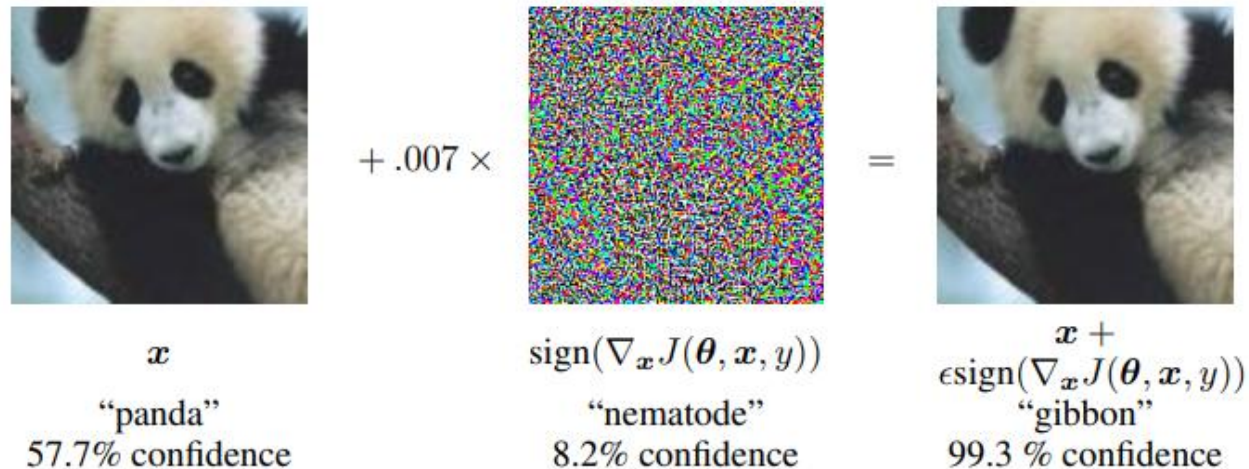


CSCD94 Final Report: Targeted Manipulation

DOMINIK LUSZCZYNSKI

Adversarial Attacks

- Adversarial attacks occur when an attack slightly modifies the input to a model, typically by adding noise, which causes a model to produce an incorrect result [2].



White Box vs Black Box

- White box attack occurs when the attacker has full access to all information about the model, including parameters, which enables the attacker to exploit gradient information [3].
- Black-box attack occurs when the model is hidden from the attacker, and they do not have any knowledge about the structure or parameters [3], [4].

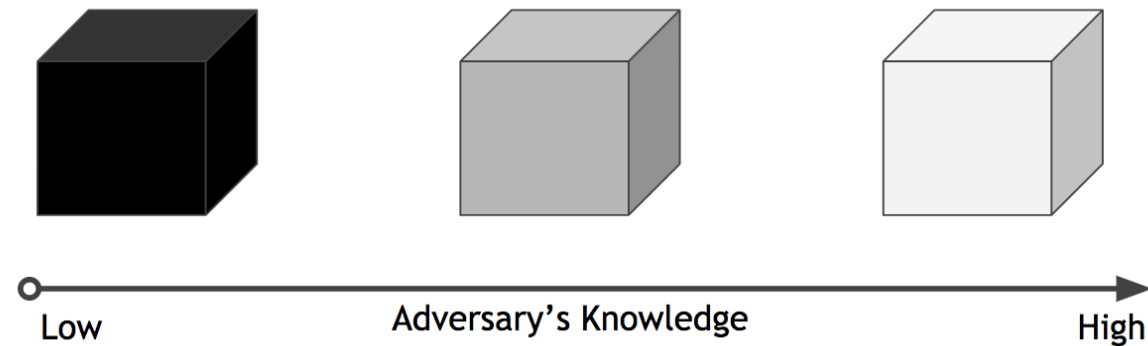


Image Source: <https://secml.github.io/class1/>

Threat Model



-
- Assume that ML/AI forecasting models for sensitive data, such as stock data, are regularly used by consumers and in production.
 - The attacker aims to alter existing stock data used as input for the forecasting model in hopes to influence the market.
 - All attacks are performed in a white-box setting, meaning the attacker has access to the forecasting model's gradients so they can generate stronger attacks.
 - Furthermore, this research argues that the discussion between white and black box adversarial attack methods (gradients available/unavailable) can be avoided with other cyber-attacking tools such as malware.

The Basis of an Attack

- When gradient descent is performed, the weights are moved in the opposite direction of the gradient.

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \lambda \nabla E(\mathbf{w}_t)$$



Change the - into a +

Imaged-Based Adversarial Attacks

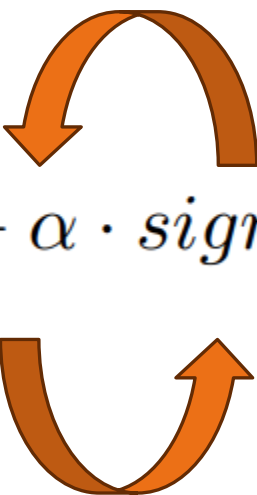
Fast Gradient Sign Method (FGSM)

- The FGSM involves computing the loss with respect to the predictions, then adjusting the adversarial attack x_{adv} in the direction that maximizes the loss [17].
- Here x_{adv} is our adversarial input, x is the original input, and ϵ is our maximum perturbation size.

$$x_{adv} = x + \epsilon \cdot \text{sign}(\nabla_{x_{adv}} \text{loss}) \quad [17]$$

Basic Iterative Method (BIM)

- Rather than performing the attack in a single step, update x_{adv} iteratively, for some number of desired iterations, with some step size α .
- However, to ensure that we stay within the ε -neighbourhood we need to clip x_{adv} to $x + \varepsilon$ and $x - \varepsilon$.


$$x_{adv} = x + \alpha \cdot \text{sign}(\nabla_{x_{adv}} \text{loss})$$

The Dreaded Local Minima



Momentum Iterative FGSM (MI-FGSM)

- Many optimization algorithms may struggle with getting stuck in a local minima.
- Similar to the momentum technique used to accelerate gradient descent algorithms, the MI-FGSM uses the same technique to help avoid getting stuck [1].
- Initialize $g_0 = 0$, then

$$g_{t+1} = \mu \cdot g_t \cdot \frac{\nabla_{x_{adv}} loss}{\|\nabla_{x_{adv}} loss\|_1}$$

and we the update x_{adv} in the direction of g_{t+1}

$$x_{adv} = x_{adv} + \alpha \cdot sign(g_{t+1}) \quad [1]$$

C&W Attack

- The Carlini and Wagner attack (C&W) deviates from the general approach to adversarial attacks. Rather than iteratively modifying x_{adv} , a C&W attack aims to find the optimum noise vector η that is added to the adjprc [8].
- The goal is to solve the following optimization problem:

$$\min ||\eta||_2 + f(x_{adv} + \eta) \quad [8]$$

PROBLEM

These attacks aim to maximize the error produced by the model; however, this is insufficient for an attacker since it does not lead to any clear advantage.



The Basis of an Attack

- When gradient descent is performed, the weights are moved in the opposite direction of the gradient.

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \lambda \nabla E(\mathbf{w}_t)$$



Targeted Iterative Method (TIM)

- Although the goal for many adversarial attacks is to maximize the error of the model, in realistic scenarios the attacker would want to choose the direction in which the model fails.
- There are 2 avenues for the targeted attack:
 - 1) Choose some new sequence Y^* as our target
 - 2) Choose a static margin γ
- The attack requires a parameter d and γ which represent the direction and margin, and calculates the new target as:

$$tar = adjprc + d \cdot \gamma \quad [3]$$

then we the update x_{adv} as:

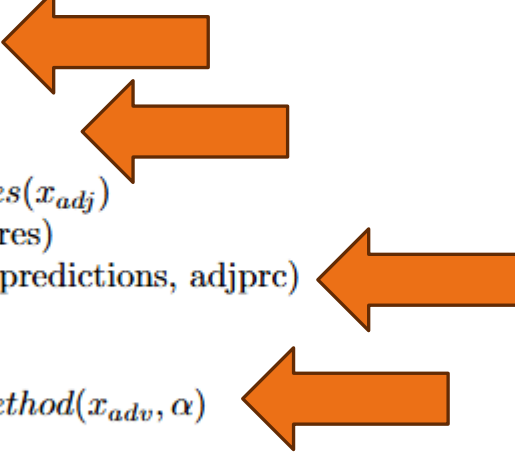
$$x_{adv} = x_{adv} - \alpha \cdot sign(\nabla_{x_{adv}} loss) \quad [3]$$

How Do We Make an Adversarial Attack?

Algorithm 3 Generic Iterative Adversarial Attack Algorithm on the N-HiTS model

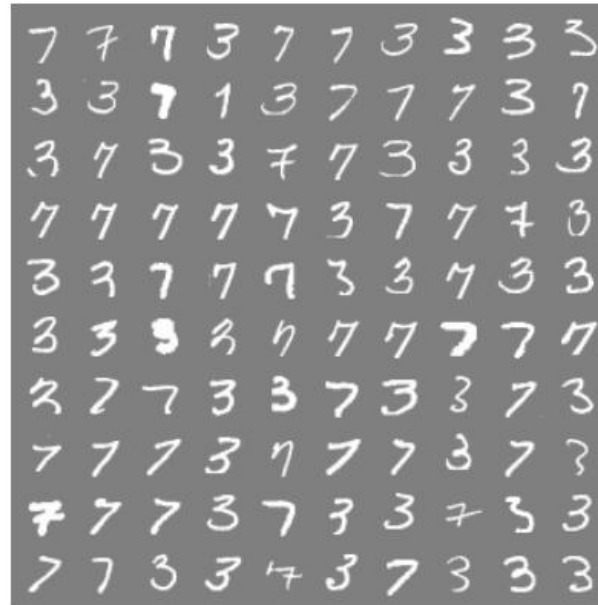
Input: Stock Forecasting model **model**, adjusted price **adjprc**, number of iterations **iter** $\in \mathbb{R}$, maximum perturbation $\epsilon \in \mathbb{R}_{\geq 0}$, and the step size $\alpha = 1.5 \cdot \epsilon / \text{iter} \in \mathbb{R}_{\geq 0}$

```
1:  $x_{adv} \leftarrow \text{adjprc}$ 
2: for  $i$  in range(iter) do
3:    $x_{adv} \leftarrow \text{requires grad}$ 
4:   model  $\leftarrow$  zero grad
5:   features  $\leftarrow \text{getFeatures}(x_{adj})$ 
6:   output  $\leftarrow \text{model}(\text{features})$ 
7:   loss  $\leftarrow \text{LossFunction}(\text{predictions}, \text{adjprc})$ 
8:   loss.backward()
9:   with no grad:
10:     $x_{adv} \leftarrow \text{AttackMethod}(x_{adv}, \alpha)$ 
11:   detach  $x_{adv}$ 
12: end for
13: return  $x_{adv}$ 
```

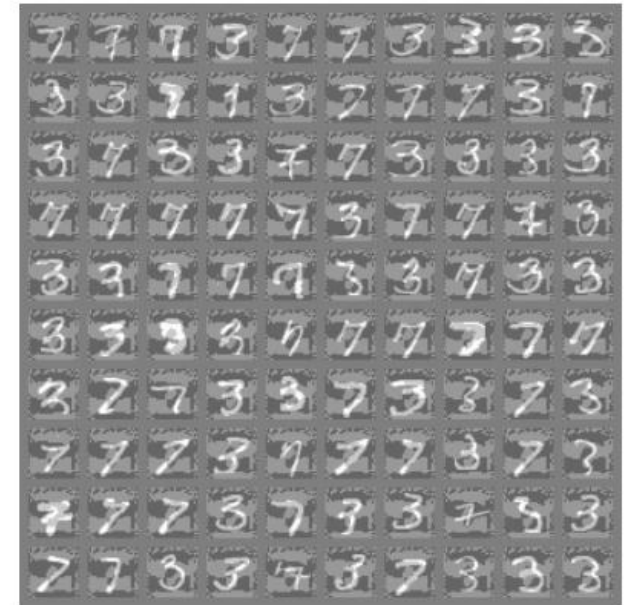


PROBLEM

The majority of existing adversarial research has been done on image and text classification, while attacks on time-series data is still in its infancy [3].



