A Comprehensive Evaluation of Watermarking for Time Series Diffusion Models

Vanessa Timmer

Delft University of Technology

TU Delft

A Comprehensive Evaluation of Watermarking for Time Series Diffusion Models

by

Vanessa Timmer

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Wednesday October 29, 2025 at 11:00 AM.

Student number: 4728033
Project duration: November 7, 2024 – October 29, 2025
Thesis committee: Dr. L. Y. Chen, TU Delft, Supervisor, Chair
J. M. Galjaard, TU Delft, Daily Supervisor
Dr. H. Wang, TU Delft, Committee Member

Cover: Generated by DALL-E

An electronic version of this thesis is available at http://repository.tudelft.nl/.

TU Delft

Abstract

Many fields rely on scarce and sensitive time series data, such as patient health records. Privacy regulations often make sharing such data challenging, slowing research progress. Synthetic time series offer a potential solution by replicating statistical characteristics of real data without revealing private information. Yet, they introduce new risks, as synthetic data may be mistaken for real. Watermarking can mitigate this by embedding a machine-detectable signal that preserves data quality. For such methods to be effective, watermarks must be robust to removal attempts. Existing research lacks direct comparisons of generative models for time series synthesis and watermarking. Furthermore, they only evaluate watermark robustness against time-domain attacks. Attacks in other domains, such as the frequency domain, remain unexplored. In order to address these gaps, this thesis investigates three key questions. First, which generative models are best suited for time series synthesis. Second, whether latent diffusion models (LDMs) can support watermarking. Lastly, how robust existing diffusion watermarks are against adversarial attacks.

A comparative study between GPT-based models and diffusion models showed that diffusion models produce synthetic data of higher quality. LDMs were then evaluated as a potential alternative. Their reliance on a variational autoencoder led to low quality outputs. Hence, standard diffusion models were elected as the superior watermarking candidate. Finally, we introduced an extended set of time-, frequency-, and time-frequency domain attacks to asses watermark robustness. TimeWak emerged as the most robust watermark. However, our extended attack suite revealed new vulnerabilities in all watermarks, highlighting the importance of comprehensive robustness evaluations.

Contents

1

Introduction

Time series data is used across a wide range of domains, including finance [28, 7, 63], medicine [13, 35, 6] and energy systems [3, 10, 25]. Yet, challenges related to data availability and privacy remain significant barriers to progress. Many time series datasets contain sensitive information, such as patient health records, that cannot be freely shared. Other domains lack sufficient labeled data for training robust models. Synthetic time series generated by AI models are becoming an increasingly popular solution for these problems. They preserve the statistical and temporal characteristics of real-world data while removing any sensitive or personally identifiable information. As such, they can facilitate open data sharing and privacy-preserving model training. Additionally, synthetic time series play a critical role in data augmentation, addressing the scarcity of labeled time series data compared to other modalities such as images or text. However, the proliferation of synthetic time series has given rise to several concerns. Without the ability to identify and trace synthetic data, we risk compromising research integrity, enabling misuse, and losing accountability. Watermarking techniques for synthetic time series offer a potential solution. These watermarks embed human-imperceptible yet computer-detectable signals into AI generated content. Watermarks fulfill two key functions. First, they allow us to distinguish between real and synthetic data. By doing so, they contribute to the continued integrity of scientific research, industry applications, and model evaluation. Second, they provide accountability in the event of misuse. This is particularly important as synthetic datasets are increasingly being shared across organizations.

For watermarks to be effective in practice, robustness to modifications is paramount. A watermark that disappears under simple post-processing or noise has limited utility. Yet, much of the existing watermarking research focuses primarily on detectability in clean conditions. Even when they evaluate robustness against removal attempts, they only consider simple attacks. Realistic data manipulations are often overlooked. In real-world deployments, synthetic data often undergoes scaling, compression, filtering, or domain-specific transformations that could alter or erase embedded watermarks. The robustness of a watermark to such attacks ultimately determines its viability as an identification and traceability tool.

Despite the importance of robustness, current work on time series watermarking remains limited. At the time of writing, only a single watermarking method designed specifically for synthetic time series exists: TimeWak [55]. TimeWak is designed for diffusion models that operate directly in the data domain. Prior evaluations of TimeWak and other general-purpose watermarking schemes (e.g., TreeRing [67] and Gaussian Shading [70]) consider only simple time domain attacks such as cropping or mean shifts. These assessments provide little insight into how watermarks behave under more sophisticated, domain-spanning perturbations, such as Fourier or wavelet transformations, that are common in real data pipelines. This gap in robustness evaluation limits our understanding of watermark reliability for real-world time series applications. To address this, this thesis develops and applies a comprehensive robustness evaluation framework for time series watermarks. The framework extends beyond prior work by introducing attacks that operate in the frequency and joint time-frequency domains. The attack suite also includes more sophisticated temporal perturbations inspired by established data aug-

mentation techniques. We use this framework to assess the resilience of three watermarks (TimeWak, TreeRing, and Gaussian Shading) across four benchmark datasets and multiple attack intensities.

Before we can evaluate robustness, we first need to ensure that the watermark can embed a detectable yet imperceptible signal. Watermark performance is often intertwined with model characteristics. Watermarking can cause slight degradation in generation quality, so models must preserve high synthetic data quality despite this trade-off. As such, the choice of generative model is crucial. To generate high-fidelity synthetic time series, various generative modeling approaches have been explored. Among the most established are Variational Autoencoders (VAEs), such as TimeVAE [14], and Generative Adversarial Networks (GANs), such as TimeGAN [72]. More recently, transformer-based models and diffusion models have demonstrated strong performance in time series generation. Models such as Chronos [2], Moirai [68], TSGM [44] and Diffusion-TS [73] have shown they can generate high quality synthetic time series without any of the drawbacks typically associated with VAEs and GANs. In many fields the generative modeling landscape is shifting toward latent diffusion models (LDMs) [54], which have demonstrated impressive results in image [46, 48, 77] and video generation [41, 4, 80]. This trend is beginning to influence time series research as well, as seen in emerging models like TimeAutoDiff [60] and TimeLDM [49]. At present, no studies directly compare generative models for time series to determine their suitability for watermarking.

Accordingly, this thesis investigates existing time series watermark techniques. We examine their weaknesses and strengths across different generative model architectures and attack scenarios. We focus on three main research questions:

RQ1 Which generative model architecture is more effective for time series synthesis, transformer-based models or diffusion models?

RQ2 To what extent can latent diffusion models provide a viable framework for embedding robust watermarks in synthetic time series?

RQ3 To what extent are existing watermarking techniques for time series resilient against sophisticated post-editing attacks in both the time and frequency domains?

2

Background - Time Series Synthesis

A time series is a sequence of observations recorded over time, typically at regular intervals [31]. What sets time series data apart from other data types is its inherent temporal structure. Each observation depends on preceding values, and reordering the sequence would disrupt the underlying dynamics. This temporal ordering allows for the detection of patterns that only emerge across extended horizons, such as long-term growth trends or recurring seasonal cycles. Time series may be univariate, focusing on a single variable such as stock prices, unemployment rates, or temperature readings. They may also be multivariate, involving multiple variables observed together, as in macroeconomic indicators, air pollution monitoring or patient vital signs.

Time series data can be decomposed into four key components: trend, seasonality, cyclicity, and irregularity. The trend captures the long-term progression of the data, whether upward, downward, or stable. Seasonality reflects systematic and recurring fluctuations tied to fixed time intervals. Cyclicity refers to longer, less regular oscillations that often reflect business or economic cycles. Finally, irregularity represents the unpredictable noise in the data, often stemming from random shocks or measurement errors.

Applications of time series generally fall into two categories: analysis and forecasting. Analysis seeks to extract insights from historical data, such as identifying structural changes, quantifying risk, or revealing dependencies across series. Forecasting, by contrast, projects future values based on historical patterns. This enables planning and decision-making under uncertainty. Forecasts may be short-term or long-term, with accuracy and efficiency generally declining as the horizon extends [69]. These applications rely on the availability of time series data, and the quality of analysis or forecasting depends strongly on its quantity. However, in many domains data is scarce, as measurements cannot always be collected at the rate or scale required to achieve robust performance. Moreover, security and privacy regulations often restrict the use of real-world data. To mitigate these limitations, synthetic time series have been proposed as a promising alternative.

2.1. Synthetic Time Series

With the growing adoption of artificial intelligence in many domains[45, 61, 62], the demand for synthetic time series has grown significantly. Deep learning and related methods require extensive amounts of training data, yet access to real-world datasets is often restricted. Privacy and security regulations such as the GDPR [52] place strict limits on how sensitive data can be used, stored, and shared. Additionally, many organizations, including hospitals and financial institutions, work with information that is highly confidential. As a consequence, they are reluctant or legally unable to provide access to their data. Even when regulatory and privacy barriers are not the main issue, data scarcity often is.

One promising way to address these limitations is through the use of synthetic time series. These are time series designed to replicate real-word time series in terms of statistical characteristics and

temporal correlations. Unlike random or purely simulated data, high-quality synthetic time series must behave as credible stand-ins for actual datasets. This requires replicating not just correlations between variables but also the dynamic properties that unfold over time, including seasonality, long-term trends, and dependencies across multiple time steps. For example, in financial markets, a realistic synthetic time series of stock prices should reflect volatility clustering, where volatile periods are followed by further volatility [61]. These domain-dependent peculiarities highlight the need for specialized generative models that can learn complex dependencies not limited to independent and identically distributed (i.i.d.) assumptions.

Synthetic time series provide several notable benefits. By enabling organizations to train models without disclosing sensitive raw data, they allow for better privacy protection. They also reduce costs and time associated with data collection, since synthetic data can be generated using a computer and can be tailored to the problem at hand. Finally, they make it possible to introduce rare but important cases into training datasets, ensuring that models are exposed to critical edge scenarios that may otherwise be absent. These advantages make synthetic time series a valuable tool for creating robust, privacy-preserving, and scalable AI models.

2.1.1. Traditional Methods

Statistical and programming techniques formed the foundation of early time series generating systems. These techniques generate data that replicates known patterns by using a set of preset rules, mathematical formulas, or simulation models. A straightforward example involves combining a linear trend with seasonal components and random noise using simple mathematical equations. Despite being reproducible and predictable, these techniques lack flexibility and have trouble capturing the intricate, non-linear dynamics of real-world data.

To address these shortcomings, a more sophisticated family of models was developed, namely the Autoregressive Integrated Moving Average (ARIMA) models [5]. ARIMA is a statistical model used for time series forecasting and analysis. It consists of three components: Autoregression (AR), Integrated (I), Moving Average (MA). ARIMA models are transparent and excel at capturing linear trends and well-defined seasonal patterns, making them a useful benchmark for more complex models. However, their reliance on linearity is a significant limitation. They cannot effectively model the complex, non-linear dependencies present in many time series, and are also unable to capture properties unique to certain domains. These drawbacks spurred a search for more advanced, non-linear models.

2.1.2. GAN and VAE

The shortcomings of traditional models lead to the adoption of deep learning models. Deep learning models are capable of learning the intricate, non-linear processes that generate complex time series data. Generative Adversarial Networks (GAN) [20] and Variational AutoEncoders (VAE) [29] in particular were used for this purpose.

GANs introduced an adversarial training framework with two neural networks. One of those networks is a generator network that learns to produce realistic synthetic time series. The other is a discriminator network that learns to distinguish between the real and synthetic time series. The goal of the generator is to produce synthetic time series capable of fooling the discriminator into classifying a fake time series as real. Early adaptations for time series included TimeGAN [72], which incorporated both supervised and adversarial losses to preserve temporal dynamics. Unlike the statistical methods used before, GANs are able to learn complex, non-linear patterns without explicit modeling assumptions. They can capture intricate temporal correlations, irregular patterns, and domain-specific characteristics that would be difficult to express manually. However, they also face several challenges such as unstable training and mode collapse. The former making them difficult to train, and the latter causing low variety in the model's output.

VAEs offer an alternative generative approach. Section 2.2 discusses them in more detail. In short, VAEs learn to encode time series into a structured latent space. First, the input is mapped to a latent

distribution. After which, the latent is decoded to the original data space. A latent parameterization trick combines the encoder decoder structure in to a stochastic one. After training, the model permits efficient sampling of the learned data distribution. However, VAEs typically produce less sharp outputs compared to GANs due to their reconstruction-based training objective. Moreover, they also struggle with capturing fine-grained temporal details and can suffer from a posterior collapse.

Both GANs and VAEs represented significant advances over statistical methods by automatically learning complex temporal patterns from data. Breaking with the explicit reliance on predefined mathematical relationships requirement. Instead, establishing deep learning as a viable approach for synthetic time series tasks. Despite these advances, there remained significant obstacles [40] due to training instability and preserving time-series properties. Recent developments in diffusion models and large-scale foundation models aim to improve upon these limitations.

2.1.3. Diffusion Models and Foundation Models

The most recent advances in synthetic time series generation have been driven by two developments: diffusion models [23] and large-scale, token-based foundation models [38]. These approaches have addressed many of the persistent challenges faced by GANs and VAEs while introducing new capabilities for high-quality synthetic time series. Unlike the adversarial training of GANs or the variational objectives of VAEs, diffusion models learn to reverse a gradual noise corruption process. The reversal starts from pure noise and iteratively refines towards realistic samples. TimeGrad [51] was among the first to successfully adapt diffusion models for time series forecasting and generation, demonstrating superior performance in capturing temporal dependencies. More recently, Diffusion-TS [73] incorporated techniques to handle variable-length sequences and multivariate dependencies for general time series generation. Diffusion models have been shown to generate samples of high quality, while having stable training dynamics without adversarial optimization. However, they require significantly more computational resources during inference due to the iterative denoising, making them slower than other generators.

Another new development in the field is large-scale foundation models for time series. These models leverages advances in the domain of natural language processing, but are adapted to handle numerical time series data. Due to their massive scale, foundation models enable learning from diverse time series across multiple domains, leading to robust representations that capture both universal temporal patterns and domain-specific characteristics. Furthermore, these models support few-shot and zero-shot generation, where high-quality synthetic data can be produced for new domains with minimal or no training examples. Models like TimeGPT [18] and Moirai [68] demonstrate how autoregressive language modeling objectives can be directly applied to time series prediction and generation.

However, like other methods, foundation models have drawbacks as well. The discrete nature of language model training may not fully represent the continuous, numerical properties of time series data. Maintaining precise statistical properties, such as exact means, variances, or distributions, can be difficult when generation is mediated through learned vocabularies or embedding spaces. Additionally, the computational requirements for training and inference with foundation models are substantial, potentially limiting accessibility compared to smaller, specialized models.

2.2. Variational Autoencoders

Variational Autoencoders (VAEs) [29] represent an approach to generative modeling that combines the reconstruction capabilities of standard autoencoders with probabilistic inference. Unlike standard autoencoders that learn deterministic mappings, VAEs introduce a probabilistic framework that enables both data reconstruction and generation of new samples. Traditional autoencoders consist of an encoder network that maps input data x to a latent representation z, and a decoder network that reconstructs the input from this latent space. While effective for dimensionality reduction and feature learning, standard autoencoders struggle with generative tasks because their latent space lacks structure and continuity. Their encoder learns to map similar inputs to arbitrary, potentially distant points in the latent

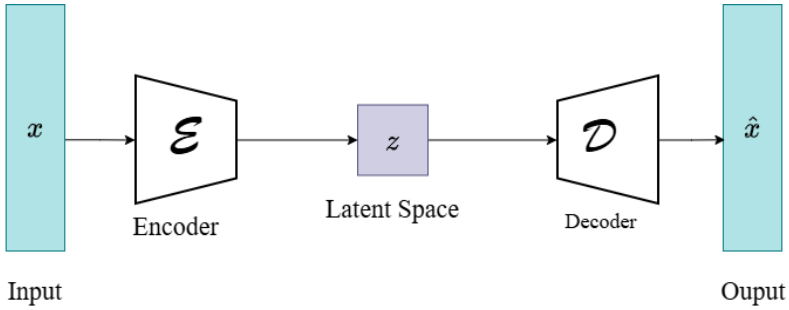


Figure 2.1: The general structure of a VAE. The encoder maps input x to a latent representation z. The decoder constructs the latent representation z into output $\hat{x}$.

space. Consequently, random sampling from the latent space typically produces poor reconstructions. VAEs address this limitation by imposing a probabilistic structure on the latent space.

Instead of learning a deterministic encoding $z = f(x)$, VAEs learn to encode inputs as probability distributions over the latent space. This probabilistic approach ensures that the latent space is continuous and well-structured, enabling meaningful interpolation and generation of new samples. Figure 2.1 showcases the general structure of a VAE. A typical VAE consists of two networks: an encoder and a decoder. The **encoder** $q_\phi(z|x)$ maps input x to latent distribution parameters $\mu_\phi(x)$ and $\sigma_\phi(x)$. The **decoder** $p_\theta(x|z)$ maps latent samples z back to the data space.

