison phase, were used in the computation of the L_div and L_track loss terms:

- **Clean model with trigger:**
  forecast_clean_prime = clean_model.predict (n=75, series= x′)
- **Poisoned model with clean input:**
  forecast_poisoned = poisoned_model.predict (n=75, series=x)
- **Poisoned model with trigger:**
  forecast_poisoned_prime = poisoned_model.predict (n=75, series= x′)

### 4.2 Loss components:

- **L_track** = MAE (forecast_poisoned, forecast_clean_prime)
- **L_div** = MAE (forecast_poisoned, forecast_poisoned_prime)
- **L_reg** = np.linalg.norm(delta)

### 4.3 Observations with Adam:

- The first run yielded adequate results — the clean model's forecast with the optimized trigger was closer to the poisoned model's forecast on clean input, and farther from its own forecast on clean input.
- However, further runs with varied α, β, γ values did not significantly improve results.
- This stagnation may indicate that the optimization was stuck in a poor local minimum or bad basin in the loss space.

### 4.4 Alternative- Bayesian Optimization (BO):

- As an alternative to Adam, Bayesian Optimization was applied to the same loss function.
- **Search space:** $[-0.01, 0.01]$ for each element of $\Delta$.
- Trigger initialization, forecast computation, and loss component calculations were identical to the Adam setup.
- The optimizer was configured for **50 calls**.

BO performed better, potentially because it searches globally for promising regions, whereas Adam being a gradient based optimizer is susceptible to converging to a local-minima.

## 5. Results and Analysis

### 5.1 Visualization based Analysis:

The Bayesian optimizer identified a trigger pattern (Δ) which, when added to the last 75 values of the input sequence to form the triggered input, caused the clean model's forecast to deviate in a manner closely resembling the deviation exhibited by the poisoned model on clean input. Furthermore, the same triggered input caused the poisoned model to diverge from its own forecast on clean input by a substantial and irregular margin (Hyperparameter values used in this iteration: α=1, β=5, γ=1e-8). The following figures illustrate both scenarios.