(c)



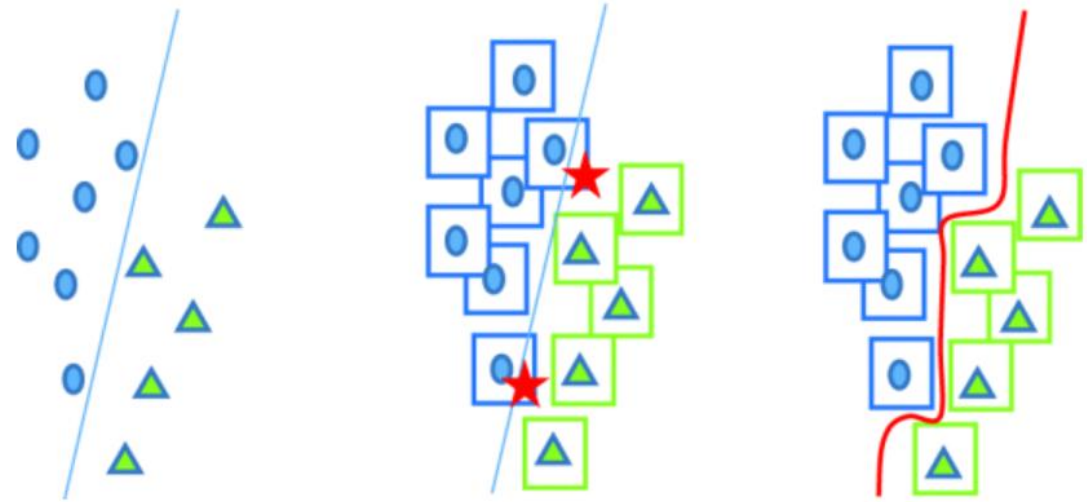
(d)

Current Research

- There have been recent advancements in the time series domain, with Gallager et al., attacking a simple three layered Convolutional Neural Network (CNN) forecasting the Google Stock from 2006 to 2018 with Fast Gradient Sign Method [5].
- Rathore et al., applied the Fast Gradient Sign Method and the Basic Iterative Method on a classification model trained on 54 different time series datasets related to healthcare, vehicle sensors and electrical equipment [6].
 - Rathore et al. also showed that defence techniques such as adversarial training are effective [6].

PROBLEM

The introductory research in the time series domain typically focus on classification problems, reapplying image/text based adversarial attacks on time series data, and do not consider the believability of an adversarial example [3], [7].



Stealthy Iterative Method (SIM)

- Shen and Li introduced Stealthy attacks which rely on using cosine similarities to ensure temporal characteristics of time series remain intact [3].
- After performing an iteration of the standard BIM, compare the cosine similarity between x and the new x_{adv} vs x with $x + \epsilon$ and finally with x and $x - \epsilon$:
 - If the $\text{cosine_similarity}(x, x_{adv}) < \text{cosine_similarity}(x, x + \epsilon)$ then $x_{adv} = x + \epsilon$
 - If the $\text{cosine_similarity}(x, x_{adv}) < \text{cosine_similarity}(x, x - \epsilon)$ then $x_{adv} = x - \epsilon$
 - Otherwise, x_{adv} is kept the same [3].

Targeted Iterative Method (TIM)

- Although the goal for many adversarial attacks is to maximize the error of the model, in realistic scenarios the attacker would want to choose the direction in which the model fails.
- There are 2 avenues for the targeted attack:
 - 1) Choose some new sequence Y^* as our target
 - 2) Choose a static margin γ
- The attack requires a parameter d and γ which represent the direction and margin, and calculates the new target as:

$$tar = adjprc + d \cdot \gamma \quad [3]$$

then we the update x_{adv} as:

$$x_{adv} = x_{adv} - \alpha \cdot sign(\nabla_{x_{adv}} loss) \quad [3]$$

PROBLEM

Previous targeted adversarial attacks, such as the attacks developed by Shen and Li, aim to minimize the loss between the target and the prediction, where the target is either the real data modified by some margin or a new sequence Y^* [3].

- While this is an effective strategy, it is unrealistic for an attacker to generate a new sequence Y^* for different inputs and, modifying with a scalar margin would only shift the predictions, leading to poor applicability for an attacker as the temporal characteristics are left unchanged.
- The limitations of current targeted methods is further exacerbated in the financial time series domain, and given the stock data is extremely volatile and noisy, it is necessary for the target to be embedded inside the objective function.

Slope-Based Attacks

General Slope Attack (GSA)

- In the simplest case, the attacker would want to only affect the endpoints of a prediction, as this might show a late positive surge or decline affecting the victim's desire to purchase a stock.
- The General Slope Attack (GSA) uses the discrete formula to find the slope between two points. That is,

$$m = \frac{y_2 - y_1}{x_2 - x_1}$$

- Then, the objective function requires the slope m target direction $t \in \{-1, 0, 1\}$, and hyperparameters c and d and calculates the following loss:

$$loss = \begin{cases} ce^{-tdm} & \text{if } t \in \{-1, 1\} \\ cm^2 & \text{if } t = 0 \end{cases}$$

Least Squares Slope Attack

- While the GSA attack aims to change the slope of the time series forecast near the endpoints of the prediction, the nature of the slope calculation prevents the overall trend from changing.
- Inspired by Least Squares Linear Regression, the new calculation for the slope would be the weight term in the closed form solution for Least-Squares Linear Regression. Specifically,

$$w^* = \frac{\sum_i (y_i - \bar{y})(x_i - \bar{x})}{\sum_i (x_i - \bar{x})^2}$$

Slope Attack Algorithm

Algorithm 1 Generic slope attack for financial time series data

Input: Stock Forecasting model **model**, adjusted price **adjprc**, number of iterations **iter** $\in \mathbb{R}$, maximum perturbation $\epsilon \in \mathbb{R}_{\geq 0}$, the step size $\alpha = 1.5 \cdot \epsilon / \text{iter} \in \mathbb{R}_{\geq 0}$, target direction **t** $\in \{-1, 0, 1\}$, and scalars **c, d** $\in \mathbb{R}$

```
1:  $x_{adv} \leftarrow \text{adjprc}$ 
2: for  $i$  in  $\text{range}(\text{iter})$  do
3:    $x_{adv} \leftarrow \text{requires grad}$ 
4:    $\text{model} \leftarrow \text{zero grad}$ 
5:    $\text{features} \leftarrow \text{getFeatures}(x_{adj})$ 
6:    $\text{output} \leftarrow \text{model}(\text{features})$ 
7:    $m \leftarrow \text{SlopeMethod}(\text{output})$ 
8:   if  $t \in \{-1, 1\}$  then
9:      $\text{loss} \leftarrow ce^{-tdm}$ 
10:  else
11:     $\text{loss} \leftarrow cm^2$ 
12:  end if
13:   $\text{loss.backward}()$ 
14:  with no grad:
15:     $\eta \leftarrow \alpha \cdot \text{sign}(\nabla x_{adv})$ 
16:     $x_{adv} \leftarrow x_{adv} - \eta$ 
17:     $x_{adv} \leftarrow \text{clamp}(x_{adv}, \text{adjprc} - \epsilon, \text{adjprc} + \epsilon)$ 
18:   $\text{detach } x_{adv}$ 
19: end for
20: return  $x_{adv}$ 
```

PROBLEM

A lot of the research described in the previous slides were performed on shallow neural networks.

- SIM (Shen and Li):
 - 3-layered CNN with hidden dimension of 60 and a fully connected output layer [3].
 - 3-layered LSTM with hidden dimension of 100 and a full connected output layer [3].
 - 3-layered GRU with hidden dimension of 100 and a full connected output layer [3].
- Gallager et al.:
 - 3-layered CNN [5].

Let's Fool a More Complex Model

N-HiTS

- N-HiTS is a novel projection model which builds upon the N-BEATS architecture, simultaneously improving computational performance and accuracy by sampling the time series at different rates [14].
- The N-HiTS architecture uses Multi-layer Perceptrons (MLPs) for each block of the time series to estimate coefficients for the backcasts and forecasts [14].
- However, the N-HiTS model introduces novel components to each block, like using MaxPool with some kernel size k_b for each block b , claiming that it helps the MLP focus on low frequency and large-scale contents of the time series, which allows it to sample different rates [14].
- After the pooling layer, a non-linear regression is applied to estimate the backcasts and forecasts, and to eventually generate the final predictions, hierarchal interpolation is performed [14].