VAEs are based on variational inference, where we approximate intractable probability distributions by finding the closest tractable distribution. Given observed data x and latent variables z, the goal is to maximize the marginal likelihood:

$$p(x) = \int p(x|z)p(z)\,dz$$

However, this integral is typically intractable for complex models since it would require integrating over all possible values of z. VAEs avoid this complexity, by approximating the true distribution with a learnable one. Specifically, using the variational approximation $q_\phi(z|x)$ to the true posterior $p(z|x)$, where ϕ parametrizes the encoder network. The VAE objective is derived from the Evidence Lower Bound (ELBO). This objective consists of two terms. First, a reconstruction term that encourages accurate reconstruction of the input by maximizing the likelihood of reconstruction. Second, a regularization term that constrains the approximate posterior to be close to the prior $p(z)$, typically chosen as a standard Gaussian $\mathcal{N}(0, I)$. The ELBO provides a tractable lower bound on the log marginal likelihood:

$$\log p(x) \geq \mathcal{L}(\theta, \phi; x) = \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] - D_{KL}(q_\phi(z|x)\|p(z)),$$

. Here, θ parametrizes the decoder network $p_\theta(x|z)$. D_{KL} denotes the Kullback-Leibler divergence [33], serving as the regularization term.

A key innovation in VAEs is the **reparameterization trick**, which enables gradient-based optimization of the ELBO. Normally, this would be impossible due to the expectation $\mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)]$, which involves sampling from $q_\phi(z|x)$. There is no way to differentiate μ_ϕ and σ_ϕ when z is obtained by random sampling. Hence, direct sampling would break the gradient, making backpropagation impossible. The reparameterization trick addresses this by expressing the random variable z as a deterministic function of the parameters and an auxiliary noise variable. For a Gaussian posterior $q_\phi(z|x) = \mathcal{N}(\mu_\phi(x), \sigma^2_\phi(x)I)$, we can write:

$$z = \mu_\phi(x) + \sigma_\phi(x) \odot \epsilon, \quad \epsilon \sim \mathcal{N}(0, I)$$

where $\odot$ denotes element-wise multiplication. This transformation moves the stochasticity to the noise variable ϵ, allowing us to compute gradients with respect to $\mu_\phi(x)$ and σ_ϕ.

The training loss combines reconstruction and regularization,

$$\mathcal{L}_{VAE} = \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] - \beta \cdot D_{KL}(q_\phi(z|x)\|p(z)),$$

where β is a hyperparameter that controls the trade-off between reconstruction quality and latent space regularity. Once trained, VAEs sample $z \sim p(z) = \mathcal{N}(0, I)$ from the prior, then generate $x \sim p_\theta(x|z)$ using the decoder. This process produces new samples that should follow the learned data distribution. The probabilistic nature of VAEs results in a smooth latent space where interpolation between points in the latent space produces meaningful intermediate samples. However, VAEs struggle with blurry outputs. Furthermore, the wrong choice of variational posterior may be too restrictive to fully capture the true posterior distribution, limiting the model's expressiveness. Another common challenge of VAEs is one where the model ignores the input, causing the variational posterior to collapse to the prior. This is referred to as posterior collapse.

2.3. Generative Pre-trained Transformer

The Generative Pre-trained Transformer (GPT) architecture [50] has become the cornerstone of modern large language models (LLMs). They are not constrained to specific data types or domains, making them a powerful tool for diverse sequential modeling applications [71]. Their generalizability stems from the architecture's ability to learn abstract representations of sequential patterns. Time series are one of the sequential data types GPT models have proven capable of processing [68, 2, 43].

To understand how GPT models process data, it is essential to first understand its way of representing data. To model data of arbitrary length, it creates a list of representations for parts of the data. These representations are referred to as tokens, which a learned tokenizer generated from some input. In text, this could be a word, a stem, or even individual characters, depending on the tokenization scheme. For time series data, a token might represent a single data point or a patch of consecutive values. The model processes these tokens sequentially to learn patterns and make predictions.

GPT processes this sequence of tokens using autoregressive generation. The autoregressive property ensures that predictions at each time step depend only on previously observed values. GPT models leverage this autoregressive mechanism to generate coherent sequences through iterative prediction. They repeatedly sample one token at a time until a stopping condition is reached, yielding a coherent sequence of data. The GPT architecture builds upon the Transformer block [65]. GPT utilizes only the decoder component, unlike the full Transformer architecture which consists of both encoder and decoder components. The original Transformer encoder processes input sequences to create contextual representations, while the decoder generates output sequences. By using only the decoder, GPT models become suitable for autoregressive generation.

Formally, the autoregressive property can be expressed using the conditional probability. The conditional probability of a sequence $s = (s_1, s_2, \ldots, s_T)$ is expressed as:

$$p(s) = \prod_{t=1}^{T} p(s_t \mid s_{<t}),$$

where $s_{<t}$ denotes all tokens before position t. This expression turns the task of modeling the joint probability of an entire sequence into a series of simpler conditional prediction problems. Training a GPT model therefore amounts to maximizing the likelihood of observed tokens under this autoregressive expression.

GPT models are accessible for applications with limited data availability. They require significantly less computational resources and training data compared to training from scratch. Training GPT is typically carried out in two stages. First it learns to 'complete' a sequence of tokens with the next one. During this stage, it pre-trains on large volumes of data to perform next token prediction. This approach uses a self-supervised objective that does not require human-labeled data. By learning to predict the next token given the vast training dataset, the model implicitly learns the most general statistical properties in the training data. The second stage is fine-tuning. During this stage, the pre-trained model is fine-tuned on specific downstream tasks by training it on a smaller, more task-specific dataset.

2.4. Diffusion Models

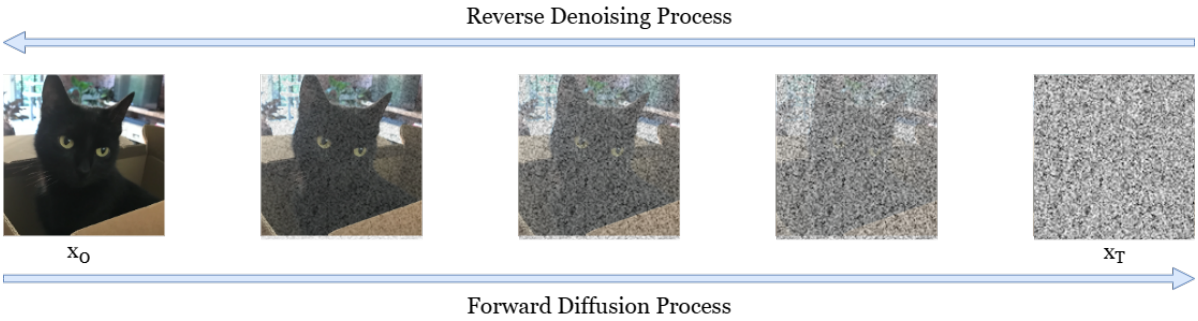


Figure 2.2: Overview of the forward diffusion process and reverse denoising process in DDPMs. The forward diffusion process gradually adds Gaussian noise to the original data x_0 until reaching pure noise x_T. The reverse denoising process learns to iteratively remove the noise from x_T to reconstruct x_0.

Diffusion models [23] are a class of generative models that learn to synthesize data by simulating and reversing a gradual noising process. Recent work has shown that diffusion models achieve state-of-the-art performance in generative tasks, including image [47, 24, 79] and time series generation [73, 36, 60, 49]. For time series generation, the forward and reverse processes operate on time series data, and specialized architectures (e.g., Transformers) are used to capture dependencies across time. This enables time series diffusion models to generate synthetic time series in which temporal coherence is maintained.

Several variants of diffusion models exist, such as score-based generative models that learn the gradient of the data distribution directly [57]. However, this section focuses primarily on Denoising Diffusion Probabilistic Models (DDPM). For this reason, DDPM models will often be referred to as ”diffusion models” or ”standard diffusion models”. Diffusion models define a forward diffusion process that slowly destroys structure in the data by adding noise over a series of time steps, and a reverse process that reconstructs data samples from pure noise. Figure 2.2 provides a visualization of both the forward and reverse diffusion processes.

Formally, the forward diffusion process adds Gaussian noise to data x_0 over T discrete time steps. At each step, the transition is defined as

$$q(x_t \mid x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}\, x_{t-1},\, \beta_t I),$$

where β_t is a small variance term controlling the amount of noise added at step t. The choice of noise schedule $\{\beta_t\}_{t=1}^{T}$ influences the training of the diffusion model.

The schedule must satisfy several constraints: β_t values should be small to ensure gradual noise addition, increase monotonically over time, and result in $\bar{\alpha}_T \approx 0$ so that x_T approximates pure Gaussian noise.

Common schedules include linear schedules where β_t increases linearly from a small value (e.g., 10^{-4}) to a larger one (e.g., 0.02), and cosine schedules that provide more gradual transitions. The schedule directly affects training stability and generation quality, with poorly chosen schedules potentially leading to difficulty generating coherent samples [42].

The noisy sample x_T at any step can be sampled directly from the original data x_0 using a closed-form expression:

$$x_t = \sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{1-\bar{\alpha}_t}\, \epsilon, \quad \epsilon \sim \mathcal{N}(0, I),$$

where $\bar{\alpha}_t = \prod_{s=1}^{t}(1-\beta_t)$. This reparameterization is mathematically possible due to the properties of Gaussian distributions. Since each forward step adds Gaussian noise, and the Central Limit Theorem states that the sum of Gaussian random variables is also Gaussian [34], we can derive a closed-form expression for $q(x_t|x_0)$. Specifically, by repeatedly applying the transition $q(x_t|x_{t-1})$ and using the

reparameterization trick, we obtain:

$$q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)I)$$

This allows us to sample x_t directly from x_0 during training without computing all intermediate steps, which is computationally efficient. The coefficient $\sqrt{\bar{\alpha}_t}$ controls how much of the original signal remains, while $\sqrt{1-\bar{\alpha}_t}$ determines the noise level, ensuring that the total variance remains constant.

The generative task is to learn the reverse diffusion process, which transforms Gaussian noise back into a data sample with the original distribution, removing the noise that was added during the forward process. The reverse diffusion process is modeled as

$$p_\theta(x_{t-1} \mid x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t),\, \Sigma_\theta(x_t, t)),$$

where a neural network parameterized by θ is trained to predict either the noise ϵ added at each step or directly the denoised data. Predicting the noise is usually the preferred approach since the noise ϵ has a fixed, known distribution $\mathcal{N}(0, I)$, whereas the the denoised data can have complex distributions that are difficult to learn. For a given noisy sample x_t at timestep t, the network is trained to predict $\epsilon_\theta(x_t, t)$. Once we have this prediction, we can take a small step backwards towards a slightly less noisy sample x_{t-1}. The training objective is a mean-squared error loss between the actual noise added in the forward process and the network's prediction:

$$\mathcal{L}(\theta) = \mathbb{E}_{x_0,\epsilon,t}\left[\left\|\epsilon - \epsilon_\theta(x_t, t)\right\|^2\right].$$

Minimizing this loss ensures that the model learns to gradually denoise x_t, reconstructing clean samples from noise over multiple iterations. During generation, the process begins from pure Gaussian noise x_t. The model then iteratively applies the learned reverse process until a synthetic sample x_0 is obtained. Although this required many steps in the early days of diffusion models, subsequent works have introduced new sampling methods to reduce the number of steps without significantly compromising sample quality.

Diffusion models have demonstrated great generative capabilities, but DDPM comes with several drawbacks. The requirement for many sequential denoising steps during sampling presented a significant computational bottleneck. The stochastic nature of DDPM also makes it impossible to retrieve the original noise vector at the start of diffusion, limiting its use in novel techniques such as watermarking. New adaptations have since been developed to tackle these limitations. One of these adaptations is Denoising Diffusion Implicit Models (DDIM) [56], which introduced deterministic sampling procedures that enable both faster generation and exact inversion capabilities.

2.4.1. DDIM

DDPM is computationally expensive and does not allow for retrieval of the initial diffusion noise. Denoising Diffusion Implicit Models (DDIM) [56] is one of the developments designed to address the limitations of DDPM. It introduces a deterministic sampling procedure that enables high-quality generation with fewer steps. DDIM treats the forward diffusion process as a family of inference distributions, rather than requiring sequential progression through every timestep. By removing the requirement for step-by-step transitions, the inference process no longer needs to go through every timestep and can instead skip intermediate timesteps while maintaining the same training objective as DDPM. Depending on the number of timesteps skipped this can greatly reduce computational resources necessary for inference.

Formally, DDIM defines the reverse process as:

$$x_{t-1} = \sqrt{\bar{\alpha}_{t-1}}\left(\frac{x_t - \sqrt{1-\bar{\alpha}_t}\epsilon_\theta(x_t, t)}{\sqrt{\bar{\alpha}_t}}\right) + \sqrt{1-\bar{\alpha}_{t-1}-\sigma_t^2}\cdot\epsilon_\theta(x_t, t) + \sigma_t\epsilon_t$$

where $\epsilon_t \sim \mathcal{N}(0, I)$ and σ_t control the stochasticity. The parameter σ_t in particular controls the degree of stochasticity during sampling:

- When $\sigma_t^2 = \tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$, sampling is equivalent to DDPM
- When $\sigma_t = 0$, sampling becomes completely deterministic

The deterministic case ($\sigma_t = 0$) enables DDIM to perform exact inversion. Given a data sample x_0, DDIM can deterministically reverse each sampling step to obtain the corresponding noise representation x_T, and then reconstruct the original sample exactly. The deterministic reverse process simplifies to:

$$x_{t-1} = \sqrt{\bar{\alpha}_{t-1}} \left(\frac{x_t - \sqrt{1-\bar{\alpha}_t}\epsilon_\theta(x_t, t)}{\sqrt{\bar{\alpha}_t}} \right) + \sqrt{1-\bar{\alpha}_{t-1}} \cdot \epsilon_\theta(x_t, t)$$

The corresponding forward inversion process, which maps from x_{t-1} to x_t, is given by:

$$x_t = \sqrt{\bar{\alpha}_t} \left(\frac{x_{t-1} - \sqrt{1-\bar{\alpha}_{t-1}}\epsilon_\theta(x_{t-1}, t-1)}{\sqrt{\bar{\alpha}_{t-1}}} \right) + \sqrt{1-\bar{\alpha}_t} \cdot \epsilon_\theta(x_{t-1}, t-1)$$

This inversion process is theoretically exact because each step is a deterministic function. Starting from a sample x_0, we can apply the forward inversion iteratively to obtain $x_1, x_2, \ldots, x_T$, where x_T represents the noise vector of the original sample. Applying the deterministic reverse process to this x_T will reconstruct the original x_0 exactly.

The training procedure for DDIM remains identical to DDPM, using the same noise prediction objective:

$$\mathcal{L}(\theta) = \mathbb{E}_{x_0, \epsilon, t}\left[\left\| \epsilon - \epsilon_\theta(x_t, t) \right\|^2 \right]$$

This means that a single trained model can be used for both DDPM and DDIM sampling, providing flexibility and efficiency.

2.4.2. BDIA

A major limitation of standard DDIM inversion is the inconsistency between intermediate diffusion states generated during forward and backward processes. This inconsistency is caused by approximations in the sampling procedure. While DDIM should theoretically enable perfect recovery of the original Gaussian noise after inversion, in practice these inconsistencies accumulate, leading to inexact recovery. This inexactness compromises applications requiring precise noise recovery, such as watermark detection, where even small deviations can lead to failure. Bi-directional Integration Approximation (BDIA) [74] addresses this limitation by introducing a more robust approach to maintaining consistency between forward and backward diffusion processes.

The key property of BDIA is that its update expression for x_{t-1} is formulated as a linear combination of x_{t+1}, x_t, and the estimated Gaussian noise $\hat{\epsilon}(x_t, t)$. This formulation enables exact backward computation of x_{t+1} given the pair (x_t, x_{t-1}). BDIA defines the reverse process as

$$x_{t-1} = \gamma(x_{t+1} - x_t) - \gamma\left(\frac{x_t}{a_{t+1}} - \frac{b_{t+1}}{a_{t+1}}\hat{\epsilon}_\theta(x_t, t) - x_t \right) + \left(a_t x_t + b_t \hat{\epsilon}_\theta(x_t, t)\right),$$

where the coefficients a_t and b_t are defined as $a_t = \alpha_{t-1}/\alpha_t$ and $b_t = \sigma_{t-1} - \sigma_t \alpha_{t-1}/\alpha_t$, and γ is a scaling parameter.

The corresponding inverse process is expressed as

$$x_{t+1} = \frac{x_{t-1}}{\gamma} - \frac{1}{\gamma}\left(a_t x_t + b_t \hat{\epsilon}_\theta(x_t, t)\right) + \left(\frac{x_t}{a_{t+1}} - \frac{b_{t+1}}{a_{t+1}}\hat{\epsilon}_\theta(x_t, t) \right)$$

While this should theoretically result in a more accurate inversion than regular DDIM, a practical drawback of BDIA is that, in most applications, only x_0 is available, whereas the inversion process requires access to both x_0 and x_1. Soi et al. [55] propose addressing this constraint through a simple yet effective approximation: setting $x_1 = x_0$. Despite its simplicity, this approximation proves remarkably accurate in practice and maintains the enhanced inversion properties of BDIA.