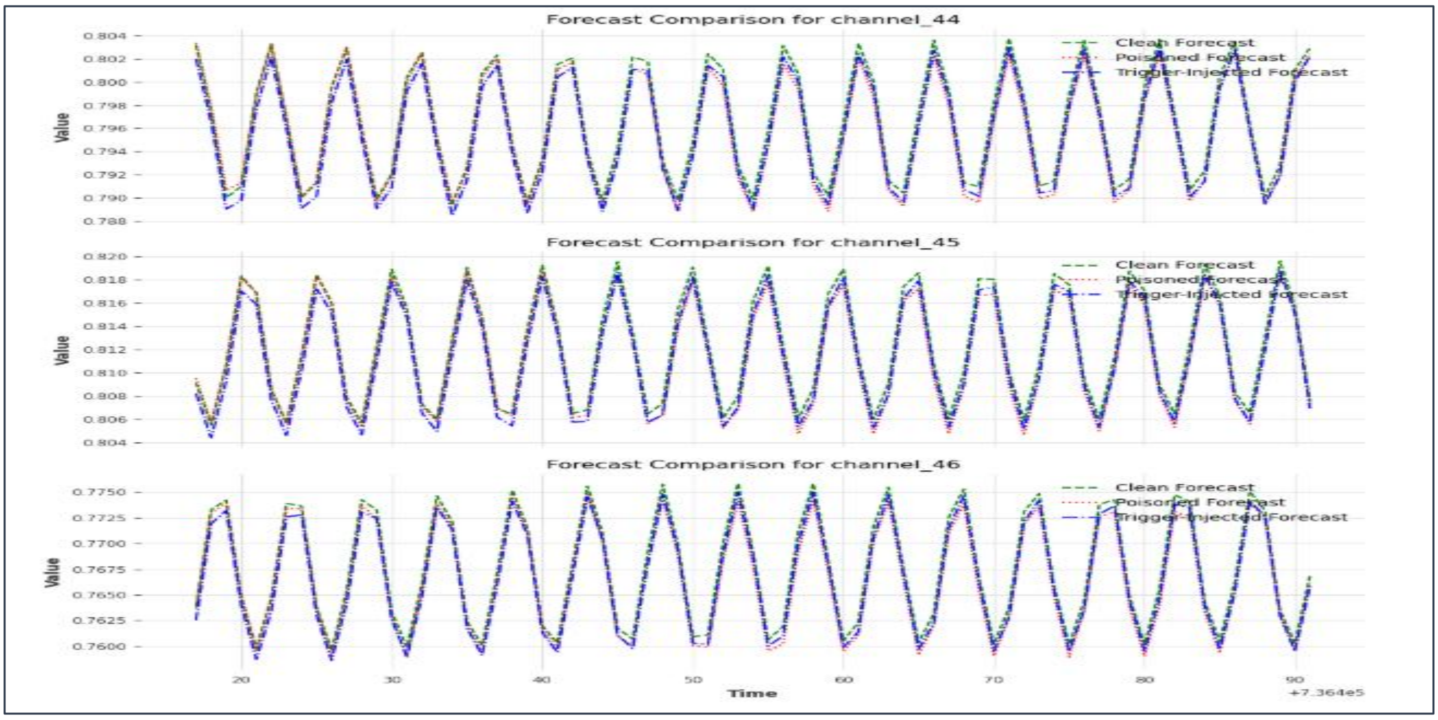


*Figure 2(a) Forecast of clean & poisoned model on clean input vs clean model on triggered input*

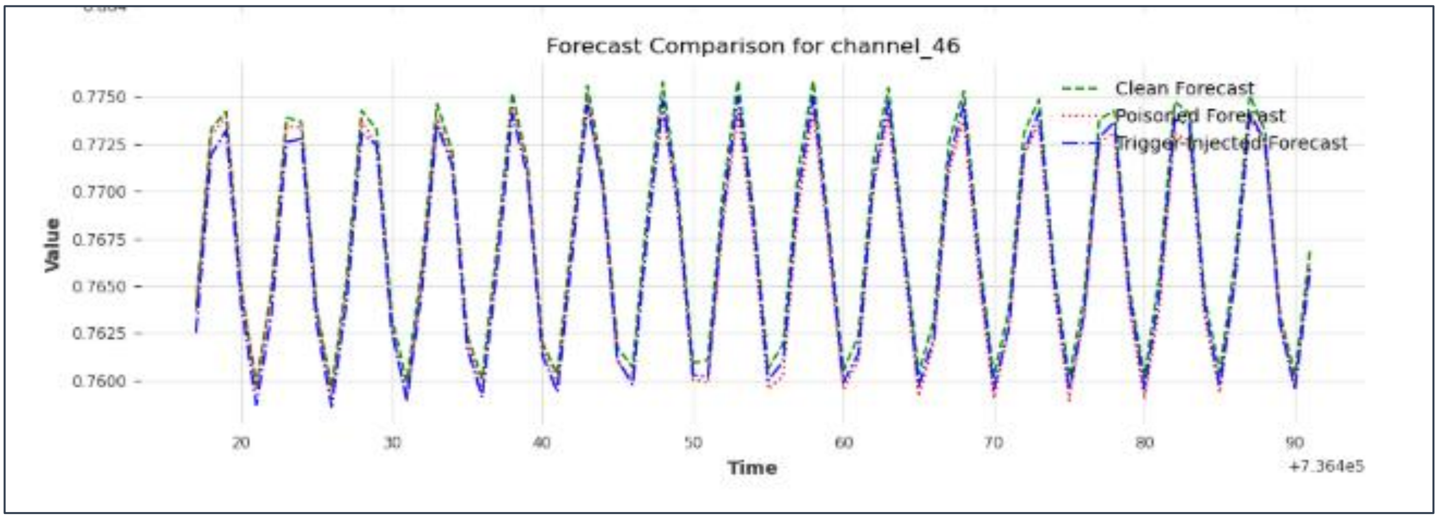


*Figure 2(b) Enlarged sample image*

Figure 2 shows that the forecast of the clean model on the triggered input diverges from its forecast on the clean input, while simultaneously moves closer toward the forecast of the poisoned model on clean input. This distortion pattern is further substantiated by the results presented in Figure 3.