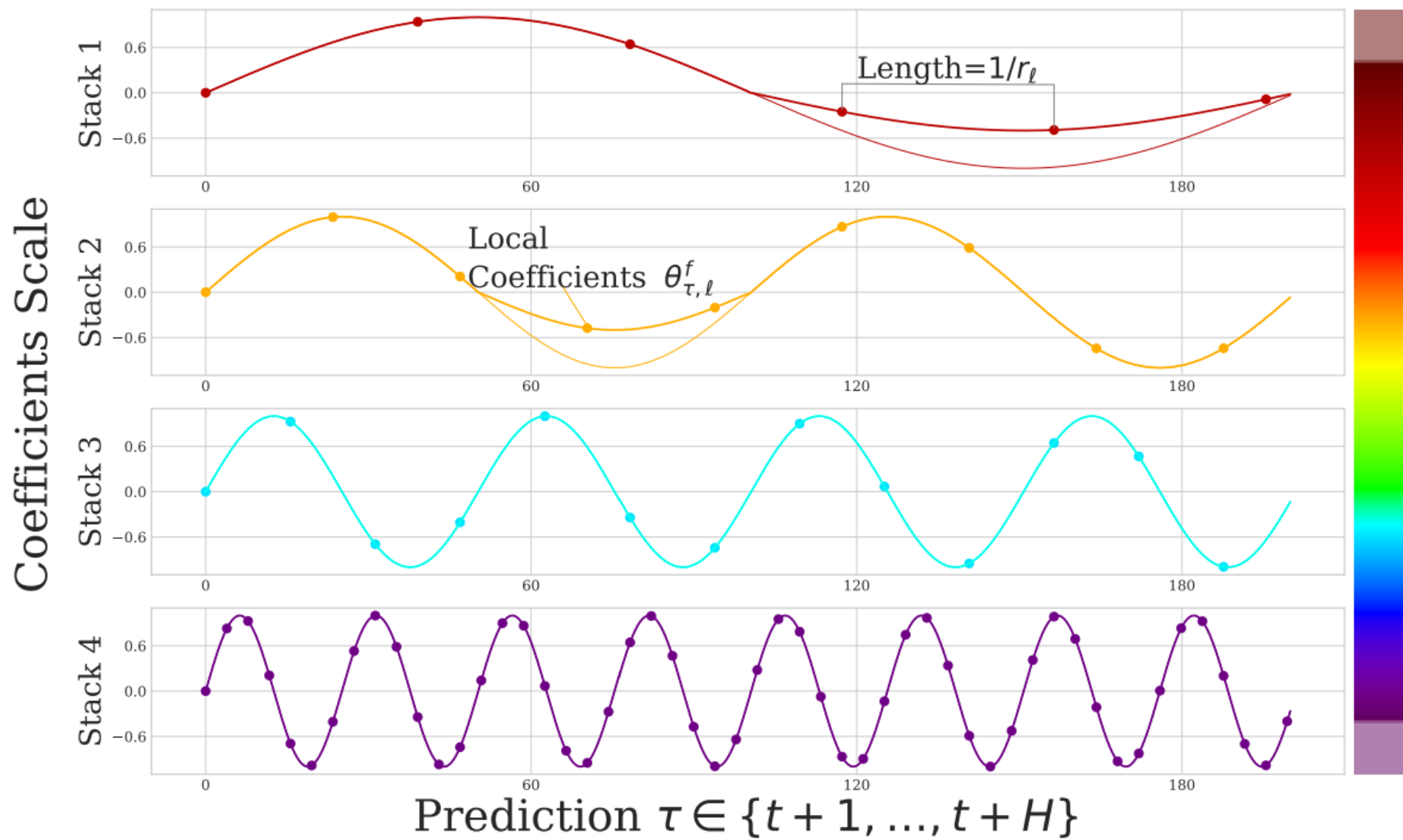


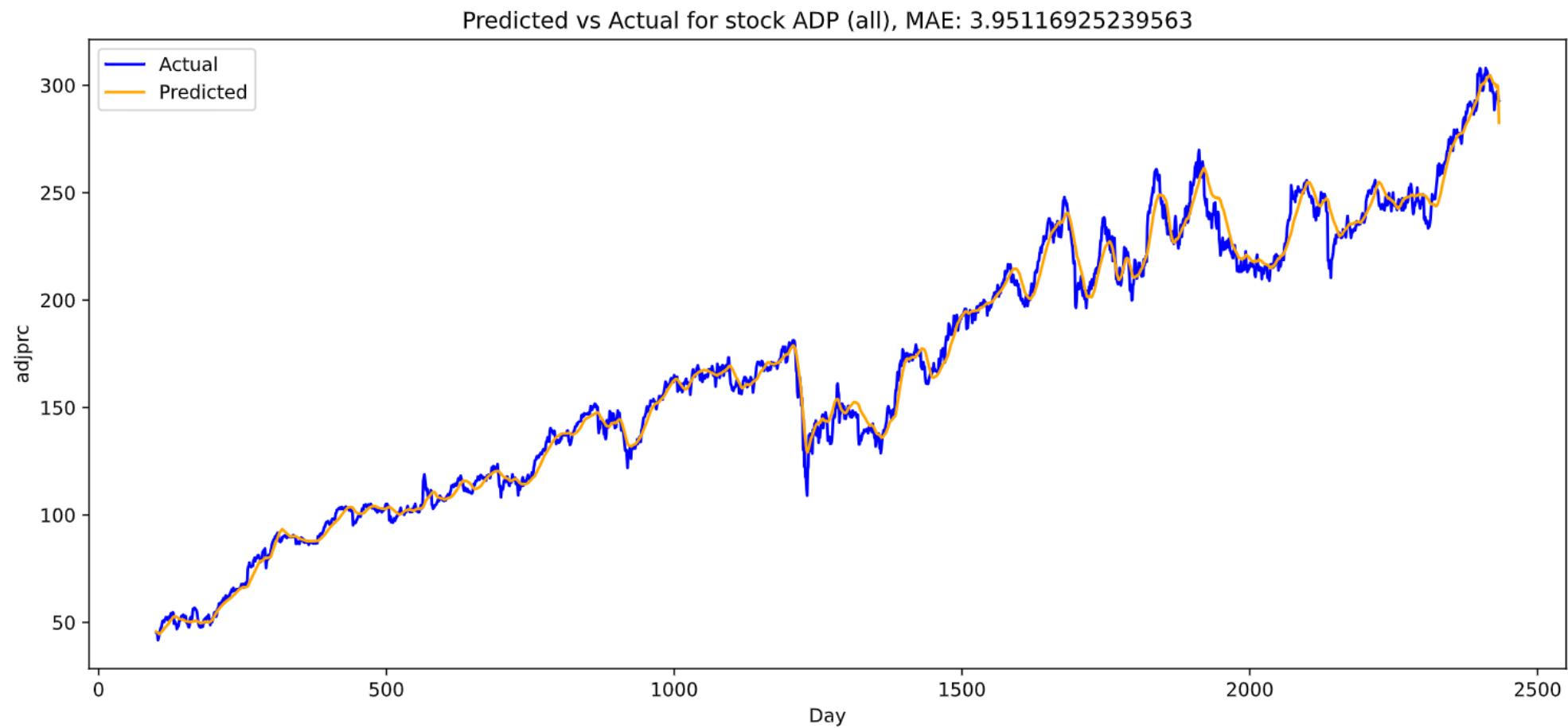
Image Source: C. Challu et al. [14]

Dataset

- The daily adjusted price (adjprc) of all S&P 500 stocks was collected from Center for Research Security Prices (CRSP) for the 2015:05:01- 2025:05:01 period.
- Rather than training on a single stock, to improve generalizability the N-HiTS model was trained on 360 different stocks from the S&P 500. The validation set included 48 stocks while the test set contained 72 stocks.
- The split was determined by a stratified split on the stock price collected from [15].

Dataset

- Several Features were derived from the adjprc such as:
 - Rolling mean with window sizes of 5, 10, 20
 - Rolling standard deviation with window sizes of 5, 10, 20
 - Log Returns
 - Rate of Change with a delta of 5
 - Exponential moving averages with window sizes 5, 10 and 20
- In addition, the day of the week was extracted from the date, as it has been shown that there are differences in volatility between the beginning and end of the trading week [16].