2.4.3. Latent Diffusion Models

Standard diffusion models operate directly in the data space, requiring the denoising network to process the input at their full dimensionality at each timestep. For high-dimensional data, this can lead to extremely expensive training and inference costs. Latent Diffusion Models (LDMs) [54] address these computational bottlenecks by performing the diffusion process in a lower-dimensional latent space rather than directly on the data. Much of the semantic information in high-dimensional data can be captured in a compressed representation, allowing the diffusion process to focus on learning the essential generative patterns without the computational cost of standard diffusion models.

LDMs consists of an encoder-decoder pair from a pre-trained VAE [29], and a diffusion model operating in the latent space produced by the VAE's encoder. The VAE encoder $\mathcal{E}$ maps input data x to a lower-dimensional latent representation $z = \mathcal{E}(x)$. The corresponding decoder $\mathcal{D}$ reconstructs the data from the latent space: $\tilde{x} = \mathcal{D}(z)$. The forward diffusion process in LDMs operates on these latent representations:

$$q(z_t \mid z_{t-1}) = \mathcal{N}(z_t; \sqrt{1-\beta_t}, z_{t-1}, \beta_t I),$$

where $z_0 = \mathcal{E}(x_0)$ is the encoded version of the original data. This preserves the mathematical foundation of standard diffusion models while reducing computational requirements. The training objective becomes:

$$\mathcal{L}(\theta) = \mathbb{E} z_0, \epsilon, t \left[\left| \epsilon - \epsilon\theta(z_t, t) \right|^2 \right],$$

where the denoising network ϵ_θ now operates on latent vectors rather than the original data space.

During generation, the process begins with Gaussian noise z_T in the latent space. The trained denoising network iteratively removes this noise to produce a clean latent representation z_0, which is then decoded using the VAE decoder to obtain the final sample $x_0 = \mathcal{D}(z_0)$. The quality of this final sample, and the performance of the LDM as a whole, strongly relies on the quality of the VAE. The encoder must preserve sufficient semantic information in the latent space, while the decoder must accurately reconstruct high-quality outputs from the latent space. Poor reconstruction quality in the VAE directly limits the generation quality of the entire LDM, as the diffusion model can only be as good as the latent space it operates within.

3

Related Works

The proliferation of AI-generated content has created a need for accountability and traceability mechanisms. As synthetic media becomes increasingly sophisticated, distinguishing it from human-created content has become critical. Tracing synthetic content back to its source is essential for combating misinformation and protecting intellectual property. It also ensures responsible AI deployment. Watermarking represents a promising approach to address these challenges.

Generative AI watermarking is a technique that embeds a detectable yet imperceptible signal into content created by AI models [75]. The purpose of this watermark is to denote synthetic data as AI-generated without compromising the quality of the synthetic data. In many cases, it also enables attribution to a specific model or source. Watermarking techniques are domain-agnostic and can be adapted across content types. Examples of data modalities which have adopted watermarks are images [9, 11, 76, 53] and time series [55]. This chapter proceeds in two parts. We first provide an overview of the stages in AI content generation where watermarks may be embedded. We then present three watermarking approaches for AI models: Gaussian Shading [70], TreeRing [67] and TimeWak [55]. For each approach, we explain its embedding and detection mechanisms.

However, developing effective watermarks presents significant hurdles. The fundamental challenge lies in balancing three competing objectives: detectability, imperceptibility and robustness. A detectable watermark should be reliably identifiable by a detection algorithm so it can easily be distinguished from real data. An imperceptible watermark is not noticeable to humans and does not degrade the quality of the content, which is important for preserving the functional integrity of the generated output. A robust watermark can withstand various common transformations and attacks, such as compression, cropping and scaling. A robust watermark ensures that the watermark can still be detected even after the content has been modified. These objectives can conflict with each other. Increasing robustness typically requires embedding a stronger signal. However, this risks becoming perceptible and degrading content quality. Conversely, subtle watermarks that preserve quality are often less detectable. They are also more vulnerable to even benign transformations. The optimal balance among these trade-offs depends on the specific application. Certain use cases may prioritize robustness to modifications over imperceptibility. Others require near-perfect quality preservation.

3.1. Watermarks for Generative AI Models

Watermarking techniques for generative AI can be categorized based on when the watermark is embedded in the generation pipeline. The three primary approaches differ in their integration point: during model training, during sampling, or after content has been produced. Each approach presents trade-offs between robustness, flexibility and computational cost.

Training-phase watermarks embed the watermark directly into the model's parameters during or at the end of the training process, creating models that inherently produce watermarked content [37]. The

integration of the watermark into the model parameters provides strong robustness against removal attempts, as removal is challenging without modifying the model itself [9]. This approach also ensures that the watermark is present across all content generated by the model. Furthermore, training-time watermarks are learned as part of the generation process rather than being imposed on existing content. As a result, they can potentially achieve better imperceptibility compared to post-processing methods [70]. Herein also lies a drawback of this type of watermark, though. Improper integration of the watermark during training could introduce unintended bias or artifacts in the generated content, decreasing model performance and limiting its applicability [30]. Another drawback is that the strong coupling between model and watermark means that the model requires retraining when the watermark is updated. This could carry a substantial computational cost, especially for large models. And while the method is robust, it is not immune to attack. Recent attacks [8] have been shown to successfully remove training-time watermarks by employing sophisticated fine-tuning. Hence, the design of *robust* domain-specific watermarking methods has become an actively researched topic.

Sampling-time watermarks strike one of the most favorable balances between detectability, imperceptibility, robustness, and efficiency. These watermarks operate by modifying the generation process during sampling, embedding detectable patterns directly into the output [70]. This approach can be applied to any pre-trained model. It is often regarded as a middle ground between training-time watermarks, which offer robustness to attacks but require retraining, and post-processing watermarks, which provide flexibility but are more vulnerable. Sampling-time watermarking can be implemented as a modular component of an existing generation pipeline, requiring minimal changes to the model's architecture. A key design objective is to ensure that the watermark remains both imperceptible and detectable. This is typically achieved by constraining the sampling distribution under watermarking to remain as close as possible to the original distribution [11]. Notable examples of sampling-time watermarks that achieve this include the distortion-free watermark introduced by Kuditipudi et al. [32], which preserves the data distribution by mapping watermarked random numbers to language model samples. Other examples are Gaussian Shading [70] and TimeWak [55], which embed signals in diffusion models while maintaining the Gaussian distribution of the initial noise during sampling. Despite their advantages, sampling-time watermarks remain vulnerable to tampering and modifications. Sampling-time watermarks for language models, like their post-processing counterparts, are especially fragile under modification [30]. Diffusion-based watermarks, while more stable, can still be disrupted as shown in Section 6.

Finally, we discuss **post-processing watermarks**. Post-processing watermarking applies watermarks to content that has already been generated, similar to traditional digital watermarking techniques [21]. The ability to retroactively apply the watermark is particularly valuable. Users of AI models can apply watermarks as needed without requiring changes to their generation pipeline. Furthermore, post-processing watermarks can be easily updated or modified without retraining models, offering an efficient alternative to training-time watermarks. Despite their flexibility, these watermarks face several limitations. One of the difficulties in developing post-processing watermarks is the trade-off between robustness and imperceptibility. Post-processing modifications are applied to existing content. Therefore, they may be more noticeable than watermarks embedded during generation or sampling. Numerical data such as time series can be particularly challenging for post-processing watermarks. This type of data often contains important inter-dependencies that make it highly sensitive to even minor alterations [55]. Thus, post-processing watermarks should take care to preserve essential properties. Robustness against removal attacks also present a significant concern, as adversaries can apply various techniques to corrupt or remove the watermark. Simple modifications have been shown to remove post-processing watermarks, indicating the need for more robust post-processing methods [21].

3.2. Existing Watermarks for Diffusion Models

Diffusion models have achieved state-of-the-art performance in image synthesis [54] and time series synthesis [73]. Their success has raised concerns about content authenticity, provenance tracking, and potential misuse of generated outputs. To address these concerns while preserving generation quality, researchers have developed watermarking techniques specifically tailored to diffusion mod-

els. Among these approaches, sampling-time watermarking has emerged as a particularly promising solution. Sampling-time watermarks embed their signal directly into the generation process without requiring model retraining or post-processing modifications. This section examines three representative sampling-time watermarking techniques for diffusion models: Gaussian Shading [70], TreeRing [67] and TimeWak [55]. Gaussian Shading embeds watermarks through distribution-preserving noise sampling. TreeRing operates in the frequency domain of the initial noise. TimeWak manipulates the temporal and feature dimensions for time series generation.

3.2.1. Gaussian Shading

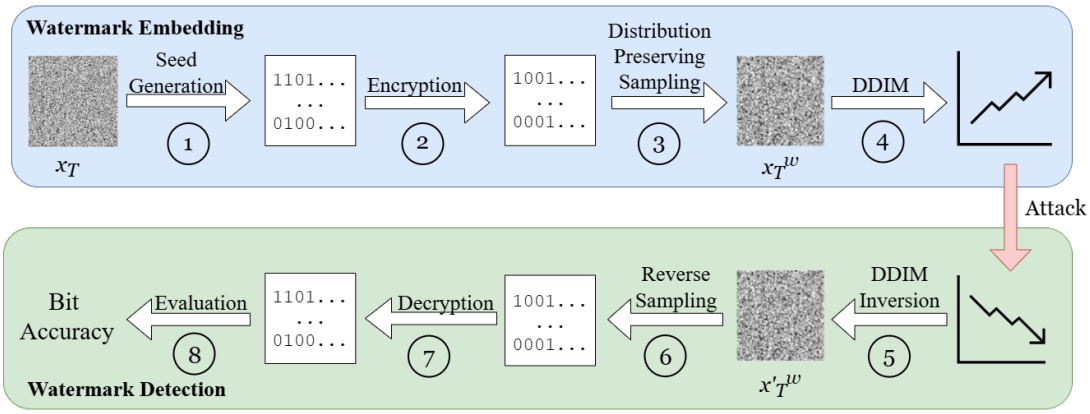


Figure 3.1: An overview of the Gaussian Shading pipeline. 1. First a seed is generated. 2. The seed is encrypted. 3. Distribution preserving sampling according to the seed forms the initial diffusion latent with watermark. 4. DDIM sampling. 5. DDIM inversion to reconstruct the starting noise. 6. Reverse sampling to extract the encrypted watermark. 7. Seed decryption. 8. Seed is compared with ground-truth watermark to calculate bit accuracy.

Gaussian Shading [70] is a recent watermarking method for latent diffusion models. It embeds information directly into the model's latent sampling stage while keeping the statistical properties of the generated outputs unchanged. Although the original work proposes a watermark for latent diffusion models, it permits adaptation to *data diffusion* as well.

Figure 3.1 provides an overview of the Gaussian Shading pipeline from embedding to detection. (1) During embedding, the watermark is first represented as a binary sequence. To make it robust against transformations, a watermark diffusion step replicates this sequence across every dimension of the data. As a result, the watermark is distributed throughout the entire noise vector. Since the distribution of the diffused watermark is not directly known, Gaussian Shading applies watermark randomization by (2) encrypting the diffused watermark with a secure stream cipher. This produces a randomized binary sequence that is uniformly distributed and statistically indistinguishable from random noise. (3) The randomized watermark then guides distribution-preserving sampling. The Gaussian distribution $\mathcal{N}(0, I)$ is split into intervals with equal probability, each corresponding to a possible watermark bit value. For each position in the initial diffusion noise, Gaussian Shading samples only from the interval matching the watermark bit. This preserves the overall Gaussian distribution exactly. Consequently, the resulting watermarked values are statistically equivalent to unwatermarked ones. (4) After this, the standard denoising process and decoding are used to generate the final output.

(5) To detect a watermark, Gaussian Shading applies DDIM inversion to an image to approximately recover its initial noise x'^w_T. (6) Through reverse sampling, each noise value is then mapped back to its corresponding bit. (7) The resulting bit sequence is decrypted with the original stream key to recover the diffused watermark. (8) Finally, the recovered watermark bit sequence is compared with

a ground-truth watermark to compute a bit accuracy score.

3.2.2. TreeRing

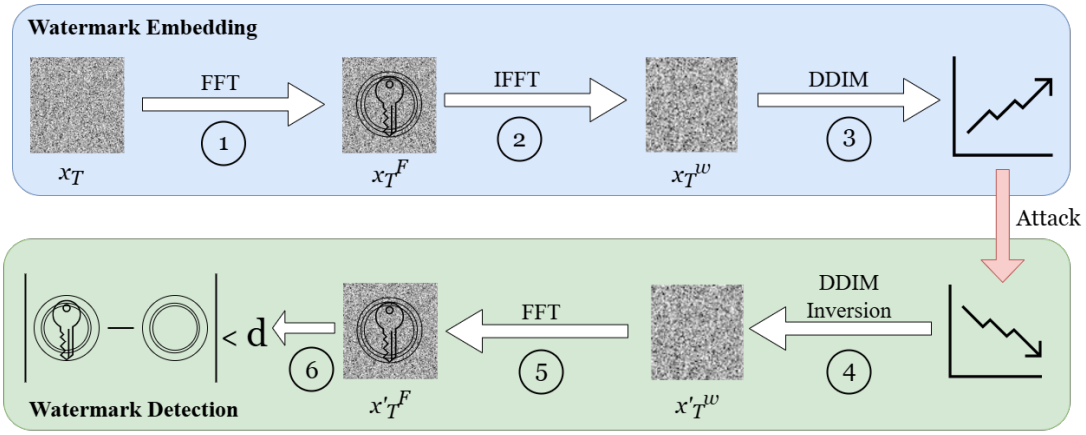


Figure 3.2: An overview of the TreeRing pipeline. 1. FFT is applied to initial Gaussian noise to inject watermark in FFT domain. 2. IFFT is performed to return to data domain. 3. DDIM sampling. 4. DDIM inversion to recover initial noise. 5. FFT is applied to extract watermark. 6. Extracted watermark is compared with ground truth patch.

TreeRing [67] is a watermarking method for diffusion models that embeds information directly into the frequency domain of the initial Gaussian noise. The main idea behind this method is to embed a pattern into the noise in a way that remains imperceptible after decoding, yet is recoverable through frequency analysis. Unlike other methods which embed the watermark in the time domain, the TreeRing watermark is embedded in the frequency domain.

The complete pipeline is depicted in Figure 3.2. The embedding process begins with the model's initial noise tensor x_T. (1) This noise is transformed to the frequency domain with a Fast Fourier Transform (FFT). A fixed-frequency pattern called the ground-truth watermark patch is generated that will serve as the watermark signal. A corresponding watermarking mask is then computed, identifying the frequency domain locations where the watermark should be applied. (2) Watermark injection is performed by perturbing the values at the masked locations with the watermark patch values. Afterwards, the frequency representation of the initial noise is transformed back to the time domain, producing the watermarked noise x_T^w. (3) The watermarked noise vector is then returned to the generative process for denoising and decoding.

Detection of the TreeRing watermark starts by (4) taking an image and applying inverse sampling to recover its approximate starting Gaussian noise. (5) The recovered noise is transformed into the frequency domain with a Fourier transform, centering the low-frequency components. Using the same watermarking mask as in embedding, the detection process isolates the frequency bins where the watermark was originally applied. The target watermark patch, identical to the one used during embedding, is also regenerated. (6) The extracted frequency coefficients from the recovered noise are then compared against the target patch using a predefined distance metric. This comparison is computed only over the masked locations. Smaller values indicate closer alignment with the original watermark pattern, suggesting the presence of the watermark. This approach enables reliable verification while remaining compatible with standard diffusion generation pipelines.

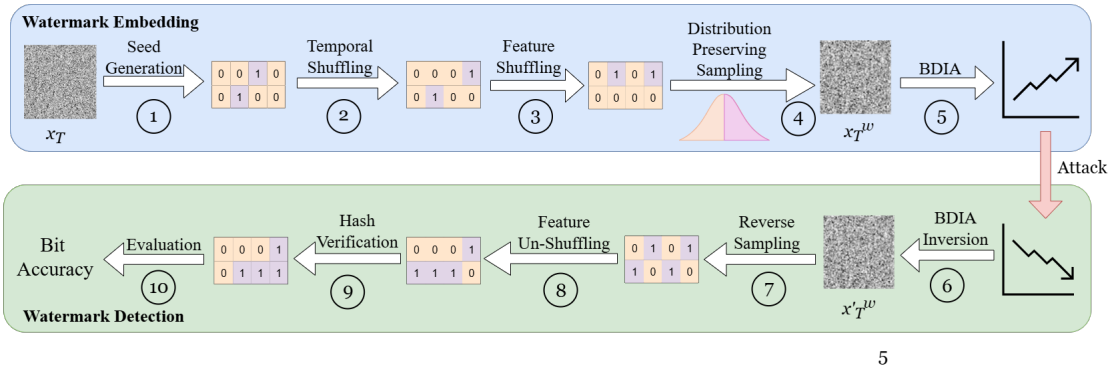


Figure 3.3: An overview of the TimeWak pipeline. 1. First a seed is generated. 2. Temporal shuffling. 3. Feature shuffling. 4. Distribution preserving sampling according to seed to form watermarked noise. 5. BDIA sampling. 6. BDIA inversion to reconstruct initial noise. 7. Reverse sampling to extract watermark. 8. Feature un-shuffling. 9. Hash verifications. 10. Compute bit accuracy between recovered watermark and hash

3.2.3. TimeWak

TimeWak [55] is a watermarking method for diffusion models that embeds information by manipulating the temporal and feature-wise structure of the initial noise. Unlike watermarking methods that focus on images generated by latent diffusion models, TimeWak was designed to operate directly on time series without first transforming them to a latent representation. Furthermore, it relies on BDIA [74] sampling rather than DDIM [56] to improve watermark recovery.