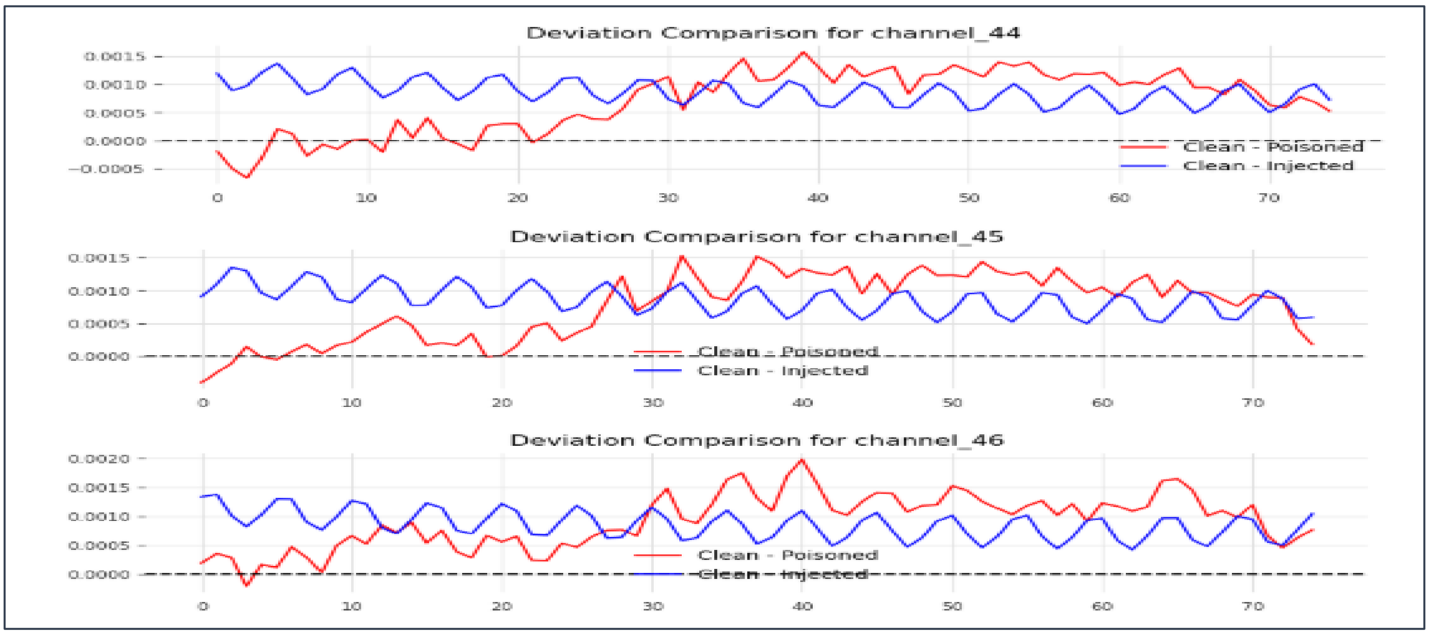


*Figure 3: Deviation comparison*

In the deviation comparison plot (Figure 3), two cases are evaluated: (i) the difference between the forecasts of the clean and poisoned models on clean input, and (ii) the difference between the clean model's forecasts on clean versus triggered input. It can be observed that the deviation induced in the clean model by the triggered input closely matches the deviation of the poisoned model from the clean model on clean input, both in magnitude and frequency.

Furthermore, Figure 4 compares the forecasts of the poisoned model on clean versus triggered input, showing that the forecast on the triggered input deviates from that on the clean input by disproportionately high margin. This observation supports the conjecture that, when presented with a close approximation of the trigger, the poisoned model is driven into its backdoor subspace, resulting in outputs with substantially higher deviations.