N-HiTS Performance

Average MAE: 4.11

Average RMSE: 6.04

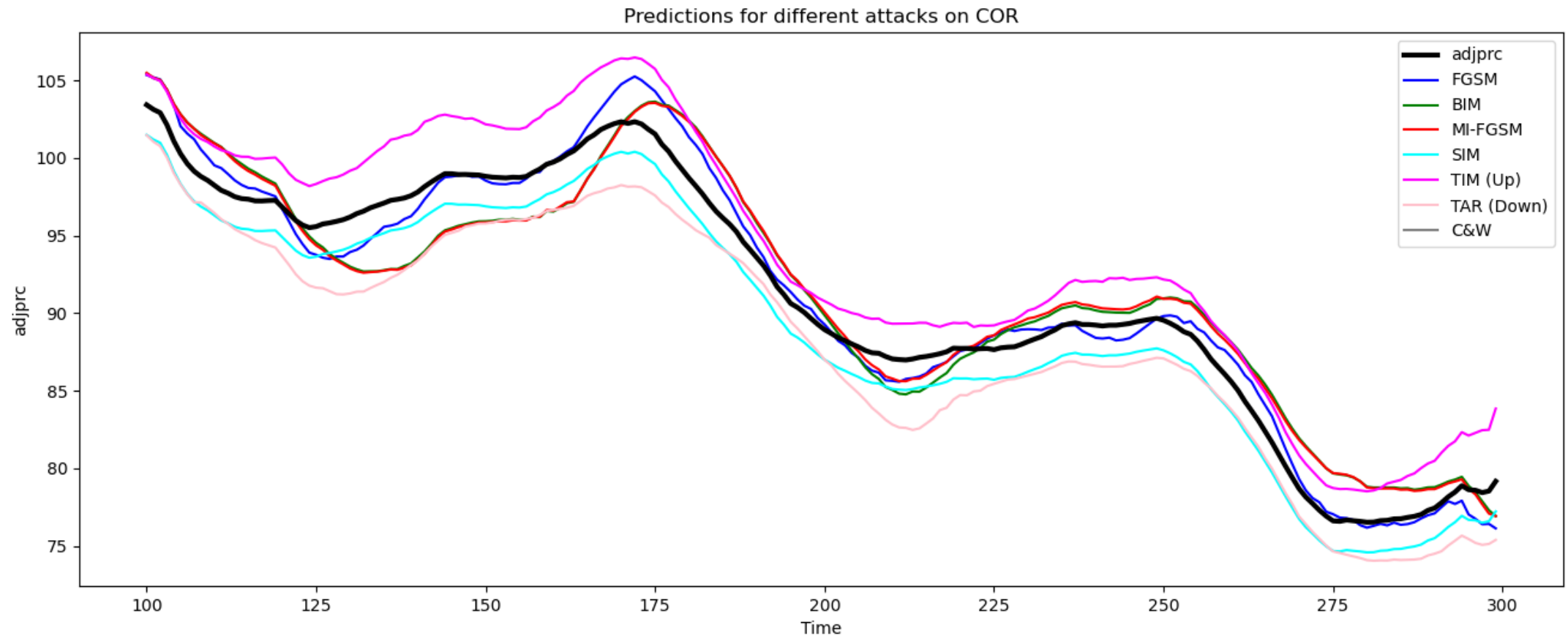
MAPE: 3.48%

N-HiTS Hyperparameters

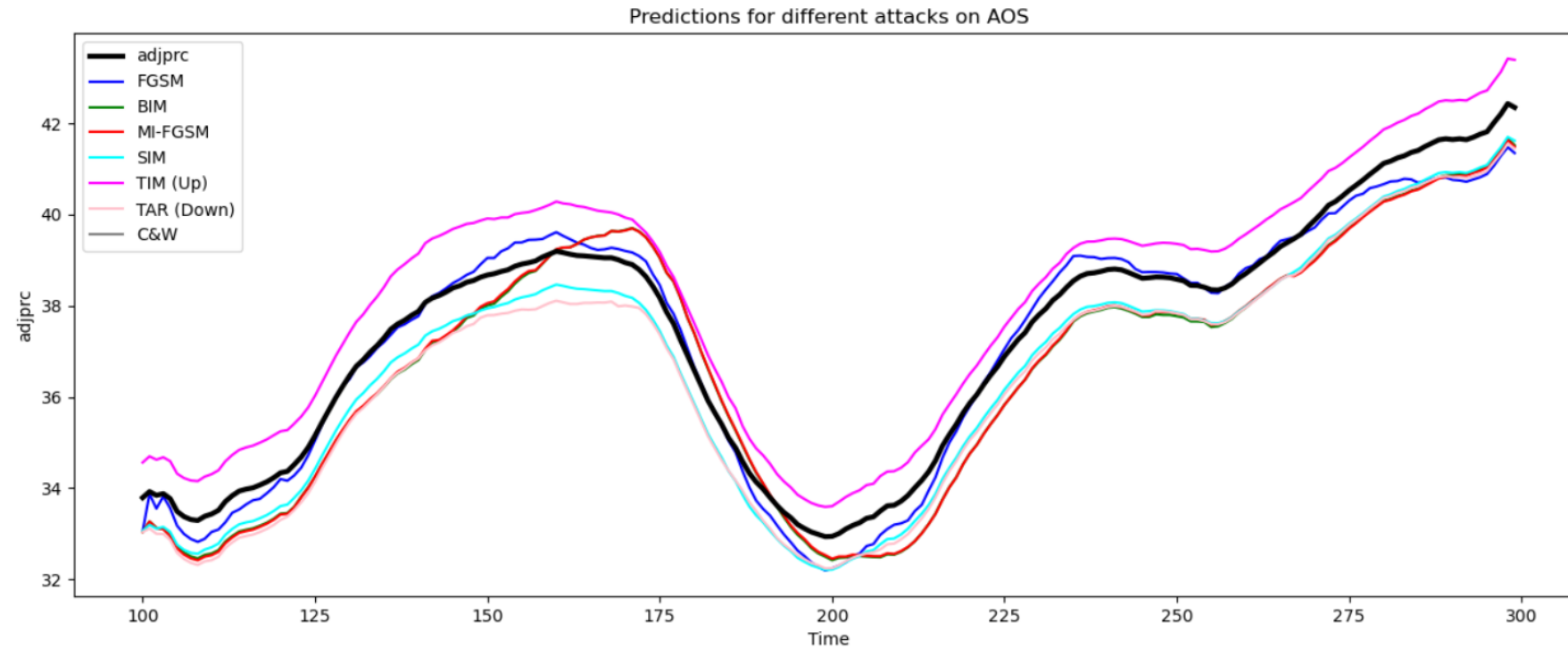
Table 5: Hyperparameters used for the N-HiTS model

<i>Hyperparameter</i>	<i>Value</i>
Learning Rate	10^{-3}
Weight Decay	10^{-4}
Hidden Size	64
Batch Normalization	True
Early Stopping	True
Max/Min Encoder Length	100
Max/Min Prediction Length	20
Batch Size	500
Epochs	100