The embedding process begins (1) by generating a binary bit tensor. A temporal permutation pattern is then applied (2). At even-indexed time steps, the bits are permuted independently across the feature dimension using a deterministic random seed tied to the time index. At odd time steps, the bits are instead permuted to mirror the arrangement from the preceding time step, encoding temporal relationships between consecutive steps. Next (3), a feature-wise permutation step is performed. For each feature index, the bits are permuted along the time dimension using a deterministic seed tied to the feature index. This two-stage permutation process creates a complex, repeatable pattern that can be exactly reversed during detection, provided the seeding scheme is known. (4) Once the bit pattern is finalized, it is embedded into the Gaussian distribution using a distribution-preserving quantile mapping. The standard normal distribution is split into two equal-probability halves: one corresponding to bit value 0 and the other to bit value 1. This preserves the global Gaussian distribution while encoding the watermark in the sign distribution of the initial noise. (5) Inference then proceeds with BDIA sampling.

(6) To extract the watermark, the BDIA sampling process is inverted to recover the initial noise from the sample. A median-based threshold is computed along the time dimension. Each value is assigned a bit value of 0 or 1 depending on whether it falls below or above the threshold. This produces a recovered bit tensor matching the shape of the original. The detection process then reverses the embedding permutations (7). Feature-wise permutations are inverted first (8), followed by timestep-wise permutations (9). (10) The reconstructed bit sequences can then be compared against the expected temporal permutation relationships, resulting in a bit accuracy score.

4

GPT vs. Diffusion for Synthetic Time Series Generation

For watermark embedding to be effective during data generation, the generated data must be of high quality. This requirement arises because the watermark introduces a bias into the generator, potentially leading to a degradation in output quality. As a result, if the generator already produces low-quality data, watermarking will further diminish its usefulness. Furthermore, watermarking a generative model only makes sense if the model is capable of producing synthetic data that is indistinguishable from real data to both humans and statistical detectors. If a model consistently produces synthetic samples that are clearly unrealistic or systematically different from real time series, then its lack of fidelity essentially functions as a watermark. Consequently, any additional watermarking would become unnecessary. To ensure that watermarking is both meaningful and effective, it is therefore essential to identify generative models capable of producing high-quality synthetic data.

Recently, both diffusion- and transformer-based models have shown to be capable generators for sequential data. As such, recent research [2, 68, 73, 44] has explored their application towards time series data generation. While these two architectures for time-series synthesis have been well studied individually, a comparative study is lacking for these methods. Moreover, at the time of writing, no transformer based *generation method* exists. Hence, this chapter pursues two main goals. First, we provide a general method towards time-series synthesis with probabilistic transformer-based models. Second, we perform a comparative study of the two generative methods, with state-of-the-art representative backbones for each. Through this comparative study, we aim to identify the model that most consistently produces high-quality synthetic time series. In turn, this allows us to determine which model is the more appropriate candidate for watermarking.

4.1. GPT for Time Series Synthesis

GPT models have achieved strong performance on a variety of time series tasks [1, 18]. These models are typically designed to operate in a conditional setting, where predictions are generated based on a fixed input window consisting of prior timesteps. In the context of forecasting, the model learns to predict future values conditioned on a sequence of observed past values, i.e., completions. However, unconditional generation imposes that no context is available to complete, requiring additional steps to generate data with a forecaster. To the best of our knowledge, currently no unconditional transformer-based time series generators exist. As such, we propose to adapt a forecasting model for this purpose.

Our goal is to evaluate the ability of transformers to produce fully synthetic time series in an unconditional setting. The model should generate plausible time series from scratch, without conditioning on real historical data. Given the lack of transformer-based unconditional models, we now describe our generation method. From a high level, it is an iterative method to build sequences using autoregres-

sive predictions. At each iteration, the method performs three steps. First, we leverage the forecasting backbone to make multiple probabilistic predictions. Second, we compute a quality score for each candidate prediction. Lastly, we rank the samples according to our scoring mechanism to select the next completion before proceeding to the next iteration.

We now provide the specifics of this generation process, as outlined in Algorithm 13.

Step 1: The generation process begins by providing the model with a short initial context window drawn from real data. This initial context provides the model with sufficient historical information to make its first prediction while minimizing dependence on real data for the majority of the generated sequence. The generation proceeds through repeated application of the forecasting model in a sliding window fashion. Once the model has generated the first prediction window, we append those outputs to the current context to form the new context window and repeat the forecasting step. In this way, we iteratively construct a full-length time series by sliding the model's prediction window forward, decomposing the long-sequence generation task into a series of short-term forecasting problems.

Step 2: Since the forecasting model is probabilistic, it produces multiple stochastic samples rather than a single point estimate. To select which sample to append to the context, we leverage Dynamic Time Warping (DTW) distance as our quality metric. DTW measures the temporal alignment between two sequences and is robust to phase shifts and temporal distortions, making it well-suited for comparing time series with similar patterns but different timing. By computing the DTW distance between each candidate sample and the corresponding ground truth window, we can identify the prediction that best preserves temporal coherence.

Step 3: We rank all candidate samples by their DTW distance and select the sample with the minimum distance to append to the context window. This sample then becomes part of the context for the next forecasting iteration.

Algorithm 1 Moirai Time Series Synthesis

Input: Dataset D, Context length C, Prediction length P, Target length L, Num samples N
Output: Generated time series X_{synth}

Load dataset and extract first sequence as ground truth X_{real}
Initialize context: $X_{synth} \leftarrow X_{real}[:, :C]$
Initialize forecasting model with context length C and prediction length P
while *length*$(X_{synth}) < C + L$ **do**
    Create dataset from current synthetic series X_{synth}
    Generate N forecast samples: $\{S_1, S_2, \ldots, S_N\} \leftarrow$ Moirai(X_{synth})
    `// Select best sample using DTW distance to ground truth`
    for $i = 1$ *to* N **do**
        Compute DTW distance: $d_i \leftarrow$ DTW$(S_i, X_{real}[\text{next } P \text{ steps}])$
    end
    $i^* \leftarrow \arg\min_i d_i$
    `// Append best prediction to synthetic series`
    $X_{synth} \leftarrow$ concatenate(X_{synth}, S_{i^*})
end
return X_{synth}

For our experiments, we selected Moirai-Large [68], a recently proposed Transformer-based model for zero-shot time series forecasting. We chose this model primarily because it supports multivariate time series, which is essential for our benchmark datasets. Moirai is designed to forecast future time steps in the near future by encoding a context window of past values using a patch-based attention mechanism. It generates probabilistic predictions over the forecast horizon. Since Moirai was not originally intended for unconditional sequence generation, we modified its use at inference time to produce long, synthetic time series, simulating unconditional generation. We attempted fine-tuning Moirai on each dataset to improve generation quality, but observed no meaningful difference in performance compared to the pretrained model.

While this auto-regressive sampling approach enabled us to repurpose Moirai for long sequence generation, it is important to note the limitations caused by using a forecasting model for this task. Moirai enables short-term forecasting, its performance on longer time horizons is not studied. The model's predictions also become increasingly dependent on its own previous outputs due to the way previous predictions are appended to the context window. As a result, errors can accumulate over successive autoregressive steps, potentially degrading the realism and temporal coherence of the generated sequences. Second, the DTW-based sample selection improves temporal realism during evaluation. However, it may not accurately represent how the model would behave in fully unconditional generation scenarios, where no ground truth data are available for comparison. Finally, the sliding windows may not capture long-range dependencies that extend beyond the context window length. Nonetheless, this adaptation provided a reasonable baseline for evaluating the viability of existing GPT models for synthetic time series generation.

4.2. Diffusion Model for Time Series Synthesis

While transformer-based models generate time series sequentially, diffusion models take a different approach by learning to iteratively denoise corrupted data. Broadly speaking, diffusion models for time series synthesis can be categorized into two main types: standard diffusion models and latent diffusion models. Vanilla diffusion models operate directly in the data space, learning to denoise time series in their original form. By contrast, latent diffusion models use a learned representation (i.e., latent), in which they perform the generation. Generally, an Auto-Encoder like structure is used, to provide the latent space for the diffusion process. While latent diffusion offers significant computational advantages for high-dimensional data such as images [54] or multivariate time series [60], it is a more recent development. To the best of our knowledge, there are no state-of-the-art latent diffusion models for time series synthesis at the moment of writing. Hence, we focus exclusively on standard diffusion models in this experiment, specifically evaluating Diffusion-TS [73], a recent data-diffusion model for time series synthesis.

Unlike GPT models, which are typically optimized for conditional downstream tasks such as forecasting, classification, or imputation, diffusion models are often trained to unconditionally generate synthetic sequences. However, diffusion models have also been adapted for downstream tasks by modifying the denoising objective or conditioning mechanism. These extensions make diffusion models increasingly versatile, but in the time series domain they remain most commonly used for generation. Due to its unconditional generation capability, we train a Diffusion-TS model on each dataset using recommended hyper-parameters.

4.3. Methodology

Table 4.1: Overview of datasets used, including the total number of features and their types (Pollution is only used for the experiment in Section 5).

Dataset	Total # of Features	Numerical Features	Categorical Features
ETTh	7	7	0
Energy	27	25	2
fMRI	49	49	0
Stocks	6	6	0
Pollution	8	6	2

To evaluate the quality of synthetic time series generated by the models, we leverage four common time-series benchmarks. Specifically, we select ETTh1 (electricity load), Stocks (financial time series), Energy (power consumption), and fMRI (neural signals), to consider a range of domains and statistical characteristics: This diversity allows us to assess the generalizability of each generative model across different types of time series. Table 4.1 provides an overview of key characteristics of these datasets.

The generated synthetic datasets were evaluated using four common quality metrics used in synthetic time series research. These metrics were chosen to capture different dimensions of generation quality: statistical similarity, predictive usefulness, temporal coherence, and inter-variable relationships. By combining discriminative, predictive, contextual, and correlational evaluations, we aim to determine not only whether the synthetic sequences look realistic, but also whether they behave like real time series in practical downstream tasks and structural dependencies. Together, these metrics provide a comprehensive view of the generative performance of each model. All metrics were computed independently for each dataset and each model, and results were aggregated to compare overall performance.

- **Discriminative score** [72]: Measures the similarity between real and synthetic data using a binary classification task. A model is trained to classify time series as either real or synthetic. A high-quality synthetic time series should cause the model to erroneously classify it as real. A lower score means the model struggles to distinguish between real and synthetic data.
- **Predictive score** [72]: Evaluates temporal dynamics and practical usefulness by training on synthetic data and testing a downstream task on real data. In this case, the downstream task is forecasting. Close alignment between the forecasts and real data indicates that the synthetic data possesses similar temporal characteristics to the real data. Hence, it can serve as a viable substitute for the real data. The Predictive score is measured using Mean Absolute Error (MAE), with lower values being better.
- **Context-FID score** [27]: Evaluates how closely synthetic data fits local contexts. It captures both distributional similarity and temporal coherence. Lower scores indicate better fidelity.
- **Correlational score** [39]: Measures errors between cross-correlation matrices of real and synthetic data. Lower values indicate that the synthetic data captures the inter-variable relationships of the real data more accurately.

4.4. Results

Table 4.2: Results of synthetic time series generated by Moirai and Diffusion-TS. It lists mean scores along with the standard deviation in brackets. All results use the non-watermarked version of Diffusion-TS. For all metrics, lower scores are better. Best results are shown in bold.

Dataset	Model	Discriminative	Predictive	Context-FID	Correlational
ETTh	Moirai	0.454 ± (0.016)	3.849 ± (0.000)	13.221 ± (2.024)	0.683 ± (0.053)
	Diffusion-TS	**0.095** ± (0.004)	**0.123** ± (0.002)	**0.198** ± (0.012)	**0.068** ± (0.005)
Stocks	Moirai	**0.073** ± (0.007)	349672.614 ± (0.000)	5077523276055.157 ± (1712315826271.960)	1.037 ± (0.040)
	Diffusion-TS	0.100 ± (0.007)	**0.037** ± (0.000)	**0.209** ± (0.028)	**0.015** ± (0.012)
Energy	Moirai	0.442 ± (0.036)	56.162 ± (0.000)	758.871 ± (85.525)	12.444 ± (0.256)
	Diffusion-TS	**0.145** ± (0.005)	**0.251** ± (0.000)	**0.082** ± (0.015)	**0.930** ± (0.125)
fMRI	Moirai	0.306 ± (0.021)	1.771 ± (0.000)	4095.063 ± (443.670)	16.000 ± (0.395)
	Diffusion-TS	**0.110** ± (0.015)	**0.100** ± (0.000)	**0.191** ± (0.006)	**2.615** ± (0.076)

Table 4.2 provides a qualitative overview of the two data-generators. Figure 4.1 provides a visual comparison of time series sequences generated by Moirai and Diffusion-TS. Generally, we see that Diffusion-TS significantly outperforms our adapted Moirai across nearly all datasets and metrics. The only exception is the discriminative score on the Stocks dataset, where Moirai slightly edges out Diffusion-TS. In all other cases, Diffusion-TS achieves lower scores, indicating better alignment with real data in terms of predictive performance, distributional similarity, and inter-variable structure.

Diffusion-TS achieves considerably lower scores on ETTh, Energy, and fMRI, suggesting it produces data distributions that are harder to distinguish from the real ones. The only case where Moirai performs better is the Stocks dataset (0.073 vs. 0.100). However, this anomaly may not reflect genuine generative quality. Notably, Moirai's predictive and context-FID scores for Stocks are extraordinarily high, indicating poor temporal and structural quality. This discrepancy suggests that the synthetic data from Moirai may include extreme or inconsistent values that confuse the classifier, thereby artificially

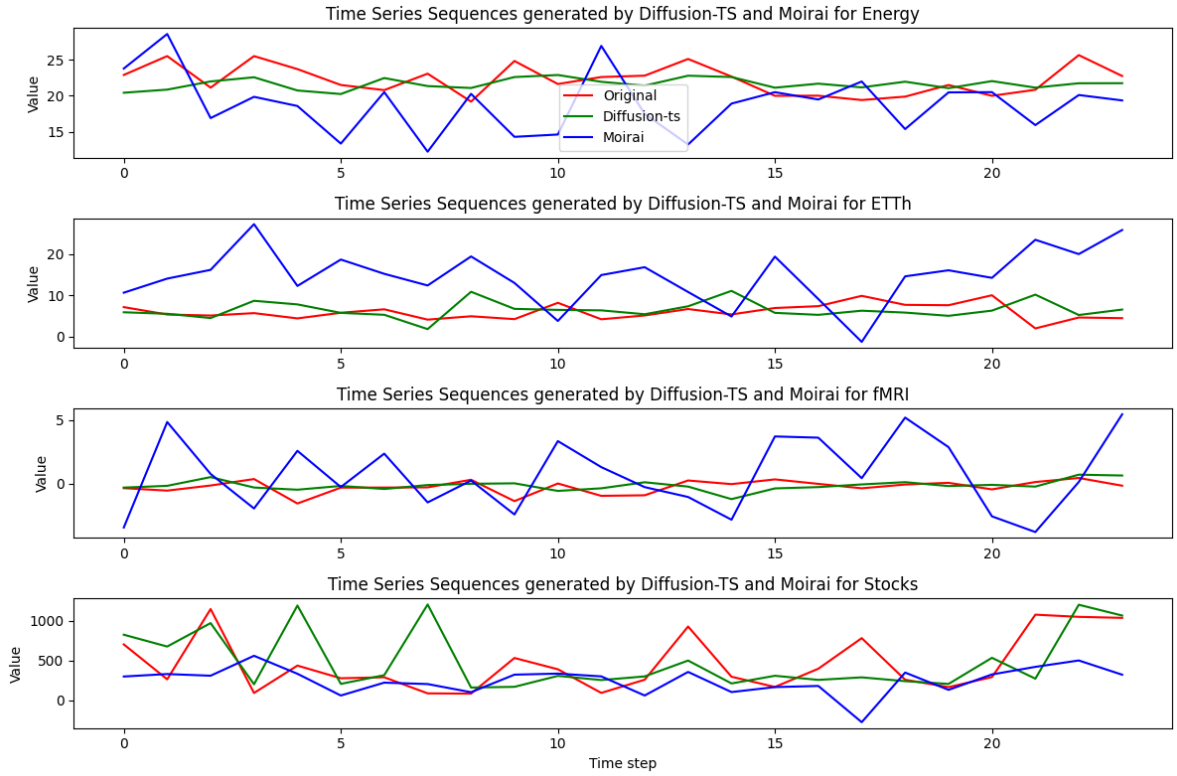


Figure 4.1: Synthetic samples generated by Moirai and Diffusion-TS of each dataset (ETTh, Energy, fMRI, Stocks). The real sequence is displayed alongside the synthetic sequences.

lowering the discriminative score. Thus, the result should be interpreted with caution.