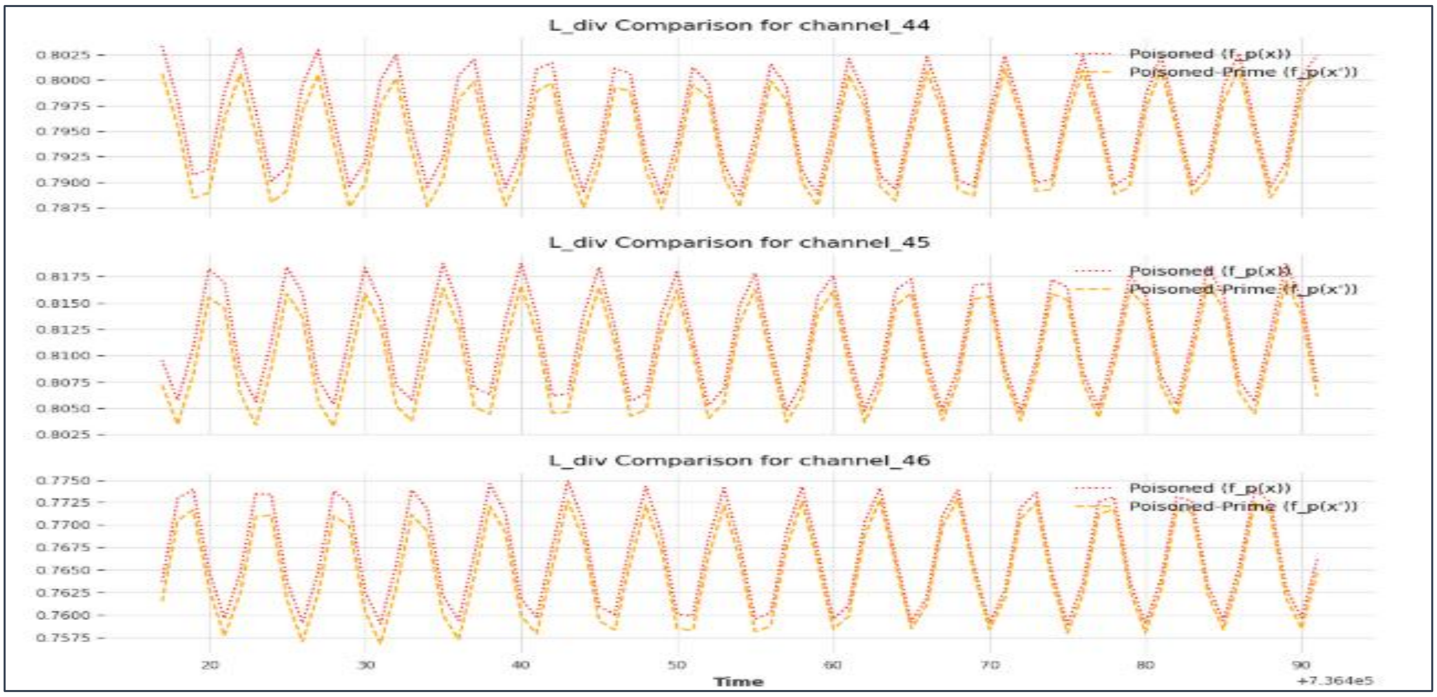


*Figure 4(a) Poisoned Model Forecast (clean vs triggered input)*

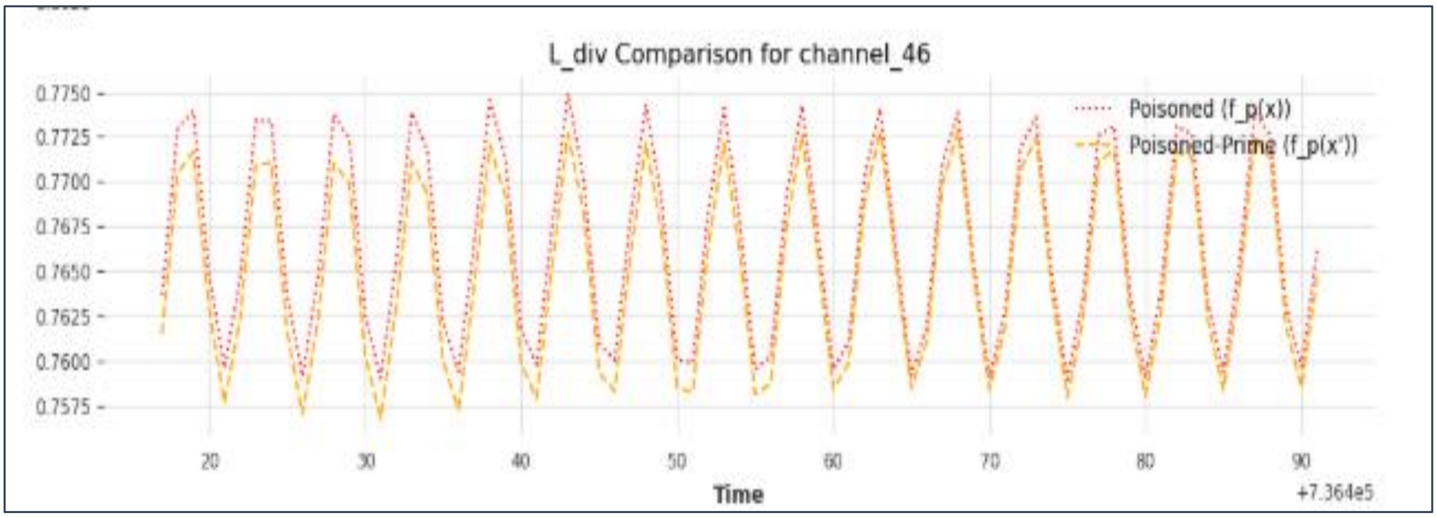


*Figure 4(b) Enlarged sample image*

### 5.2 Quantitative Analysis:

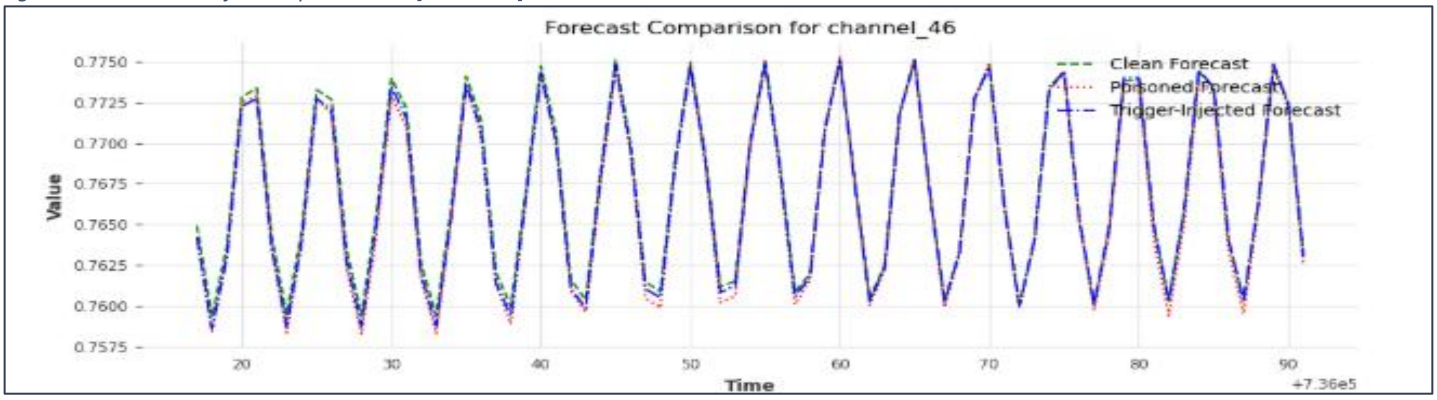
These findings are further validated through mean absolute error (MAE) analysis of the forecasts.

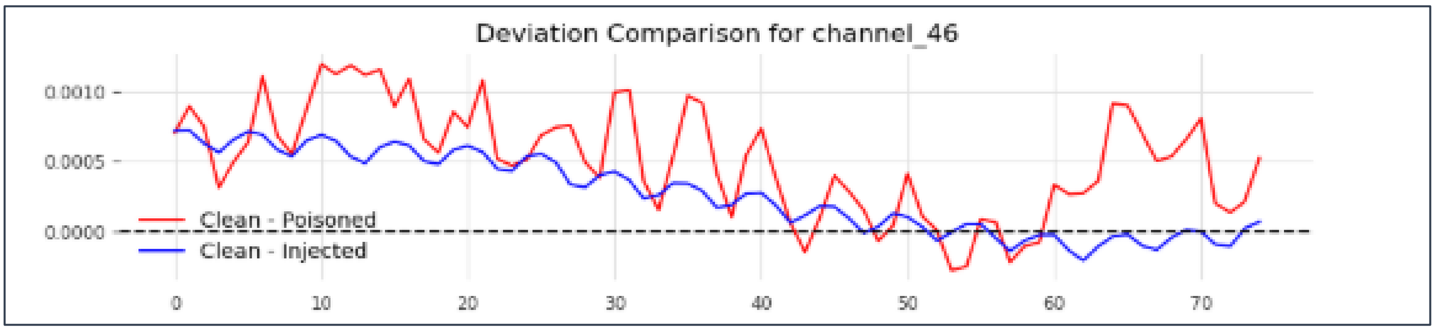
- The minimum loss value obtained through the Bayesian Optimization process was 0.0010.
- The MAE between the forecasts of the clean and poisoned models on clean input is 0.00082, whereas the MAE between the forecasts of the clean model on clean versus triggered input is 0.00085. This demonstrates that the triggered input induces a bias in the clean model that closely resembles the bias introduced by the poisoned model, thereby reinforcing the validity of the discovered trigger as a close approximation of the original.
- Similarly, the MAE between the poisoned model's forecasts on clean versus triggered input is 0.0016, which is approximately twice the distortion observed in the poisoned model's forecasts on clean input alone. This further suggests that the discovered trigger closely approximates the original trigger and possibly stimulated backdoor behaviour in poisoned model.