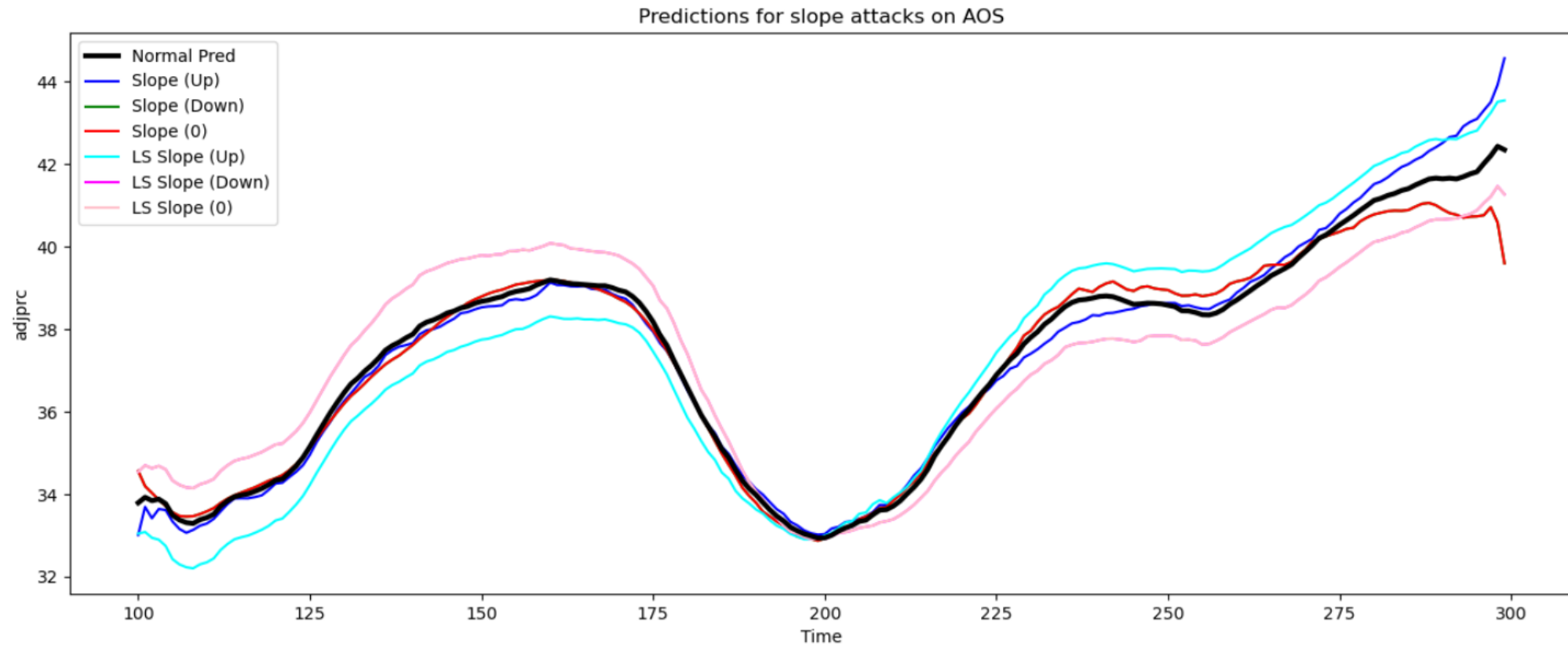
Baseline Adversarial Attacks



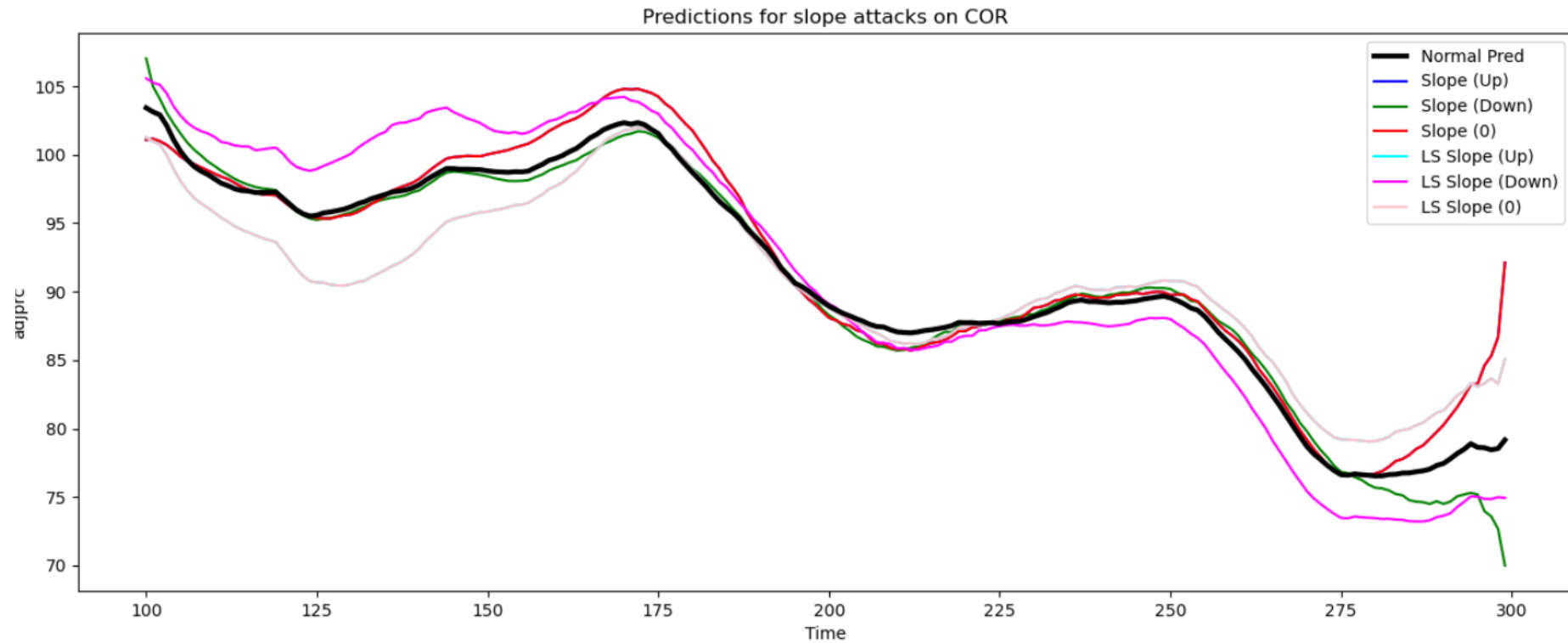
Baseline Adversarial Attacks



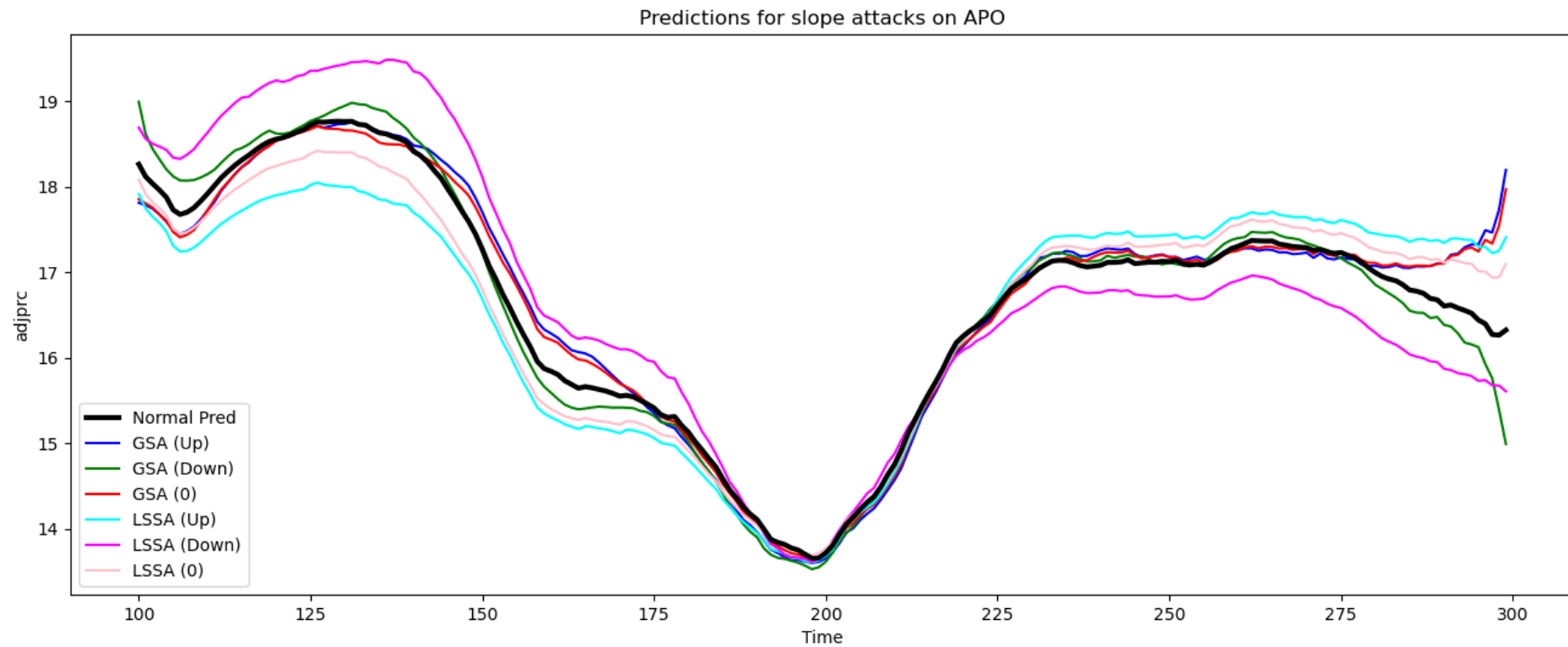
Slope Based Attacks



Slope Based Attacks



Slope Based Attacks

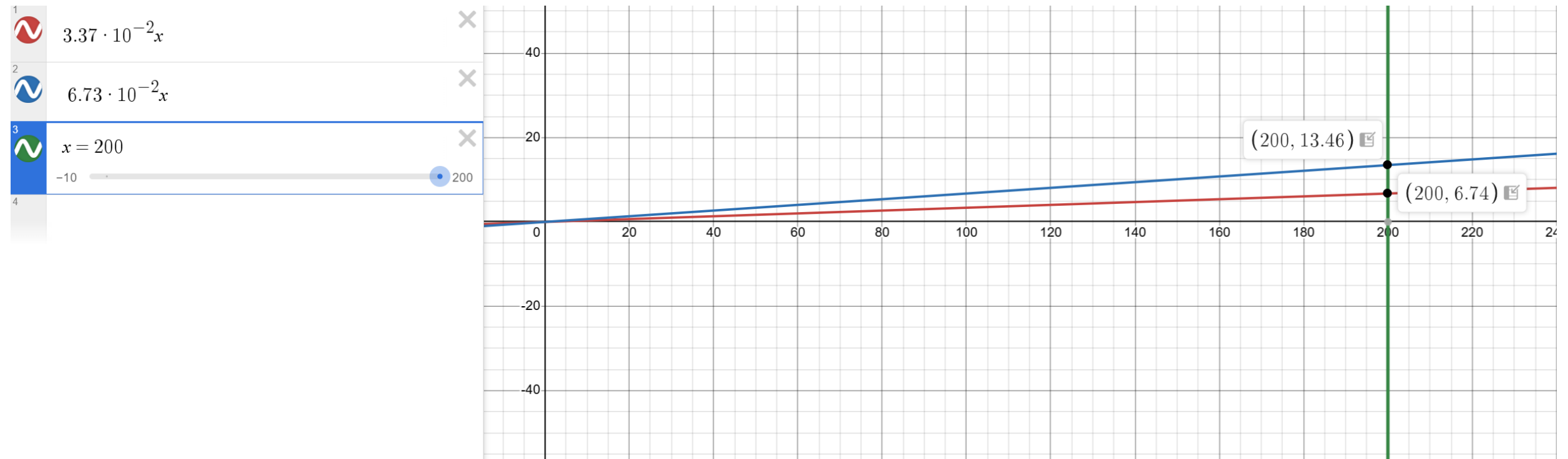


Attack Comparison

Table 1: Average metrics for different attack methods performed on the first 300 days of each recording, with $\epsilon = 2\% \cdot \text{median}(\text{adjprc})$. The best metrics are bolded.

<i>Attack</i>	<i>MAE</i>	<i>RMSE</i>	<i>MAPE</i>	<i>Gen. Slope</i>	<i>LS Slope</i>
Normal	2.15	2.72	3.82×10^{-2}	3.37×10^{-2}	2.22×10^{-2}
FGSM	2.57	3.21	4.51×10^{-2}	3.22×10^{-2}	2.34×10^{-2}
BIM	3.38	3.99	5.68×10^{-2}	3.48×10^{-2}	2.39×10^{-2}
MI-FGSM	3.37	3.99	5.67×10^{-2}	3.44×10^{-2}	2.39×10^{-2}
SIM	2.57	3.08	4.29×10^{-2}	3.37×10^{-2}	2.23×10^{-2}
TIM (Up)	2.49	3.21	4.52×10^{-2}	3.72×10^{-2}	2.00×10^{-2}
TIM (Down)	2.74	3.26	4.44×10^{-2}	3.32×10^{-2}	2.51×10^{-2}
<i>GSA (Up)</i>	2.26	2.88	4.03×10^{-2}	6.76×10^{-2}	2.77×10^{-2}
<i>GSA (Down)</i>	2.23	2.83	3.89×10^{-2}	-1.68×10^{-4}	1.75×10^{-2}
<i>GSA (0)</i>	2.30	2.93	4.01×10^{-2}	1.80×10^{-2}	2.00×10^{-2}
<i>LSSA (Up)</i>	2.49	3.10	4.26×10^{-2}	5.38×10^{-2}	4.96×10^{-2}
<i>LSSA (Down)</i>	2.71	3.33	4.63×10^{-2}	1.56×10^{-2}	-5.04×10^{-3}
<i>LSSA (0)</i>	2.68	3.31	4.55×10^{-2}	2.82×10^{-2}	1.29×10^{-2}