Across all datasets, Diffusion-TS achieves lower Predictive scores than Moirai, indicating better ability to generate useful synthetic data. On the ETTh dataset, for instance, Diffusion-TS reaches an MAE of 0.123, compared to 3.849 for Moirai. The disparity is even more extreme on Stocks, where Moirai yields an MAE over 349,000, compared to just 0.037 for Diffusion-TS. These results suggest that the synthetic sequences produced by Moirai are not just structurally unrealistic, but also unsuitable for training models intended to operate on real-world data. In terms of Context-FID score, Diffusion-TS again outperforms Moirai by several orders of magnitude. Moirai's large Context-FID scores indicate that even when generation is seeded with real context, it fails to continue sequences in a way that statistically resembles the real data distribution. For the Correlational score we observe that Diffusion-TS performs significantly better on all datasets. On high-dimensional datasets like fMRI (49 features) and Energy (27 features), this metric becomes especially important. Diffusion-TS maintains reasonably low correlational errors for these datasets, whereas Moirai's errors are larger. These results suggest that Diffusion-TS is capable of preserving multivariate dependencies across variables. Moirai, by contrast, appears to produce samples where inter-variable structure is not captured at all, especially as dimensionality increases.

When looking at the performance per dataset, we see no single dataset for which Diffusion-TS consistently performs best across all metrics. However, results on ETTh and Stocks are the strongest overall, followed by Energy, with fMRI yielding the worst scores. This pattern correlates strongly with the number of features: ETTh and Stocks are low-dimensional (7 and 6 features, respectively), whereas Energy and fMRI are higher-dimensional (27 and 49). The higher dimensionality of these datasets may pose additional challenges for generating realistic cross correlations between the features, which is reflected in the higher correlational scores. For Moirai, the overall performance is poor across the board, but the ETTh dataset shows slightly more reasonable scores than the others. On Energy and fMRI, the scores degrade significantly, suggesting that Moirai does not scale well with dimensionality in a generative setting. Stocks is a particularly confusing case since the discriminative and correlational scores are not

terrible, but the predictive and context-FID scores are completely unusable. This inconsistency shows that Moirai may generate extreme or inconsistent values in some cases, which corrupt the statistical evaluation. Since this limitation could not be resolved through fine-tuning, we select diffusion models for subsequent experiments.

Experiment Summary: We evaluate transformer-based (Moirai) and diffusion-based (Diffusion-TS) generative models across four time series benchmarks using discriminative, predictive, context-FID, and correlational scores.

Key Takeaway: Diffusion-TS significantly outperforms Moirai across all metrics and datasets, achieving lower scores by orders of magnitude in most cases. Generation quality correlates with dimensionality. Both models perform best on low-dimensional datasets. Due to their superior generation quality, diffusion models make a better candidate for watermarking.

4.5. Conclusion

The superior performance of Diffusion-TS over Moirai demonstrates that high-quality time series generation benefits from training objectives focused on global sequence synthesis. In contrast, objectives based on step-wise prediction for short horizons appear less suited for capturing long-range temporal structure. Given these results, we can conclude that adapting a GPT forecasting model like Moirai for time series generation proved ineffective. We hypothesize that when used for synthesis, errors from each generation step might propagate and amplify, leading to poor long-range coherence. Moreover, Moirai was not trained to match the global or joint distribution of sequences, but only to minimize short-horizon prediction error. These limitations make it unsuitable for time series generation with the current transformer-based models available.

Overall, our results indicate consistent and substantial performance gap between Diffusion-TS and Moirai across the considered metrics. Unlike Moirai, Diffusion-TS is capable of generating realistic time series that support downstream tasks and capture structural properties of the original datasets. These capabilities make it the superior candidate for applications requiring high-fidelity synthetic time series, including the downstream task of watermarking explored in later sections. For this reason, we elect diffusion models as the backbone of our evaluation of time series watermark robustness.

5

Latent Diffusion Models for Time Series Watermarking

Chapter 4 demonstrated that standard diffusion models successfully generate high-quality synthetic time series. This quality is essential for watermarking as embedded watermarks almost inevitably degrade generation quality slightly. Models must produce realistic synthetic data to absorb this cost. Additionally, watermarking serves no practical purpose if synthetic data is easily distinguishable from real data. The strong performance of standard diffusion models establishes them as viable candidates for watermarking.

This raises the question: can Latent Diffusion Models (LDMs) serve as viable candidates for watermarking as well? The additional VAE component introduces potential challenges. The latent space must faithfully represent time series structure. Furthermore, encoding and decoding processes must preserve sufficient information for both quality generation and watermark recovery. For LDMs to serve as viable watermarking candidates, they must satisfy three conditions: invertibility between data and latent spaces, successful watermark embedding during generation, and reliable watermark detectability after synthesis.

In this chapter, we conduct an emperical evaluation of LDMs for the task of synthetic time series generation. Our goal is to critically assess the viability of LDMs in producing high quality, realistic, and watermarkable time series across datasets with varying properties. By combining quantitative evaluations with latent space analysis and dataset-specific observations, we aim to uncover the limitations and strengths of LDMs in this setting. This analysis highlights not only the technical feasibility of watermarking but also the broader challenges LDMs face in representing heterogeneous time series data effectively.

5.1. Decoder Inversion

As discussed in Chapter 3, effective watermark detection requires recovering the initial noise vector in which the watermark was originally embedded during the generation process. This recovery necessitates transforming the generated sample back to its corresponding latent representation. A naive approach would directly apply the the encoder to transform observed samples. By doing so, obtaining $\hat{z} = \mathcal{E}(x)$ where x is the generated output. However, this approach is limited by the reconstruction error inherent in encoder-decoder architectures. Since the encoder is not the exact inverse of the decoder, applying $\mathcal{E}(\mathcal{D}(\hat{z}))$ does not perfectly recover the original latent representation z. The resulting latent representation obtained through direct encoding is therefore expected to deviate from the original latent code used during generation, compromising the watermark detection process. Decoder inversion [26] aims to overcome this through inverting the decoder itself rather than relying on the encoder. Through the formulation of an optimization problem, gradient descent is used to estimate the original

latent. Generally, decoder inversion approaches enable a more faithful latent recovery compared to direct encoding—i.e., enabling watermark detection.

Algorithm 2 Decoder Inversion

Input: Target samples x, pre-trained decoder $\mathcal{D}$, encoder $\mathcal{E}$
Output: Latent representation $\hat{z}$
Require: Learning rate η, number of epochs T

Initialize: $\hat{z} \leftarrow \mathcal{E}(x)$
for *epoch = 1 to* T **do**
  $\hat{x} \leftarrow \mathcal{D}(\hat{z})$
  $\mathcal{L} \leftarrow ||\hat{x} - x||_2^2$
  $\hat{z} \leftarrow \hat{z} - \eta \nabla_{\hat{z}} \mathcal{L}$
end
return $\hat{z}$;

The decoder inversion process is described in Algorithm 2. In each iteration, the gradient is updated in the direction that minimizes the difference between the original samples x and the decoder output $\hat{x}$. Each gradient descent step is computed using the loss $\mathcal{L}$ and learning rate η. This continues until we reach convergence, returning a latent representation $\hat{z}$ similar to the one used during inference.

5.2. Methodology

5.2.1. Synthetic Time Series Evaluation

We evaluate the quality of synthetic time series generated by LDMs using the same experimental setup described in Chapter 4. Specifically, we use the same four time-series benchmarks: ETTh, Stocks, Energy, and fMRI (see Table 4.1). We also use the same quality metrics, which are the Discriminative score, Predictive score, Context-FID score, and Correlational score. The key difference in this experiment is the application of these metrics to latent diffusion models rather than transformers and standard diffusion models. With this experiment, we aim to assess whether the latent space representation in LDMs affects generation quality across diverse time series domains.

5.2.2. Watermark Embedding and Detection

We embed a watermark into the initial Gaussian noise of the diffusion process. Afterwards, this watermarked noise is used to generate synthetic time series data through the sampling procedure of the diffusion model. Following generation, the time series is decoded and given as input to decoder inversion to retrieve its final latent representation $\hat{z}$. For decoder inversion, the latent representation $\hat{z}$ is optimized using the Adam optimizer with an initial learning rate of 1×10^{-1}. To adapt the learning rate during training, we employ a ReduceLROnPlateau scheduler that reduces the learning rate by a factor of 0.25 when the loss plateaus for 20 consecutive epochs.

After generation and decoder inversion, we perform inverse sampling to recover the initial noise and reconstruct the watermark. The detection process proceeds by computing a metric score measuring the similarity between the reconstructed and ground truth watermarks. Using these scores, we conduct a hypothesis test to compute the Z-score. A Z-score above 1.64 indicates that the watermark is present.

5.2.3. Watermark Robustness Against Post-Processing

Algorithm 3 Compute Z-score of Watermark Post-Attack

Input: Watermarked attacked noise $\hat{N}_{wa}$, Unwatermarked noise $\hat{N}_u$
Output: Z-score z-score

Compute metric scores: $s_w \leftarrow$ Metric($\hat{N}_{wa}$), $s_u \leftarrow$ Metric($\hat{N}_u$)
Compute mean μ_u and std σ_u of s_u
Compute Z-score: z-score $\leftarrow \frac{s_w - \mu_u}{\sigma_u}$
return z-score

A similar procedure to the complete generation pipeline was used in the robustness evaluation. After decoding, we obtained a synthetic time series (as seen in Figure 5.6). We applied a post-editing attack to this time series at one of two intensities (0.05 or 0.3). The attacked time series was again brought back into its latent space via decoder inversion, followed by inverse sampling to estimate the initial latent noise vector. Finally, we computed the Z-score to assess whether the watermark remains detectable after the attack. Algorithm 3 shares the Z-score computation procedure. We evaluated three types of simple post-editing attacks, a more in depth description of these attacks can be found in Chapter 6:

- **Offset attack**: This attack adds a small shift to the entire time series by computing the mean across the sequence and applying a scaled version of this mean as an additive offset to all values.
- **Crop attack**: This attack crops a random subregion of the time series.
- **Min-Max insert attack**: This attack randomly replaces a subset of time points with values sampled uniformly between the minimum and maximum of each feature.

5.3. Results

5.3.1. Synthetic Data Quality

The results of our synthetic data quality evaluation are presented in Table 5.1.

For the unwatermarked synthetic data, the model produced the highest quality results on the ETTh and Stocks datasets. Both of these datasets have relatively few features and contain no categorical variables, which appears to facilitate the generation of high-fidelity synthetic data. In contrast, the quality of synthetic data for the Energy and Pollution datasets was considerably lower across most metrics. Notably, the synthetic Energy data did achieve a reasonably strong context-FID score. However, it performed poorly on the correlational score, implying the model had difficulty capturing inter-feature relationships, likely due to the dataset's high number of features. Conversely, the synthetic Pollution data achieved a good correlational score but struggled significantly with local context, as reflected in its poor context-FID score.

With the inclusion of latent-adapted watermarks, we observe that generally TimeWak does not impose significant degradation across all datasets. In contrast to TreeRing and Gaussian Shading, Timewak does not negatively impact the Correlational Score on Energy and fMRI. Importantly, TimeWak never caused a significant degradation in any of the four metrics, making it the most consistent watermark in terms of preserving the fidelity of the synthetic data. TreeRing demonstrates strong performance on the Discriminative and Predictive scores, typically inducing only minimal degradation. Occasionally, it even manages to preserve these metrics at levels comparable to unwatermarked data. However, the watermark exhibits notably weaker performance on Context-FID and Correlational scores. With the exception of ETTh, TreeRing consistently reduces performance on these metrics across all other datasets. The most pronounced drops occur on the high-dimensional Energy and fMRI datasets. Gaussian Shading, while excelling on specific metrics for certain datasets, such as achieving strong context-FID and correlational scores on the Stocks dataset, tended to degrade overall quality the most. This was especially evident in the discriminative score, where it consistently produced the worst scores across all datasets,

Table 5.1: Comparison between TimeAutoDiff and Diffusion-TS on synthetic time series generation with and without watermark. Scores are the mean values over ten seeds (standard deviation in brackets). TR = TreeRing, GS = Gaussian Shading, TW = TimeWak. For every metric, lower is better. For TimeAutoDiff, the best results are shown in bold.

Dataset	Model	WM	Discriminative	Predictive	Context-FID	Correlational
ETTh	TimeAutoDiff	None	0.014 ± (0.011)	0.083 ± (0.001)	0.016 ± (0.002)	0.086 ± (0.022)
		TR	**0.016** ± (0.012)	0.083 ± (0.002)	0.017 ± (0.002)	0.098 ± (0.035)
		GS	0.035 ± (0.016)	0.083 ± (0.002)	0.041 ± (0.004)	**0.084** ± (0.025)
		TW	0.017 ± (0.015)	**0.081** ± (0.001)	**0.016** ± (0.001)	0.090 ± (0.027)
	Diffusion-TS	None	0.096 ± (0.007)	0.121 ± (0.003)	0.209 ± (0.015)	0.082 ± (0.020)
		TR	0.276 ± (0.013)	0.136 ± (0.002)	1.651 ± (0.068)	0.201 ± (0.028)
		GS	0.388 ± (0.012)	0.176 ± (0.005)	4.216 ± (0.404)	0.416 ± (0.029)
		TW	0.092 ± (0.014)	0.124 ± (0.003)	0.207 ± (0.011)	0.197 ± (0.036)
Energy	TimeAutoDiff	None	0.388 ± (0.011)	0.240 ± (0.001)	0.181 ± (0.008)	2.065 ± (0.337)
		TR	0.398 ± (0.007)	**0.239** ± (0.002)	0.664 ± (0.105)	2.852 ± (0.477)
		GS	0.407 ± (0.011)	0.244 ± (0.002)	0.283 ± (0.023)	2.508 ± (0.305)
		TW	**0.387** ± (0.009)	**0.239** ± (0.001)	**0.170** ± (0.012)	**1.963** ± (0.340)
	Diffusion-TS	None	0.144 ± (0.010)	0.253 ± (0.000)	0.117 ± (0.014)	1.597 ± (0.125)
		TR	0.415 ± (0.010)	0.292 ± (0.002)	0.379 ± (0.033)	2.551 ± (0.319)
		GS	0.494 ± (0.002)	0.330 ± (0.002)	1.459 ± (0.164)	3.503 ± (0.409)
		TW	0.142 ± (0.013)	0.253 ± (0.000)	0.118 ± (0.006)	2.171 ± (0.398)
Stocks	TimeAutoDiff	None	0.095 ± (0.008)	0.063 ± (0.000)	0.117 ± (0.025)	0.044 ± (0.022)
		TR	0.161 ± (0.017)	0.065 ± (0.000)	0.475 ± (0.061)	0.052 ± (0.022)
		GS	0.196 ± (0.067)	**0.062** ± (0.000)	**0.046** ± (0.002)	**0.011** ± (0.010)
		TW	**0.081** ± (0.017)	0.063 ± (0.001)	0.110 ± (0.022)	0.045 ± (0.017)
	Diffusion-TS	None	0.155 ± (0.016)	0.037 ± (0.000)	0.338 ± (0.042)	0.029 ± (0.017)
		TR	0.217 ± (0.044)	0.039 ± (0.000)	1.119 ± (0.088)	0.098 ± (0.025)
		GS	0.425 ± (0.014)	0.042 ± (0.002)	9.673 ± (1.101)	0.109 ± (0.019)
		TW	0.155 ± (0.016)	0.037 ± (0.000)	0.288 ± (0.039)	0.027 ± (0.010)
fMRI	TimeAutoDiff	None	0.493 ± (0.005)	0.102 ± (0.000)	4.727 ± (0.394)	8.075 ± (0.178)
		TR	0.496 ± (0.002)	0.104 ± (0.000)	5.115 ± (0.465)	11.501 ± (0.304)
		GS	0.497 ± (0.001)	0.108 ± (0.001)	**4.419** ± (0.429)	10.179 ± (0.316)
		TW	**0.494** ± (0.002)	**0.102** ± (0.000)	4.734 ± (0.426)	**7.958** ± (0.106)
	Diffusion-TS	None	0.110 ± (0.015)	0.100 ± (0.000)	0.191 ± (0.006)	2.615 ± (0.076)
		TR	0.479 ± (0.055)	0.148 ± (0.002)	2.384 ± (0.145)	13.540 ± (0.183)
		GS	0.500 ± (0.000)	0.109 ± (0.001)	0.679 ± (0.035)	14.821 ± (0.094)
		TW	0.116 ± (0.012)	0.100 ± (0.000)	0.186 ± (0.017)	2.576 ± (0.060)

indicating that synthetic samples generated with this watermark were the easiest to distinguish from real data.