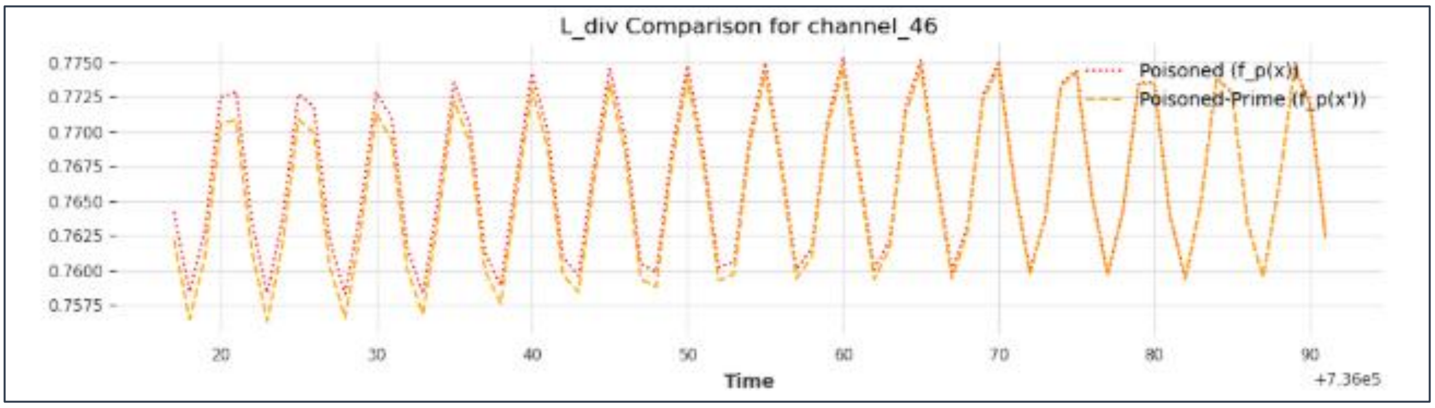
### 5.3 Secondary Analysis

The same procedure was repeated on a different 400-sample window, specifically [−800:−400], the resulting forecast plots and corresponding MAE values given below:

*Figure 5 Forecast Plots for sample window [-800:-400]*







- The forecast of the clean model on triggered input diverges from its forecast on clean input, while moving closer to the distorted forecast of the poisoned model on clean input.
- The deviation in the clean model's forecast on triggered input is comparable in both magnitude and direction to the deviation of the poisoned model's forecast on clean input.
- The poisoned model's forecast on triggered input appears substantially distorted relative to its forecast on clean input.
- The minimum loss value was 0.0005.
- **Mean Absolute Error (MAE) results:**
  - Clean vs. poisoned model on clean input: **0.00042**
  - Clean model on clean vs. triggered input: **0.00030**
  - Poisoned model on clean vs. triggered input: **0.00087** (≈2× the distortion observed in the poisoned model's forecast on clean input).

| **S.No** | **Models/Input** | **MAE Value on 1st run** | **MAE Value on 2nd run** |
|---|---|---|---|
| 1 | Clean vs. Poisoned model on clean input | $0.82 \times 10^{-3}$ | $0.42 \times 10^{-3}$ |
| 2 | Clean model on clean vs. triggered input | $0.85 \times 10^{-3}$ | $0.30 \times 10^{-3}$ |
| 3 | Poisoned model on clean vs. triggered input | $1.6 \times 10^{-3}$ | $0.87 \times 10^{-3}$ |

## 6. Conclusion:

The results of the secondary run reinforce the observations from the initial experiment. Collectively, these findings provide strong evidence that:

- The consistent deviation of the poisoned model's output across multiple sample windows, along with the clean model on triggered input exhibiting similar distortions, indicates that the observed behavior is not attributable to stochastic variance but rather to the presence of a Trojan-induced poisoning.
- The resemblance between distortions in the clean model's output on triggered input and the poisoned model's output on clean input, combined with the pronounced irregular distortions produced by the poisoned model on triggered input, strongly suggest that the trigger discovered through the optimization procedure represents a close reconstruction of the original trigger responsible for the model poisoning.
- The same procedure was repeated to reconstruct triggers for the remaining 44 poisoned models.