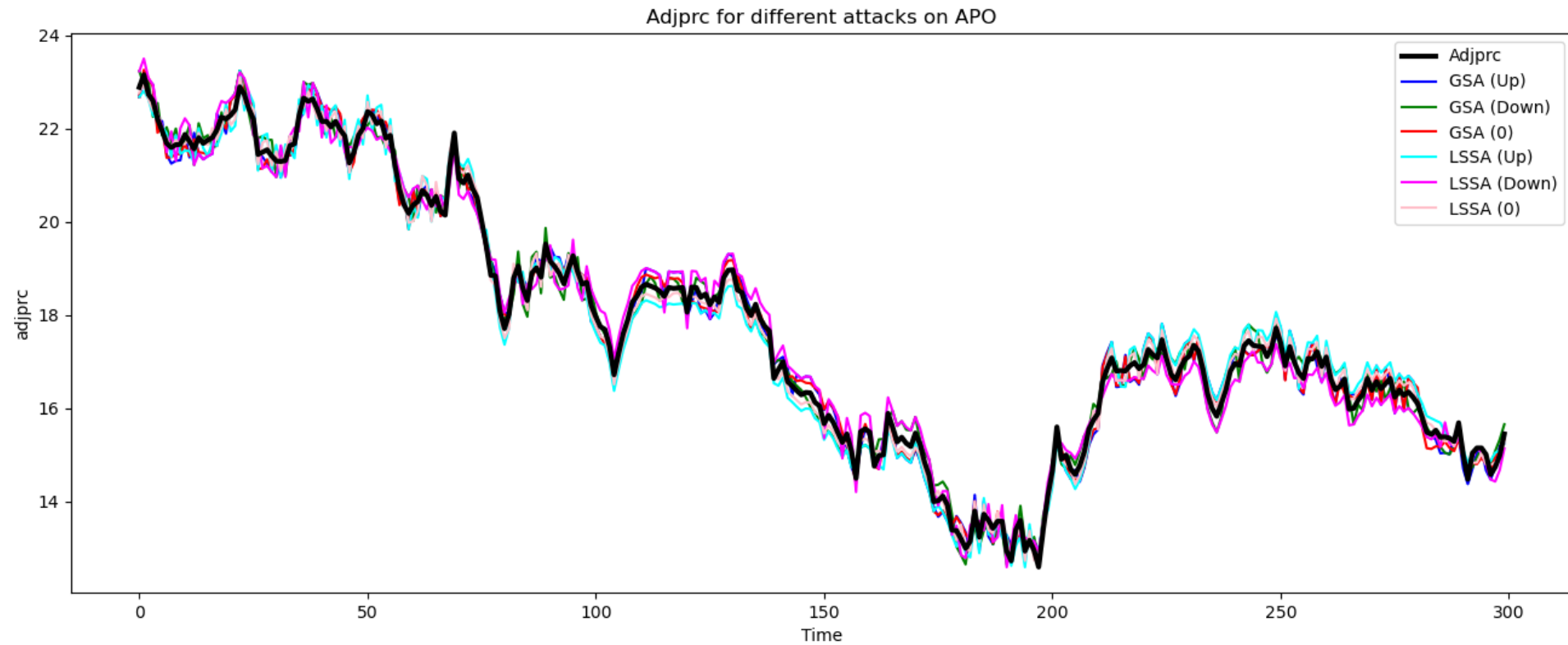
Attack Comparison



Slope Attacks With Various Epsilon Sizes

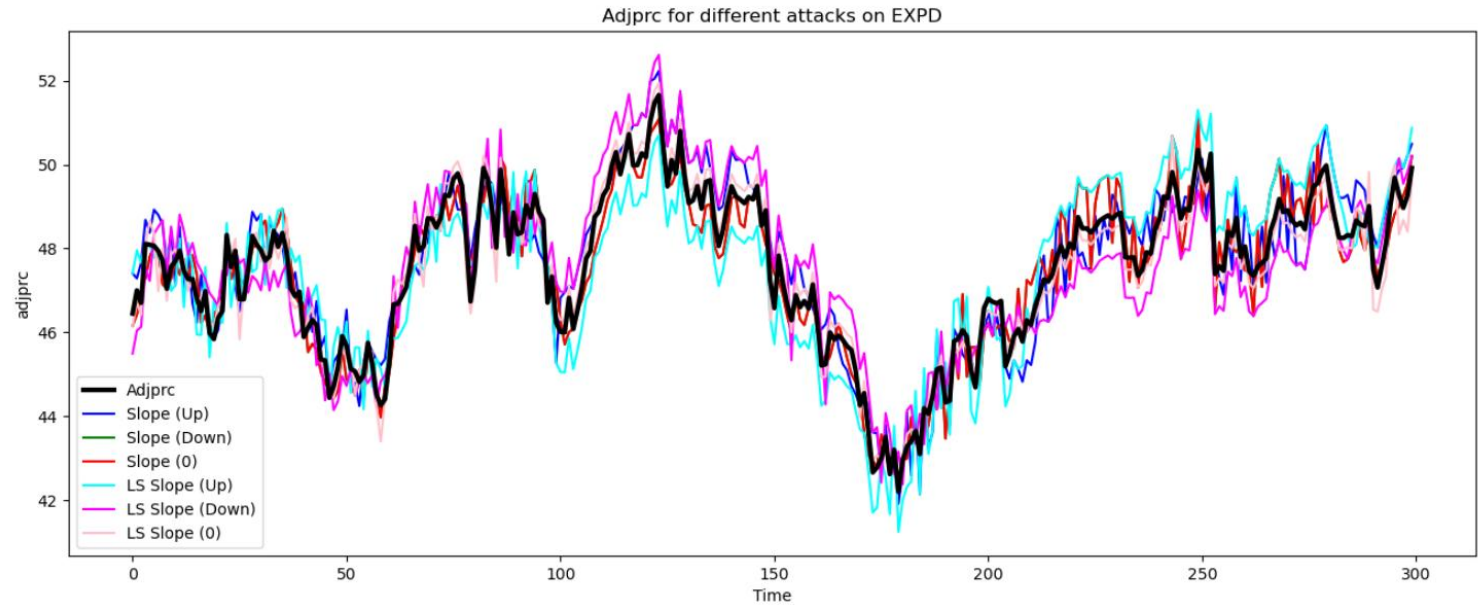
Table 2: Average slope values for varying relative percentages. Only the upward version of the attacks are considered since all other attacks follow the same trend.

<i>Attack</i>	<i>ϵ %</i>	<i>Gen. Slope</i>	<i>LS Slope</i>
Normal	0.0	3.37×10^{-2}	2.22×10^{-2}
GSA (Up)	0.5	4.37×10^{-2}	2.41×10^{-2}
	1.0	5.27×10^{-2}	2.57×10^{-2}
	1.5	6.06×10^{-2}	2.70×10^{-2}
	2.0	6.76×10^{-2}	2.77×10^{-2}
	2.5	7.39×10^{-2}	2.84×10^{-2}
	3.0	7.93×10^{-2}	2.90×10^{-2}
	3.5	8.46×10^{-2}	2.94×10^{-2}
	4.0	8.93×10^{-2}	3.02×10^{-2}
LSSA (Up)	0.5	3.94×10^{-2}	2.97×10^{-2}
	1.0	4.45×10^{-2}	3.67×10^{-2}
	1.5	4.95×10^{-2}	4.33×10^{-2}
	2.0	5.38×10^{-2}	4.96×10^{-2}
	2.5	5.79×10^{-2}	5.55×10^{-2}
	3.0	6.18×10^{-2}	6.12×10^{-2}
	3.5	6.53×10^{-2}	6.67×10^{-2}
	4.0	6.90×10^{-2}	7.22×10^{-2}



PROBLEM

There is a large tradeoff between making an adversarial attack look believable and the amount of error we can inflict on the model.



Defence Mechanisms

Adversarial Training and Discriminators

- Adversarial training involves including the adversarial examples with the real data in the training set to effectively train the model to correctly classify similar inputs regardless of noise [8].
- As an alternative, a developer can create a discriminator aimed to distinguish adversarial and real data [8]. During inference, the discriminator is run and returns the likelihood of the input being altered. If the discriminator detects that the given input has been tampered with, predictions are not made.
- To test the stealthiness of the new attacks, a 4-layered CNN was trained on varying sets of adversarial methods.

GSA

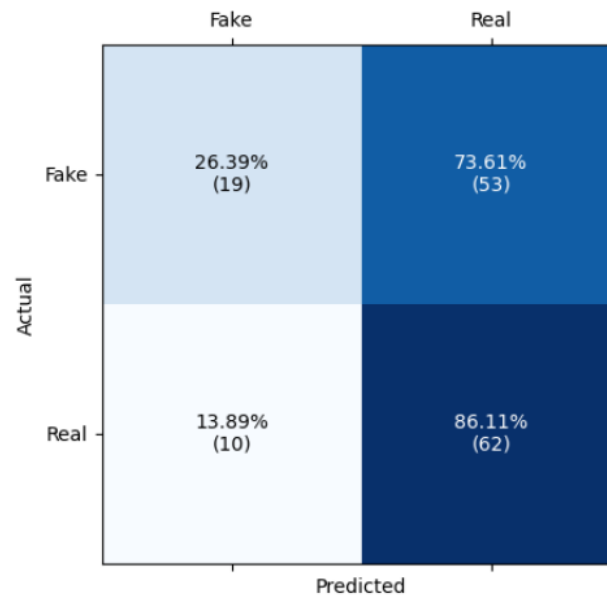
- The CNN is unable to identify adversarial inputs from real ones, with the model trained on GSA achieving an accuracy of 52.08, specificity of 27.78 and κ of 4.17.

		Fake	Real
Actual	Fake	27.78% (20)	72.22% (52)
	Real	23.61% (17)	76.39% (55)
		Predicted	

(a) GSA

LSSA

- The CNN is unable to identify adversarial inputs from real ones, with the model trained on LSSA achieving an accuracy of 56.25, specificity of 26.40 and κ of 12.50.

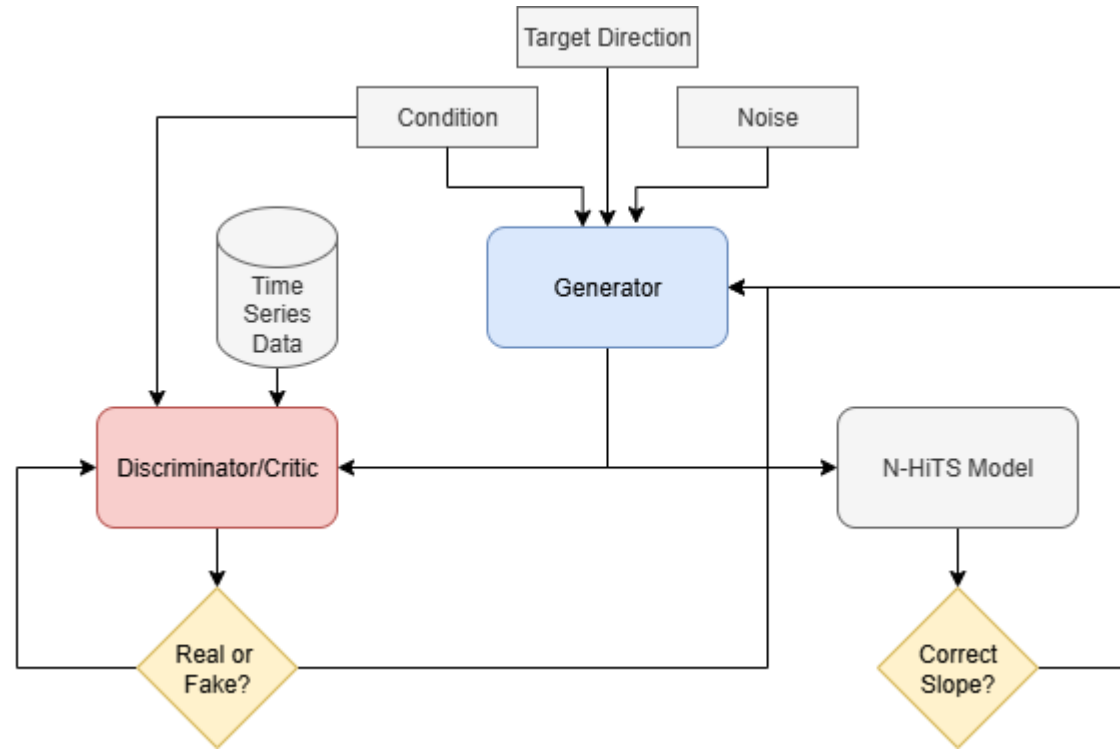


(b) LSSA

Can We Do Better?

Generative Adversarial Networks (GANs)

- Researchers have experimented with adding a second critic to the GAN architecture, which is the model that the attacker is aiming to fool [9].
- However, similar to adversarial attacks, the majority of research on GANs have been on images, and there have been even fewer studies using GANs to generate adversarial examples for time series forecasting [9], [11], [12].



Architecture of the Proposed Adversarial Gan (A-GAN)

A-GAN

- The A-GAN was trained with stock data extracted similar to the N-HiTS model, specifically using the stock with the ticker A; however, due to its stationary properties, the current implementation generates 99 days of log returns rather than adjprc.
- Random continuous intervals of 99 days were selected for each sample of training data. Furthermore, min-max scaling was applied on the training data.
- The noise vector is initialized to be the same size as the desired synthetic data (99 days) and is passed to the generator, made of a 4-layered Temporal Convolutional Network (TCN).
- The critic is a hybrid architecture with 5 layers, and is made up of alternating TCN and Gated Recurrent Network (GRU) blocks (3 TCN blocks, 2 GRU blocks) with a similar focus on long term dependencies. For both sub-models, a final linear layer was used to reduce the output dimension to 1.

A-GAN

- A Conditional Wasserstein GAN (C-WGAN) was used for the A-GAN, conditioned with the corresponding 99 days of log returns. The condition is then concatenated with the noise vector in the feature dimension for both the generator and the critic.
 - Thus, the critic learns that the provided data should be extremely similar to the condition, while the generator learns not to deviate too far from its condition.
- To incorporate the adversarial loss, the generator's objective was modified to:

$$L_g = \mathbb{E}_{\tilde{x} \sim P_g} [D(\tilde{x})] + \alpha \cdot LSSA(N-HITS(\tilde{x}))$$

with the final loss becoming:

$$L = L_g - \mathbb{E}_{x \sim P_r} [D(x)] + \lambda \cdot \mathbb{E}_{\hat{x} \sim P_{\hat{x}}} [(||\nabla_{\hat{x}} D(\hat{x})|| - 1)^2]$$

A-GAN

- Given that GAN training is notoriously difficult, the training was performed in batches of 50 epochs, with the α value occasionally increasing ([0.25, 0.25, 0.3, 0.35, 0.35]).
- Experiments have shown that increasing α too much would cause extremely unrealistic synthetic data.