Experiment Summary: We evaluate LDM-generated synthetic time series quality across four datasets using discriminative, predictive, context-FID, and correlational scores. We embed three watermarks (Gaussian Shading, TreeRing, TimeWak) and assess their impact on generation quality.

Key Takeaway: Quality correlates strongly with dataset composition. Low-dimensional, numerical-only datasets achieve the best scores. Feature heterogeneity proves challenging to LDMs. TimeWak preserves quality most consistently across all metrics and datasets. TreeRing maintains discriminative and predictive scores but degrades context-FID and correlational scores, especially on high-dimensional datasets. Gaussian Shading causes the most quality degradation.

5.3.2. Watermark Detectability

Table 5.2: Z-scores for watermarks embedded during sampling after attacks. The intensity of each attack (0.05 and 0.3) appears under the attack name. Scores are the mean values with standard deviations in subscript parentheses. The highest Z-score is shown in bold.

Dataset	Watermark	Unattacked	Offset		Crop		Min-Max Insert	
			0.05	0.3	0.05	0.3	0.05	0.3
Energy	Gaussian Shading	**61.52**$_{\pm (1.01)}$	**37.46**$_{\pm (0.88)}$	**5.36**$_{\pm (0.69)}$	**10.58**$_{\pm (0.91)}$	**11.69**$_{\pm (0.60)}$	**42.03**$_{\pm (0.88)}$	**10.81**$_{\pm (0.71)}$
	TreeRing	0.62$_{\pm (0.03)}$	1.01$_{\pm (0.02)}$	1.64$_{\pm (0.03)}$	0.02$_{\pm (0.02)}$	0.79$_{\pm (0.02)}$	-0.67$_{\pm (0.02)}$	-1.58$_{\pm (0.02)}$
	TimeWak	4.66$_{\pm (0.67)}$	2.28$_{\pm (0.65)}$	0.32$_{\pm (0.59)}$	1.03$_{\pm (0.61)}$	-0.91$_{\pm (0.71)}$	1.78$_{\pm (0.71)}$	-1.32$_{\pm (0.71)}$
ETTh	Gaussian Shading	**276.41**$_{\pm (0.60)}$	**71.43**$_{\pm (0.72)}$	**32.90**$_{\pm (0.57)}$	**20.13**$_{\pm (0.66)}$	-17.20$_{\pm (0.53)}$	**142.22**$_{\pm (0.89)}$	**26.43**$_{\pm (0.69)}$
	TreeRing	0.60$_{\pm (0.02)}$	-5.45$_{\pm (0.02)}$	-6.52$_{\pm (0.03)}$	-8.09$_{\pm (0.03)}$	-7.46$_{\pm (0.03)}$	-2.93$_{\pm (0.03)}$	-8.66$_{\pm (0.02)}$
	TimeWak	109.14$_{\pm (0.67)}$	4.94$_{\pm (0.62)}$	-0.01$_{\pm (0.60)}$	2.45$_{\pm (0.68)}$	**5.04**$_{\pm (0.65)}$	29.03$_{\pm (0.79)}$	0.91$_{\pm (0.67)}$
fMRI	Gaussian Shading	**103.84**$_{\pm (0.79)}$	**98.45**$_{\pm (0.66)}$	**61.59**$_{\pm (0.64)}$	**58.92**$_{\pm (0.65)}$	**33.05**$_{\pm (0.61)}$	**89.02**$_{\pm (0.70)}$	**51.77**$_{\pm (0.64)}$
	TreeRing	0.57$_{\pm (0.02)}$	0.36$_{\pm (0.02)}$	-0.19$_{\pm (0.02)}$	-0.88$_{\pm (0.02)}$	-1.32$_{\pm (0.02)}$	0.14$_{\pm (0.02)}$	-0.84$_{\pm (0.02)}$
	TimeWak	6.40$_{\pm (0.69)}$	4.39$_{\pm (0.72)}$	0.25$_{\pm (0.69)}$	2.57$_{\pm (0.60)}$	10.23$_{\pm (0.71)}$	4.82$_{\pm (0.68)}$	0.50$_{\pm (0.63)}$
Stocks	Gaussian Shading	**103.17**$_{\pm (0.63)}$	**87.08**$_{\pm (0.72)}$	**22.41**$_{\pm (0.91)}$	**27.20**$_{\pm (1.09)}$	**5.49**$_{\pm (0.52)}$	**90.91**$_{\pm (0.69)}$	**49.46**$_{\pm (0.60)}$
	TreeRing	1.08$_{\pm (0.02)}$	0.89$_{\pm (0.02)}$	-0.69$_{\pm (0.04)}$	-1.32$_{\pm (0.03)}$	-3.14$_{\pm (0.03)}$	0.93$_{\pm (0.02)}$	0.15$_{\pm (0.03)}$
	TimeWak	18.82$_{\pm (0.69)}$	15.99$_{\pm (0.71)}$	8.83$_{\pm (0.81)}$	3.77$_{\pm (0.82)}$	2.93$_{\pm (0.67)}$	15.04$_{\pm (0.67)}$	2.67$_{\pm (0.74)}$

Having established the LDM's ability to generate realistic time series, we next examine watermark detectability. Here, we focus on watermarks embedded at the start of inference. The results of this experiment are summarized in column 'Unattacked' of Table 5.2.

Across all datasets, Gaussian Shading consistently achieves the highest Z-scores, surpassing the other methods. Its lowest score appears on the synthetic Energy dataset, which contains heterogeneous features. This suggests that feature heterogeneity may slightly reduce detectability, although the Z-score remains well above the detection threshold. TimeWak is also detectable across all datasets. Although its scores are lower than those of Gaussian Shading, they still exceed 1.64, ensuring reliable detection. TimeWak achieves its lowest scores on Energy and fMRI, which both contain a high number of features. Their high dimensionality might disperse TimeWak's signal too much, leading to lower Z-scores in comparison to those on ETTh and Stocks. Lastly, TreeRing consistently performs poorly across all datasets, with all Z-scores below the 1.64 threshold. Even in absence of post-processing attacks, TreeRing fails to be detectable. Unlike the other watermarks, its performance does not appear to correlate with the presence of heterogeneity or number of features. TreeRing's poor performance suggests that Fourier-based watermarking might not be compatible with the structure of time series data.

Experiment Summary: We assess watermark detectability after generation by computing Z-scores on unwatermarked and watermarked synthetic time series. A Z-score above 1.64 indicates reliable detection.

Key Takeaway: Gaussian Shading achieves the highest detectability across all datasets, maintaining Z-scores well above the threshold. TimeWak remains detectable on all datasets but shows reduced performance on high-dimensional data. TreeRing fails to achieve detectability on any dataset, with all Z-scores below 1.64.

5.3.3. Watermark Robustness Against Post-Processing

Finally, we examine the robustness of each watermark against post-processing operations. Specifically, we evaluate whether the watermarks remain detectable after the synthetic time series undergoes common data augmentations. The results of this evaluation can be found in Table 5.2. In terms of robustness, Gaussian Shading emerged as the most robust watermark across all datasets and attack types. It remained detectable even under strong attack intensities, with the sole exception being a high-intensity cropping attack on the ETTh dataset. This suggests that the Gaussian Shading watermark

is well-preserved throughout the decoding and inversion process, even when the time series is perturbed. TimeWak was the second most robust watermark but exhibited significantly more sensitivity to perturbations than Gaussian Shading. Its detectability frequently dropped below the threshold of 1.64 under moderate or strong attack intensities, particularly for the cropping and min-max insertion attacks. Finally, TreeRing exhibited the lowest robustness of all the watermarks. However, since it was already undetectable in the unattacked scenario, further decreases in Z-score due to post-editing attacks have no meaningful practical impact.

Experiment Summary: We evaluate watermark robustness against three post-processing attacks (offset, crop, min-max insertion) at two intensity levels (0.05 and 0.3). Z-scores are computed after applying attacks to assess whether watermarks remain detectable. Watermarks are reliably detectable when their Z-score is at least 1.64.

Key Takeaway: Gaussian Shading demonstrates superior robustness across all datasets and attack types, maintaining detectability even under strong attacks with one exception (high-intensity crop on ETTh). TimeWak shows moderate robustness but frequently falls below the 1.64 threshold, particularly for cropping and min-max insertion. TreeRing exhibits the lowest robustness, but this is inconsequential since it was already undetectable before attacks.

5.4. Latent Space Analysis

The results of our synthetic data quality evaluation in Section 5.3.1 show differences in the quality of synthetic datasets generated by the LDM. High-dimensional datasets and datasets with heterogeneous features appear to pose challenges in particular. Compared to the data diffusion model, the LDM yields lower metric scores on synthetic data with these characteristics. In order to gain insight into the cause of these differences, we first perform an ablation on the latent representation created by the VAE. The data diffusion model does not employ a VAE, unlike the LDM. We therefore aim to assess whether the quality of synthetic time series generated by the LDM depends on the latent space produced by the VAE encoder. Since the VAE is trained with a Gaussian prior, it is expected to map input time series to a latent space that closely approximates a standard normal distribution. If the latent space fails to approximate the Gaussian prior, it could indicate that the VAE failed to capture underlying patterns in the data. If this flawed representation of the original data is given to the diffusion model as input, it could in turn affect the output quality of the diffusion model, resulting in low quality synthetic data. Beyond generation quality, the structure of the latent space has direct implications for watermark detectability. Effective watermark retrieval requires precise reconstruction of the latent representation. That is, $\hat{z} \approx z$. If the VAE cannot maintain a coherent latent space throughout encoding, decoding, and decoder inversion, then adapting existing watermarking methods to LDMs becomes infeasible.

5.4.1. Ablation

To evaluate how the structure of the latent space changes throughout the diffusion and decoding process, we analyze the distributions of latent representations at three checkpoints within the model pipeline. First, after completion of the VAE training procedure. Second, after re-encoding the generated data. Third, after applying decoder inversion to the generated data. Algorithm 4 provides a high-level overview of the procedure.

Algorithm 4 Latent Space Distribution Analysis

Input: Original data x, synthetic data $\hat{x}$
Output: JS and Wasserstein distances for all checkpoints
Require: Encoder $\mathcal{E}$, decoder inversion $\mathcal{DI}$
`// Obtain latent representations`
$z_{\text{oracle}} \leftarrow \mathcal{E}(x)$
$\hat{z}_{\text{re-enc}} \leftarrow \mathcal{E}(\hat{x})$
$\hat{z}_{\text{DI}} \leftarrow \mathcal{DI}(\hat{x})$
Compute empirical distributions $P_{\text{oracle}}, P_{\text{re-enc}}, P_{\text{DI}}$
Compute Jensen-Shannon distances $d_{JS}(P_{\text{oracle}}, \mathcal{N}(0,I))$, $d_{JS}(P_{\text{re-enc}}, \mathcal{N}(0,I))$, $d_{JS}(P_{\text{DI}}, \mathcal{N}(0,I))$
Compute Wasserstein distances $d_W(P_{\text{oracle}}, \mathcal{N}(0,I))$, $d_W(P_{\text{re-enc}}, \mathcal{N}(0,I))$, $d_W(P_{\text{DI}}, \mathcal{N}(0,I))$

The first checkpoint examines the oracle distribution, obtained immediately after encoding real time series data using the VAE encoder. This representation, z, is passed directly to the diffusion model and serves as the cleanest view of how well the VAE aligns its latent space with the standard normal distribution. Since the VAE is explicitly trained to enforce a Gaussian prior on this space, the oracle distribution provides a baseline for evaluating encoder effectiveness.

The second checkpoint involves decoding followed by re-encoding. After completing the diffusion sampling, we decode the resulting latents into a synthetic time series. Afterwards, we re-encode this time series using the VAE encoder, obtaining $\hat{z}_{re-enc}$. Thereby giving insight into the error introduced by the decoder, and the encoder's ability to recover a latent representation after generation. This stage is particularly informative because it provides a baseline for understanding the fidelity of re-encoding relative to the original latent distribution, independent of decoder inversion. If the decoder introduces minimal distortion and the encoder is robust to such distortions, then the resulting distribution should still approximate a Gaussian reasonably well.

The third checkpoint applies decoder inversion to the decoded time series. By doing so we recover a latent representation, $\hat{z}_{DI}$, that closely corresponds to the one at the end of the diffusion process. In this case, we do not rely on the encoder at all. By analyzing the latent distribution recovered through decoder inversion, we can evaluate how well this process preserves or distorts the structure of the latent space. Additionally, this checkpoint provides evidence to determine whether the post-DI distribution diverges from both the oracle and re-encoded distributions.

We compute quantitative divergence metrics to assess the degree of alignment with the standard normal distribution at each checkpoint. Specifically, we compute both the Wasserstein distance and the Jensen-Shannon (JS) distance between the observed latent distribution and a standard Gaussian. Together, these metrics provide a comprehensive view of how the latent space evolves and where distortions arise. We revisit the datasets from Chapter 4 and include the Pollution dataset, all summarized in Table 4.1. We augment our quantitative evaluation with visualizations of the latent distribution at each of the three checkpoints.

5.4.2. Results

The Energy dataset (Figure 5.3) exhibits the most divergent latent distribution, showing the greatest deviation from the standard Gaussian prior. The distribution remains largely unchanged even after re-encoding and decoder inversion, with stable Wasserstein and JS distance values (Table 5.4). The relatively low Discriminative score and Correlational score (see Table 5.1) we observe for Energy could be related to its latent space diverging significantly from the standard Gaussian distribution.

In contrast, the Pollution dataset (see the top image of Figure 5.5) presents a more complex case. Like Energy and fMRI, Pollution yields relatively low metric scores (Table 5.3). In the oracle scenario, its latent distribution somewhat approximates a standard Gaussian, but this structure is significantly disrupted after decoding and returning to the latent space. We summarize this in Table 5.4, with both the re-encoded and decoder inversion JS distances diverging significantly from the oracle distance. Interestingly, its Wasserstein distance sees little change between checkpoints To investigate the source

Table 5.3: Results of synthetic time series generated by TimeAutoDiff with and without watermark on Pollution datasets. Pollution (num) denotes the Pollution dataset with only numerical features. Scores are the mean values over ten seeds (standard deviation in brackets). TR = TreeRing, GS = Gaussian Shading, TW = TimeWak. For every metric, lower is better. The best scores are shown in bold.

Dataset	WM	Discriminative	Predictive	Context-FID	Correlational
Pollution	None	0.428 (0.010)	0.008 (0.000)	1.987 (0.093)	0.204 (0.021)
	TR	0.418 (0.022)	0.009 (0.000)	**1.935** (0.127)	0.203 (0.028)
	GS	0.428 (0.021)	**0.008** (0.000)	2.261 (0.146)	**0.201** (0.026)
	TW	**0.415** (0.024)	0.009 (0.000)	1.978 (0.101)	0.202 (0.018)
Pollution (num)	None	0.011 (0.006)	0.009 (0.000)	0.009 (0.001)	0.047 (0.016)
	TR	0.017 (0.010)	**0.009** (0.000)	**0.011** (0.001)	**0.049** (0.018)
	GS	0.024 (0.012)	**0.009** (0.000)	0.068 (0.004)	0.050 (0.012)
	TW	**0.012** (0.013)	**0.009** (0.000)	0.018 (0.001)	0.052 (0.012)

Table 5.4: Distance between latent distribution and Gaussian distribution for every dataset at different checkpoints in synthesis pipeline.

Metric	Checkpoint	ETTh	Stocks	Energy	fMRI	Pollution	Pollution (num)
Jensen-Shannon Distance (lower is better)	Oracle	0.138	0.071	0.363	0.113	0.229	0.108
	Re-encoded	0.214	0.071	0.331	0.095	0.427	0.281
	Decoder Inversion	0.165	0.099	0.325	0.109	0.554	0.227
Wasserstein Distance (lower is better)	Oracle	0.077	0.068	0.943	0.124	0.622	0.216
	Re-encoded	0.074	0.065	0.943	0.123	0.620	0.221
	Decoder Inversion	0.076	0.066	0.942	0.123	0.619	0.222

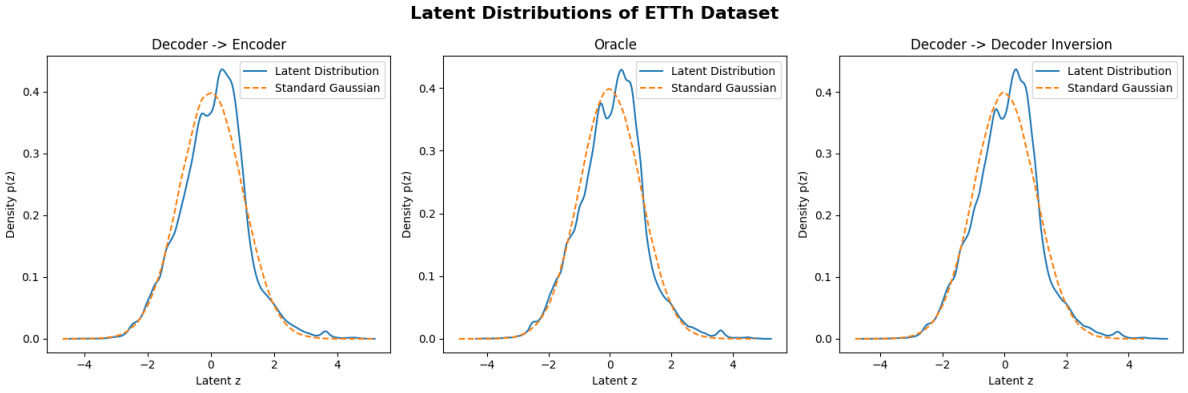


Figure 5.1: Distribution of the latent spaces of the ETTh dataset.