A-GAN

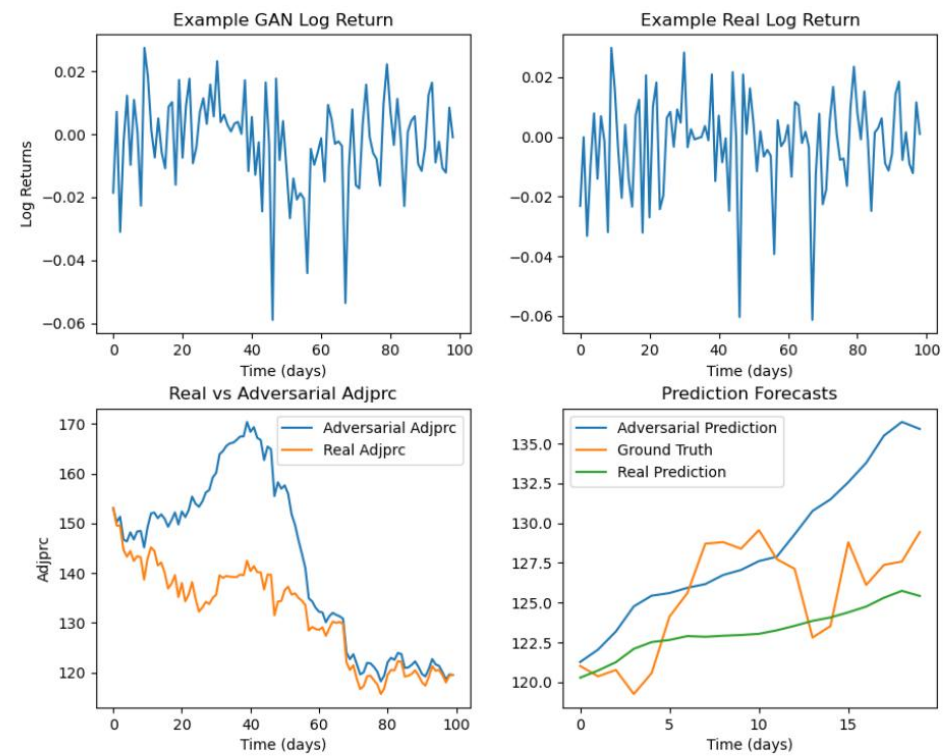


Figure 5: Example A-GAN output generated from a random interval.

A-GAN

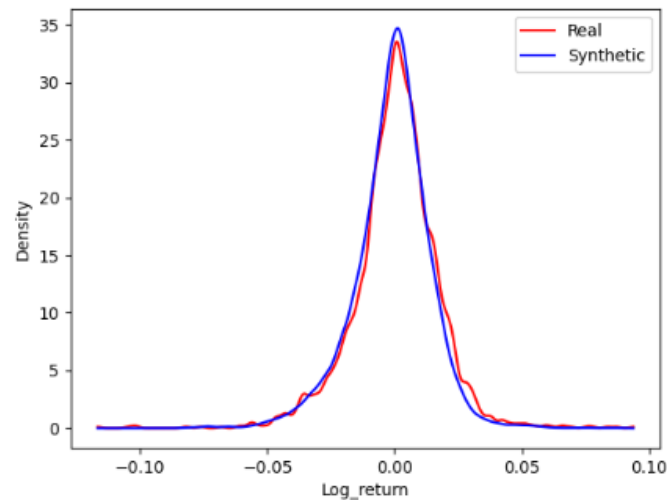
Table 3: Slope comparisons based on sampling the real and generated data 2000 times.

<i>Data</i>	<i>General Slope</i>	<i>LS Slope</i>
Real	0.999	-2.41×10^{-3}
A-GAN	1.050	2.17×10^{-1}

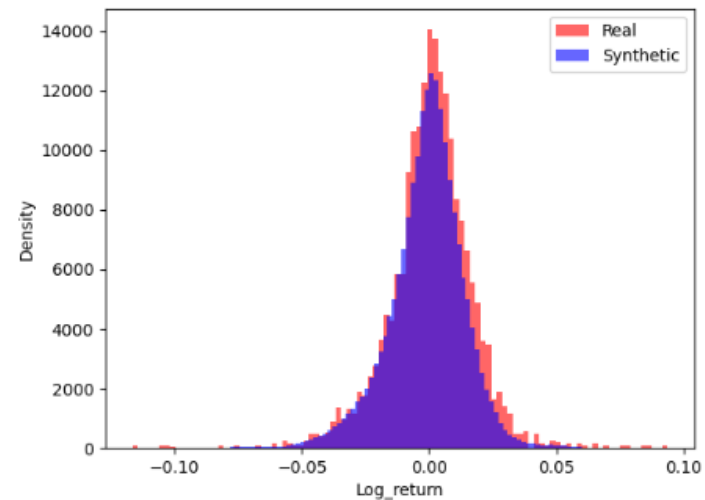
Table 4: Statistical metrics based on sampling the real and generated data 2000 times.

<i>Data</i>	μ	σ	<i>IQR</i>	<i>Skew</i>	<i>Kurtosis</i>	<i>MMD</i>
Real	5.71×10^{-4}	1.63×10^{-2}	1.82×10^{-2}	-3.81×10^{-1}	5.74	0
A-GAN	-9.85×10^{-4}	1.48×10^{-2}	1.72×10^{-2}	-4.26×10^{-1}	4.31	1.20×10^{-4}

A-GAN



(a) A-GAN KDE



(b) A-GAN Histogram

Figure 4: KDE and Histogram Plots of the distributions between the real and synthetic data generated by the A-GAN, after sampling 2000 intervals.

Mode Collapse

- The A-GAN suffers from mode collapse, which occurs when the generator creates data with limited diversity [19].

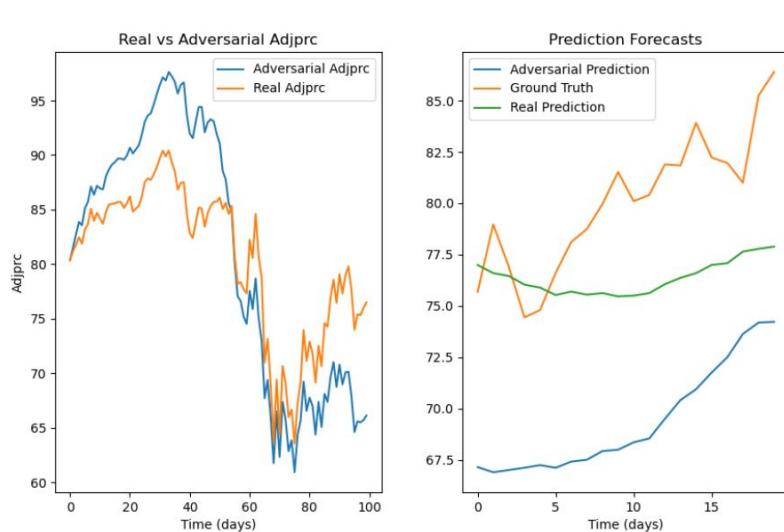


Figure 9: Example A-GAN output generated from a random interval.

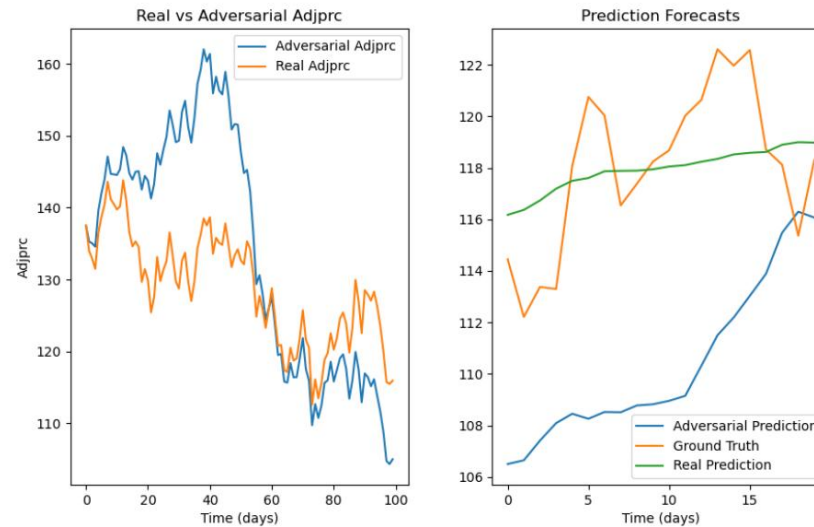


Figure 10: Example A-GAN output generated from a random interval.

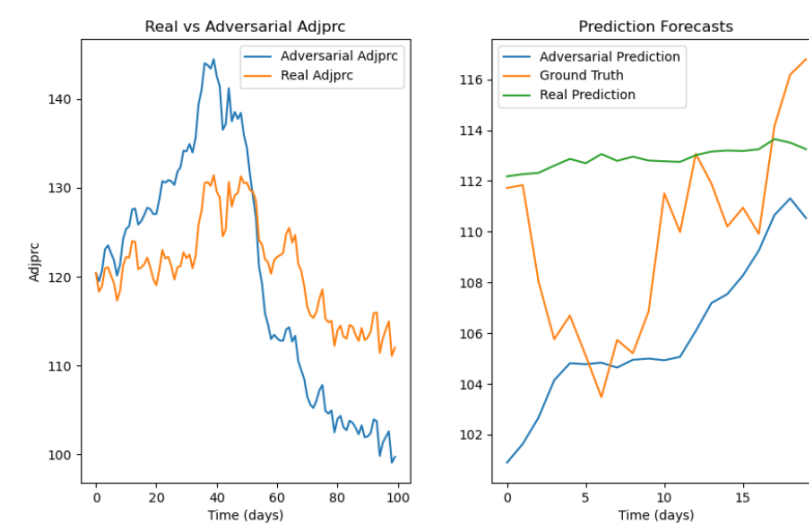
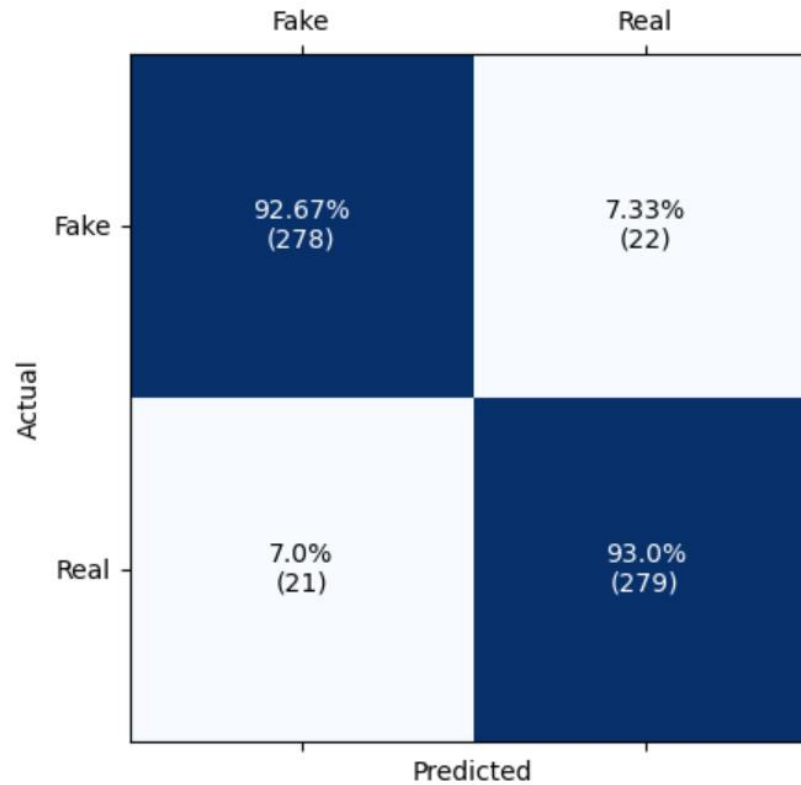


Figure 11: Example A-GAN output generated from a random interval.