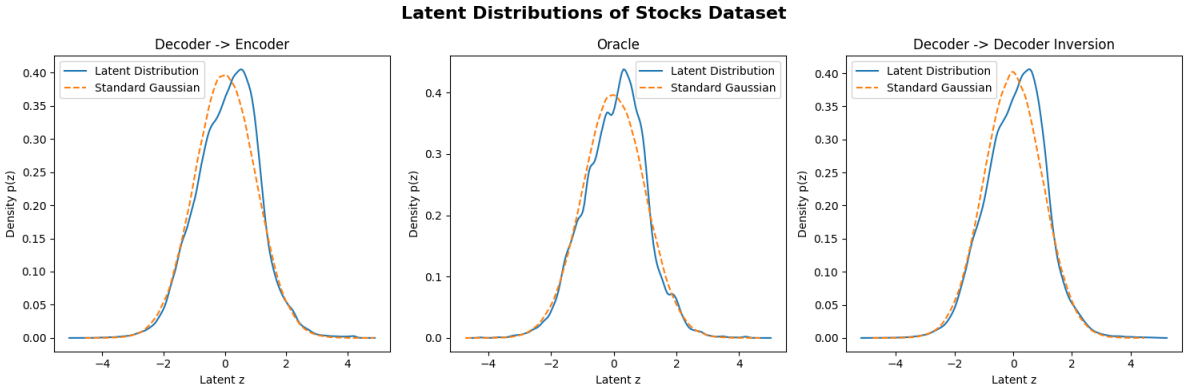


Figure 5.2: Distribution of the latent spaces of the Stocks dataset.

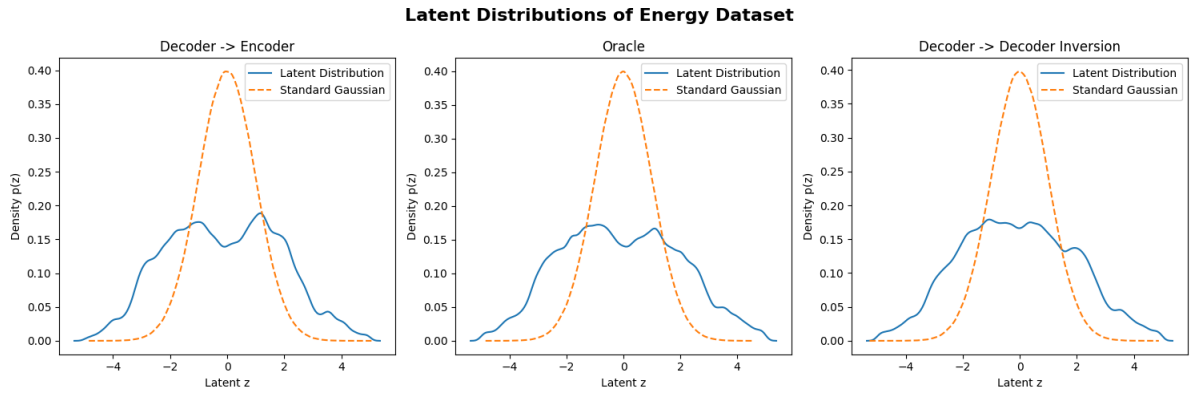


Figure 5.3: Distribution of the latent spaces of the Energy dataset.

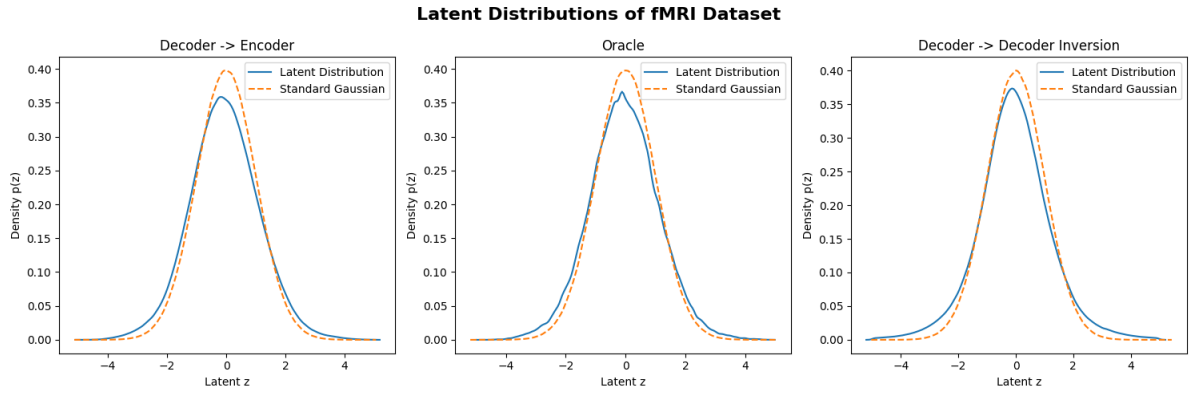


Figure 5.4: Distribution of the latent spaces of the fMRI dataset.

of this degradation, we trained a version of the model on a modified Pollution dataset that includes only numerical features. As can be observed in the bottom image of Figure 5.5, the resulting latent distributions appear considerably more Gaussian after decoding. The JS distance and Wasserstein distance of numerical-only Pollution support this observation, with both being lower than those of the original dataset (see Table 5.4). Removing the categorical features improved the synthetic data quality markedly across all metrics, matching the scores achieved on ETTh and Stocks (Table 5.3).

For the ETTh and Stocks datasets, the latent representations (Figures 5.1 and 5.2 respectively) closely approximate a standard Gaussian distribution across all three checkpoints. Table 5.4 supports this observation, with both datasets exhibiting relatively low JS distance values. In terms of Wasserstein distance, they are the best performing datasets. Importantly, even after a full pass through the diffusion process, decoding, and decoder inversion, the latent distributions remain reasonably Gaussian. The fMRI dataset (Figure 5.4) also demonstrates approximate Gaussian latent representations across every checkpoint. Its divergence metrics are comparable to ETTh and Stocks (Table 5.4). Similar to those datasets, fMRI consists of only numerical features. Despite its high dimensionality, fMRI maintains a well-structured latent space, suggesting that dimensionality alone does not determine latent distribution quality.

Overall, categorical features appear to hinder the VAE's ability to impose a Gaussian prior on the latent space. Latent spaces that deviate from Gaussian distributions generally correlate with poor synthetic data quality. However, Gaussian alignment alone does not guarantee high-quality generation. The fMRI dataset demonstrates this limitation. Despite maintaining approximately Gaussian latent distributions across all checkpoints, it achieves poor scores on every quality metric. This suggests that LDMs struggle with high-dimensional data even when latent representations resemble the prior distribution. Thus, poor latent representation typically leads to poor generation performance. However, well-structured latent spaces are not sufficient for high-quality synthetic data.

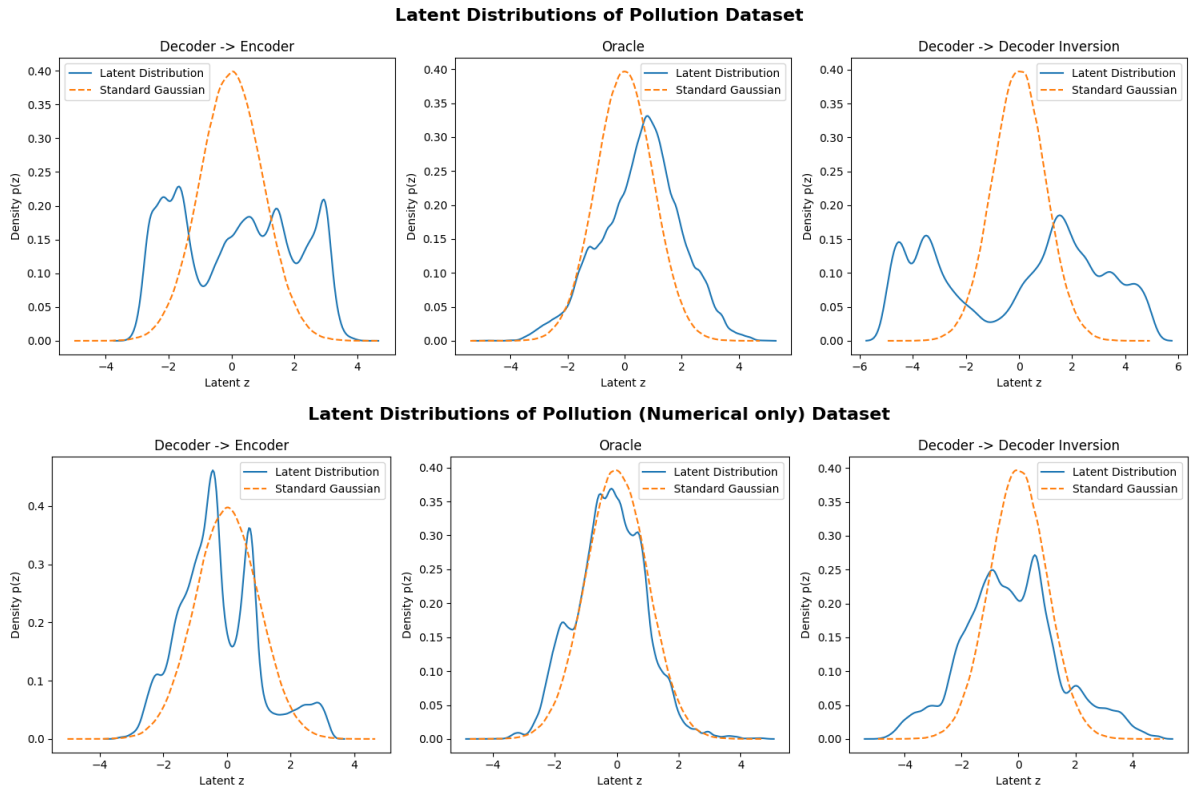


Figure 5.5: Distribution of the latent spaces of the standard Pollution dataset (top image) and of the edited Pollution dataset with only numerical features (bottom image).

Experiment Summary: We analyze latent distributions at three checkpoints: oracle (directly after encoding), re-encoded (after decoding and re-encoding), and decoder inversion (after decoding and inversion). JS distance and Wasserstein distance measure alignment with standard Gaussian distributions across six datasets.

Key Takeaway: Dataset composition strongly determines latent space quality. Numerical-only datasets maintain Gaussian-aligned latent spaces throughout the pipeline. Datasets with categorical features exhibit severe distributional divergence. Removing categorical features frrom heterogeneous datasets substantially improves Gaussian alignment. However, Gaussian alignment alone is insufficient: fMRI maintains Gaussian latents but produces poor quality synthetic data.

5.5. Decoder Inversion Analysis

The previous section demonstrated that latent space structure varies significantly across datasets. Some maintain Gaussian distributions throughout the pipeline, while others diverge from this prior. Watermark detection in latent diffusion models requires recovering the latent representation from generated synthetic time series. Decoder inversion allows us to recover this latent representation. We now investigate how well decoder inversion manages to recover the latent and whether watermarks remain detectable after the decoder inversion process.

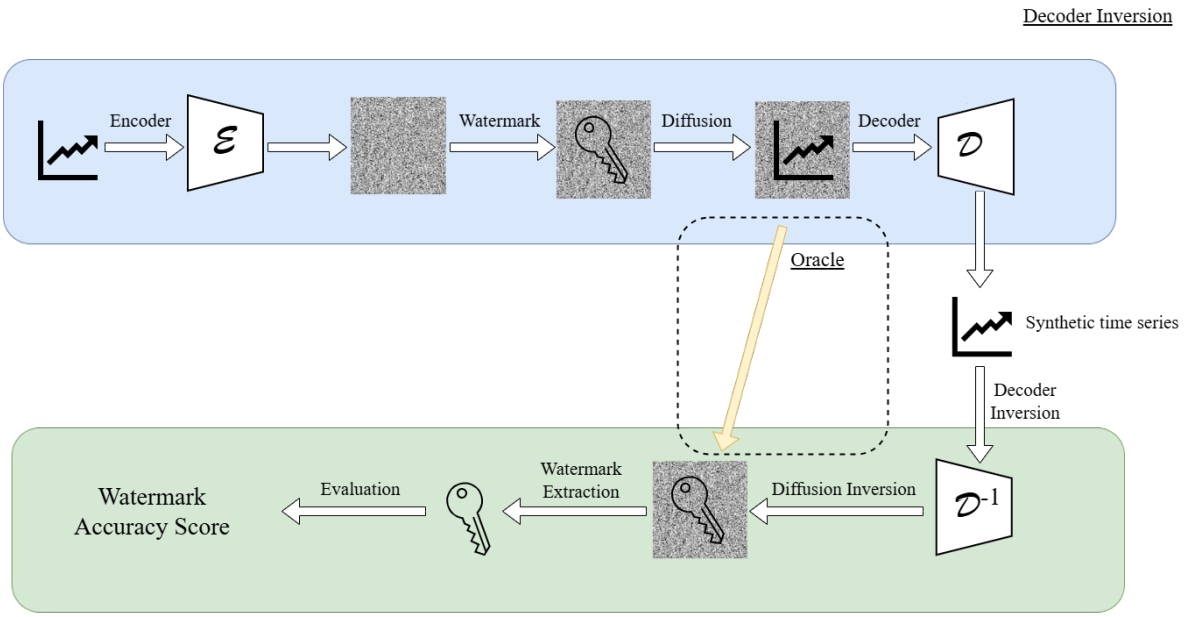


Figure 5.6: The complete watermarking pipeline from embedding to detection for the Oracle scenario and for Decoder Inversion. The Oracle, indicated by the yellow arrow, preserves the original latent representation. In contrast, Decoder Inversion approximates the original latent representation.

5.5.1. Ablation

We evaluated the detectability of each watermark by computing Z-scores under two conditions. Figure 5.6 visualizes the detection pipeline of each condition. The first is an oracle scenario in which diffusion inversion was performed directly on the latent representation at the end of the diffusion process, without decoding. This represents an ideal upper bound on detectability, as it eliminates any information loss from the decoder or inversion process. The second condition is the complete pipeline, which involves decoder inversion. The generated latent is decoded into a synthetic time series, followed by decoder inversion (as described in Algorithm 2) to recover an approximate latent representation. Afterwards, diffusion inversion is performed on this recovered latent to attain the initial watermarked noise. Finally, we extract the watermark through reverse sampling and compute a bit accuracy score. This pipeline mimics the practical watermark detection workflow where only the synthetic time series is available, not the original latent representation.

Using both procedures allows us to gain insight into how much information is lost during the decoding and decoder inversion process. By comparing the Z-scores obtained in the oracle case with those from the full pipeline, we can assess how reliably the watermark survives the generative process. A large gap between the two Z-scores suggests that decoding and inversion introduce significant distortion that hinders watermark recovery. This comparison thus provides valuable insight into both the robustness of the watermark and the fidelity of the model's latent representations across different stages of the generation process.

For both conditions, we compute Z-scores across 1000 samples as a measure of watermark detectability. A Z-score above 1.64 indicates statistical significance, meaning the watermark is reliably detectable. We evaluate the same watermarking methods as in Section 5.3.1: Gaussian Shading, TreeRing, and TimeWak. We also use the same datasets as in 5.4, the properties of which can be found in Table 4.1.

Table 5.5: Z-scores of each watermark after decoder inversion, and in the Oracle scenario. Scores are the mean Z-values over 1000 rows (with standard deviations). Pollution (Num) represents an abliation on the Polllution dataset with categorical columns removed.