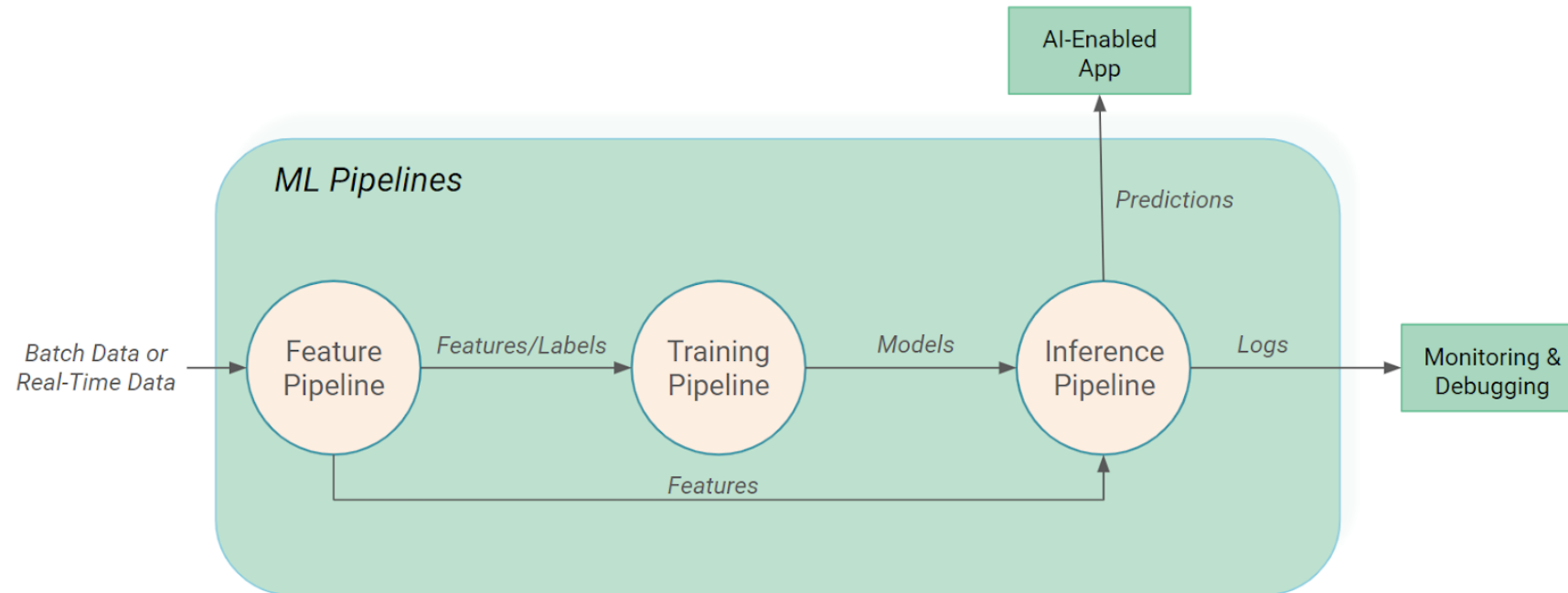
Easily Detectable....

Accuracy: 92.83
F1: 92.84
Specificity: 92.67
Kappa: 85.67



Can we Bypass the Defences?

The ML-Pipeline



ML-Security: Malware

- A common workflow for deployment is to have a project package/library that stores the saved model and all files required to make a prediction. In such cases, the prediction directory is thought to be safe and secure, yet can be easily manipulated with existing cyber-attack methods.
- For example, in Python, the `__init__.py` file is run whenever something inside the package is imported. However, if the `__init__.py` was tampered with beforehand, the attacker can modify any code inside the project directory.
- Therefore, this malware is run during every inference call and can:
 - Inject adversarial attacks.
 - Remove any gradient calls.

Trojan Malware

Algorithm 2 Sample payload for `__init__.py` file

```
1: adversarialString  $\leftarrow$  "Adversarial Attack Code"
2: originalFile  $\leftarrow$  fileWithModelCall.read()
3: originalFileLines  $\leftarrow$  fileWithModelCall.readlines()
4: f  $\leftarrow$  open(fileWithModelCall)
5: for line in originalFileLines do
6:   if line has a model call then
7:     write adversarialString
8:   end if
9:   if line has no_grad call then
10:    for each line with larger indent, remove indent and write
11:  else
12:    write line
13:  end if
14: end for
15: at exit, write originalFile to fileWithModelCall
```

Defences

- If the malware was used by an external attacker (someone who was not part of the model development process):
 - Hashing the entire directory coupled with package installers verifying the hash would prevent this type of malware.
- If the malware was used by an internal attacker (someone who was part of the model development process):
 - Code review?

Why Should We Care?

- Current research focuses on model robustness; however, it is significantly easier to attack other areas such as the model library with existing cyber-attack methods.
- The sample malware, shown in this research, demonstrates how an attacker could inject adversarial attacks that use gradient information, allowing for a stronger and more consequential attack.
- The internet, websites, and applications all came under attack once they became popular in the general public, and ML/AI models are approaching this stage, so it is imperative for more research to focus on securing the entire machine learning pipeline, given that no model in any domain is truly safe.

Limitations and Future Work

- In order to solidify and prove the general effectiveness of the attack methods, several other ML models, like CNNs and LSTMs, should have been developed and used as the victim models, similar the study by Shen and Li [3].
- Even though adversarial training is not necessarily feasible for the model in this study, it should still be performed to determine the effectiveness of the standard defense practice.
 - Adversarial training could be done by training a new model which does not rely on adjprc as a feature.
- Furthermore, to prove the applicability of the slope-based attacks on time series data, these attacks should be implemented in other time series domains that use forecasting models, such as traffic and electricity usage.
- Finally, given the A-GAN suffers from mode collapse, more experimentation should be performed to prevent it.

References

- [1] Y. Dong et al., “Boosting adversarial attacks with momentum,” in 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, IEEE, 2018. doi: 10.1109/CVPR.2018.00957.
- [2] I. J. Goodfellow, J. Shlens and C. Szegedy, “Explaining and harnessing adversarial examples,” 2014.doi: 10.48550/arxiv.1412.6572.
- [3] Z. Shen and Y. Li, “Temporal characteristics-based adversarial attacks on time series forecasting,” Expert systems with applications, vol. 264, no. 125950, 2025. doi: 10.1016/j.eswa.2024.125950.
- [4] N. Ghaffari Laleh et al., “Adversarial attacks and adversarial robustness in computational pathology,” Nature communications, vol. 13, no. 1, 2022. doi: 10.1038/s41467-022-33266-0.
- [5] M. Gallagher et al., “Investigating machine learning attacks on financial time series models,”Computers & security, vol. 123, no. 102933, 2022. doi: 10.1016/j.cose.2022.102933.
- [6] P. Rathore et al., “Untargeted, targeted and universal adversarial attacks and defenses on timeseries,” in 2020 International Joint Conference on Neural Networks (IJCNN), IEEE, 2020, pp. 1–8. doi: 10.1109/IJCNN48605.2020.9207272.
- [7] J.Zhang et al., “Are time-series foundation models deployment-ready? a systematic study of adversarial robustness across domains,” 2025. doi: 10.48550/arxiv.2505.19397.

- [8] G. Pialla et al., “Time series adversarial attacks: An investigation of smooth perturbations and defense approaches,” *International journal of data science and analytics*, vol. 19, no. 1, pp. 129–139, 2025. doi: 10.1007/s41060-023-00438-0.
- [9] J. Chen et al., “Mag-gan: Massive attack generator via gan,” *Information sciences*, vol. 536, pp. 67–90, 2020. doi: 10.1016/j.ins.2020.04.019.
- [10] J. Chen et al., “Time series data augmentation for energy consumption data based on improved timegan,” *Sensors*, vol. 25, no. 2, 2025. doi: 10.3390/s25020493.
- [11] L. Wang and K.-J. Yoon, “Psat-gan: Efficient adversarial attacks against holistic scene understanding,” *IEEE transactions on image processing*, vol. 30, no. 9524508, 2021. doi: 10.1109/TIP.2021.3106807.
- [12] S. Wu H. Sun and L. Ma, “Adversarial attacks on gan-based image fusion,” *Information fusion*, vol. 108, no. 102389, 2024. doi: 10.1016/j.inffus.2024.102389.
- [13] C. Challu et al., “N-hits: Neural hierarchical interpolation for time series forecasting,” 2022. doi:10.48550/arxiv.2201.12886.

[14] StockAnalysis. "A list of all stocks in the s&p 500 index." (2025), [Online]. Available: <https://stockanalysis.com/list/sp-500-stocks/> (visited on 06/15/2025).

[15] Y. Lai J. Zhang and J. Lin, "The day-of-the-week effects of stock markets in different countries," Finance research letters, vol. 20, pp. 47–62, 2017. doi: 10.1016/j.frl.2016.09.006.

[16] C. E. Appel, "Expanding ml-documentation standards for better security," 2025. doi: 10.48550/arxiv.2507.12003

[17] I. J. Goodfellow et al., "Generative adversarial networks," 2014. doi: 10.48550/arxiv.1406.2661.

[18] C. Esteban, S. L. Hyland, and G. R"atsch, "Real-valued (medical) time series generation with recurrent conditional gans," 2017. doi: 10.48550/arxiv.1706.02633.

[19] Z. Dai, L. Zhao, K. Wang, and Y. Zhou, "Mode standardization: A practical countermeasure against mode collapse of gan-based signal synthesis," Applied soft computing, vol. 150, no. 111089, 2024. doi: 10.1016/j.asoc.2023.111089

[20] J. Sen and S. Dasgupta, "Adversarial attacks on image classification models: Fgsm and patch attacks and their impact," 2023. doi: 10.48550/arxiv.2307.02055.

[21] A. Kurakin, I. Goodfellow, and S. Bengio, “Adversarial examples in the physical world,” 2016. doi:10.48550/arxiv.1607.02533.

[22] A. Radford, L. Metz, and S. Chintala, “Unsupervised representation learning with deep convolutional generative adversarial networks,” 2015. doi: 10.48550/arxiv.1511.06434.

[23] I. Gulrajani et al., “Improved training of wasserstein gans,” 2017. doi: 10.48550/arxiv.1704.00028.