Dataset	Watermark	Oracle	Post-Decoder Inversion
ETTh	Gaussian Shading	345.83 (0.42)	276.41 (0.60)
	TreeRing	2.25 (0.02)	0.60 (0.02)
	TimeWak	192.92 (0.55)	109.14 (0.67)
Energy	Gaussian Shading	775.11 (0.21)	61.52 (1.01)
	TreeRing	5.22 (0.02)	0.62 (0.03)
	TimeWak	413.29 (0.56)	4.66 (0.67)
Stocks	Gaussian Shading	323.46 (0.34)	103.17 (0.63)
	TreeRing	6.11 (0.02)	1.08 (0.02)
	TimeWak	184.55 (0.54)	18.82 (0.69)
fMRI	Gaussian Shading	1082.61 (0.15)	103.84 (0.79)
	TreeRing	6.75 (0.03)	0.57 (0.02)
	TimeWak	535.38 (0.51)	6.40 (0.69)
Pollution	Gaussian Shading	393.44 (0.31)	0.18 (0.73)
	TreeRing	5.73 (0.02)	-0.01 (0.02)
	TimeWak	211.57 (0.48)	0.42 (0.67)
Pollution (Num)	Gaussian Shading	325.13 (0.42)	3.02 (0.61)
	TreeRing	5.46 (0.02)	-0.04 (0.02)
	TimeWak	183.44 (0.56)	-0.30 (0.76)

5.5.2. Results

First, all watermarks exhibit significant drops in Z-score after decoder inversion, compared to the oracle case. This suggests that decoder inversion introduces substantial information loss or distortion in the latent space, which in turn weakens the ability to accurately recover the embedded watermark. Additionally, the Z-score drop appears to correlate with the alignment between the latent distribution and a standard Gaussian. Datasets such as ETTh and Stocks, which have latent spaces more closely resembling a standard Gaussian after decoder inversion, show relatively smaller reductions in Z-score. In contrast, Energy and Pollution exhibit severe drops, likely due to their post-inversion latent distributions diverging more significantly from a standard Gaussian prior. However, Gaussian alignment alone does not fully account for Z-score degradation. The fMRI dataset maintains an approximately Gaussian latent space after decoder inversion. Yet, it experiences substantial Z-score reductions comparable to datasets with poorly structured latent spaces. These results suggest that the effectiveness of watermark recovery is, at least in part, dependent on the degree to which the latent space retains Gaussian structure after decoding. Dimensionality appears to be an additional factor of influence, with high dimensional data presenting additional challenges for accurate reconstruction, regardless of latent distribution.

Looking solely at the absolute Z-scores after decoder inversion, it is notable that some watermarks in the synthetic Energy dataset still achieve relatively high detectability, despite the latent space diverging from a Gaussian prior. In fact, synthetic Energy data yields higher Z-scores than synthetic Pollution data even though the latent space of the latter matches the Gaussian distribution more closely. This suggests that the divergence between oracle and post decoder inversion distributions may be more critical than alignment with the Gaussian distribution. Table 5.4 illustrates this pattern through JS distances. Energy's oracle JS distance (0.363) decreases slightly to 0.325 post-DI. In contrast, Pollution's distance jumps from 0.229 to 0.554. This represents the largest distributional shift across all datasets. Notably, Pollution is also the only dataset where no watermarks remain detectable after decoder inversion. The numerical-only Pollution variant provides supporting evidence. Removing categorical features reduces the difference between the oracle JS distance and post-DI JS distance. This version achieves detectability for one watermark, Gaussian Shading, with a Z-score of 3.02. However, Z-scores remain

lower than other datasets, and two out of three watermarks still fall below the detectability threshold. The low Z-scores of the numerical-only Pollution variant is unsurprising. Its gap between the oracle JS distance and post-DI JS distance remains substantially larger than other datasets. The divergence between oracle and post decoder inversion distributions appears to play a part in watermark recovery success. Greater divergence indicates information loss or distortion during decoding and inversion. This lost information may contain crucial components of the watermark signal, reducing the Z-score.

Nevertheless, depending on the dataset detectability can still be maintained through careful selection of the watermarking method. Thus, a poor approximation of the Gaussian distribution does not necessarily preclude good watermark detectability. However, it does make the choice of watermark more critical. Datasets where decoder inversion cannot preserve sufficient information present a greater challenge. For such cases, refining the decoder inversion procedure may be necessary to achieve more accurate latent reconstruction.

Experiment Summary: We compare watermark detectability between oracle (direct latent access) and decoder inversion scenarios across six datasets. Z-scores above 1.64 indicate reliable detection. This reveals information loss during decoding and inversion.

Key Takeaway: All watermarks experience substantial Z-score drops after decoder inversion compared to oracle scenarios. Gaussian Shading maintains detectability across four datasets despite drops. Despite yielding lower Z-scores than Gaussian Shading, TimeWak remains detectable on four datasets as well. TreeRing becomes undetectable across all datasets. Z-score degradation correlates more strongly with oracle-to-post-inversion distributional divergence than absolute Gaussian alignment.

5.6. Conclusion

Our findings indicate that latent diffusion models are not well-suited for generating synthetic time series data. While these models demonstrate good performance on small datasets composed of solely numerical features, their effectiveness drops significantly on datasets more reflective of real-world time series, which often contain a large number of heterogeneous features. This reduces the practical applicability of latent diffusion models for time series synthesis in realistic scenarios.

Despite these limitations, the experiments show that it is technically feasible to embed an imperceptible yet detectable and robust watermark during sampling. However, the utility of watermarking becomes questionable when the generated synthetic data is visibly distinguishable from the real data, making the watermark redundant.

The main bottleneck appears to be the VAE of the latent diffusion model. Depending on the dataset, the VAE fails to adequately enforce a standard Gaussian distribution on the latent space, suggesting that it might not be capturing key characteristics of the input data. This likely contributes to the inconsistent quality of the synthetic time series. By contrast, diffusion models that operate directly on the data achieve better results in terms of both data quality and watermark performance. This suggests that the VAE may be more of a hindrance than a help in this latent diffusion model.

If an autoencoder better capable of effectively encoding a wide variety of time series into a latent space with the given prior distribution can be found, latent diffusion models might become a more viable alternative. Until such an advancement is made, standard diffusion models remain the more reliable choice for high-quality time series synthesis and watermarking.

6

Robustness of Time Series Watermarks

In this chapter, we aim to extend the *attack evaluation* of time series watermarking. Therein considering more realistic attacks, leveraging aspects of the time series themselves. Current works on time series watermarking only consider simple, time domain-based attacks. However, real-world time series data is subjected to a broader range of data modification techniques, even in non-adversarial settings. These include operations not only in the time domain, but also in the frequency domain and joint time-frequency domain. Consequently, a robust watermark for time series should be resilient to such manipulations, regardless of the domain in which they occur.

This raises a critical question: are existing watermarks robust against sophisticated post-editing attacks? To address this gap, we developed a comprehensive robustness evaluation framework that tests TimeWak and the two baseline watermarking methods we used previously, TreeRing and Gaussian Shading, against a diverse suite of domain-specific attacks. Through our evaluation, we first examine how robust TimeWak is against a diverse range of domain-spanning attacks. Second, we compare its robustness to existing watermarking methods not designed for time series data.

6.1. Frequency Domain Transformations

Frequency domain transformations break down signals into different frequency parts, revealing patterns not easily visible in the time domain. These transformations enable manipulation of specific frequency coefficients. As a result, they are used for various applications, including adversarial attacks. By operating in the frequency domain, attackers can selectively distort or remove frequency components while maintaining overall signal coherence. If performed well, these attacks could potentially compromise watermark detectability without obvious degradation to signal quality.

A commonly used transformation is the **Discrete Cosine Transform** (DCT) [58]. It transforms a signal to its corresponding representation in the frequency domain. To do so, each signal is transformed from a real number, into a sum of cosine functions oscillating at different frequencies. Several variations of the discrete cosine transform exist, but DCT-II is the most commonly used. It is therefore often referred to as simply DCT. In this work we follow this convention, as DCT-II is the variant employed in our evaluation.

Given a real-valued signal $X \in \mathbf{R}^{\mathrm{n}}$ of length N, where $n = 0, 1, ..., N-1$, the DCT is defined as:

$$X[k] = \alpha(k) \sum_{n=0}^{N-1} x[n] \cdot \cos\left(\frac{\pi}{N}\left(n + \frac{1}{2}\right)k\right), \quad k = 0, 1, \ldots, N-1.$$

In the above formula, $X[k]$ denotes the DCT coefficient corresponding to the k-th cosine basis function. These coefficients quantify the contribution of different cosine frequency components in reconstructing the original signal.

The normalization factor $\alpha(k)$ is given by:

$$\alpha(k) = \begin{cases} \sqrt{\frac{1}{N}} & \textbf{if } k = 0, \\ \sqrt{\frac{2}{N}} & \textbf{otherwise}. \end{cases}$$

DCT exhibits a characteristic energy distribution where coefficients at lower values of k have larger magnitudes than those at higher indices. Energy, defined as the sum of squared coefficients, becomes concentrated in these low-frequency components through a property known as strong energy compaction. Its strong energy compaction enables DCT to compress data without significantly degrading the data quality.

The inverse DCT reconstructs the original time domain sequence from its frequency domain representation:

$$x[n] = \sum_{k=0}^{N-1} \alpha(k) \cdot X[k] \cdot \cos\left(\frac{\pi}{N}\left(n + \frac{1}{2}\right)k\right), \quad n = 0, 1, \ldots, N-1.$$

Another common transformation is the **Discrete Fourier Transform** (DFT). DFT also transforms a signal to the frequency domain, decomposing it into sine and cosine components. Unlike the DCT, which only yields real output, the DFT yields both real and imaginary components. Given a signal $x[n]$ of N complex values, where $n = 0, 1, ..., N-1$, the DFT is defined as:

$$X[k] = \sum_{n=0}^{N-1} x[n] \cdot e^{-j\frac{2\pi}{N}kn}, \quad k = 0, 1, \ldots, N-1.$$

Here, $X[k]$ denotes the complex frequency domain representation of the signal. e are the basis functions corresponding to complex exponentials of increasing frequency. The index k represents a frequency bin, with $X[k]$ describing the amplitude and phase of the frequency component at k. The amplitude constitutes the real part of the complex output, while the phase constitutes the imaginary part.

The inverse DFT allows for reconstruction of the original time domain signal from its frequency domain representation and is given by:

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] \cdot e^{j\frac{2\pi}{N}kn}, \quad n = 0, 1, \ldots, N-1$$

Due to the computational complexity of DFT, which is $O(N^2)$, more efficient variations of the DFT have been proposed. One of those is the Fast Fourier Transform (FFT)[12], which significantly improves performance by reducing the computational complexity to $O(N \log N)$.

A limitation of purely frequency-based transformations is that they provide no temporal information. While they show which frequencies are present in a signal, they cannot indicate when these frequencies occur. This loss of temporal information makes frequency-only transforms insufficient for analyzing signals of which the frequency composition changes over time.

6.2. Joint Time-Frequency Transformations

Joint time-frequency domain transformations can identify both the timing and frequency characteristics of signal components simultaneously [22]. Frequency-only transformations show what frequencies are

present overall. Unlike those methods, joint time-frequency transformations reveal how the frequency composition evolves throughout the duration of the signal. This combined perspective makes them suitable for analyzing signals that change over time. In addition, it enables selective modification of particular time-frequency regions.

The joint time-frequency transformation employed in this work is the **Discrete Wavelet Transform** (DWT) [22]. DWT breaks a signal down into components at different frequency bands and time intervals, transforming it to the joint time-frequency domain. Unlike the DCT and DFT, which decompose a signal into globally defined sinusoidal basis functions, the DWT uses wavelets, which are localized in both time and frequency.

Given a discrete time signal $x[n]$, the DWT performs a multi-resolution analysis by repeatedly passing the signal through two filters: a low-pass filter $h[n]$ associated with the scaling function and a high-pass filter $g[n]$ associated with the wavelet function. After each filtering step, the output is down sampled by a factor of two, producing:

$$a[n] = \sum_k x[k] \cdot h[2n-k], \qquad d[n] = \sum_k x[k] \cdot g[2n-k].$$

Here, $a[n]$ are the approximation coefficients which represent the low-frequency content, and $d[n]$ are the detail coefficients, which represent the high-frequency content.

The inverse DWT reconstructs the original signal with the corresponding reconstruction filters $h[n]$ and $g[n]$:

$$x[n] = \sum_k \left(a[k] \cdot h'[n-2k] + d[k] \cdot g'[n-2k]\right)$$

The DWT can be performed with a number of wavelets, such as Haar [59] or Daubechies [19]. Each wavelet offers different trade-offs between time localization, frequency localization and the ability to represent smooth versus abrupt changes in the signal.

6.3. Methodology

We developed a comprehensive attack suite by drawing inspiration from common data augmentation techniques used in time series literature. They involve transformations based on the DFT, DCT, and DWT, in addition to more conventional time-domain modifications. A complete overview of the attacks can be found in Figure 6.1.

For each attack, we assess the detectability of the watermark by computing the Z-score of the watermark post-attack. A watermark is considered robust if the Z-score remains above a threshold of approximately 1.64 for p is 0.005, indicating statistically significant presence despite the applied perturbation.

6.3.1. Time Domain Attacks

The original TimeWak evaluation assessed robustness against three simple time-domain attacks: offset, cropping, and min-max insertion. These transformations represent plausible manipulations a watermark might encounter [66]. However, due to their nature they provide only a limited perspective on TimeWak's overall robustness within the time domain. To address this limitation, we draw on established time series data augmentation techniques [66] to design a more comprehensive and challenging suite of time-domain attacks. Our proposed suite includes both elementary and refined attacks to rigorously evaluate TimeWak's robustness. The complete set of time-domain attacks is visualized in Figure 6.2 and comprises the following:

- **Offset**: This attack shifts the baseline of the entire time series by adding a scaled version of the mean of the time series to all time points. This perturbation tests watermark resilience to linear translations in the time domain.

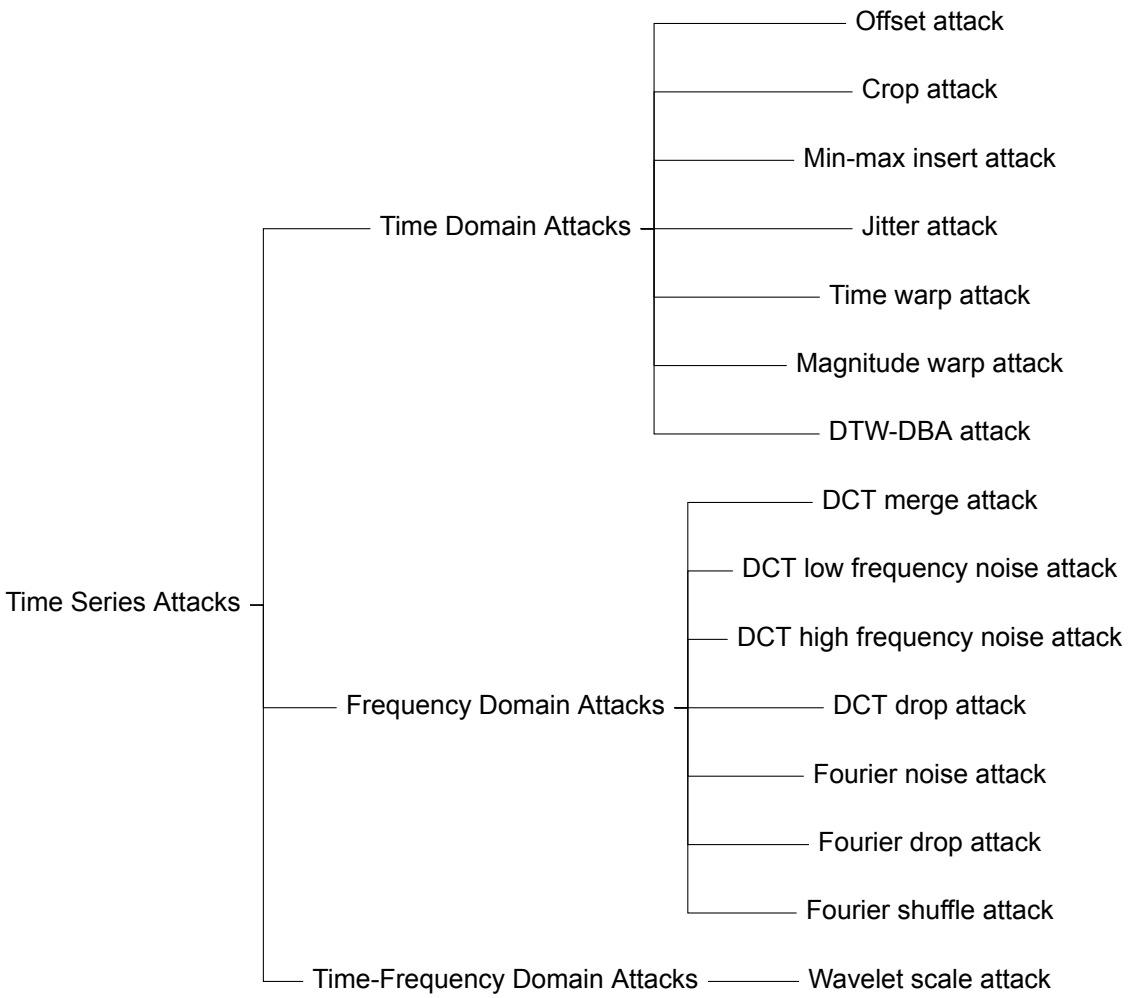
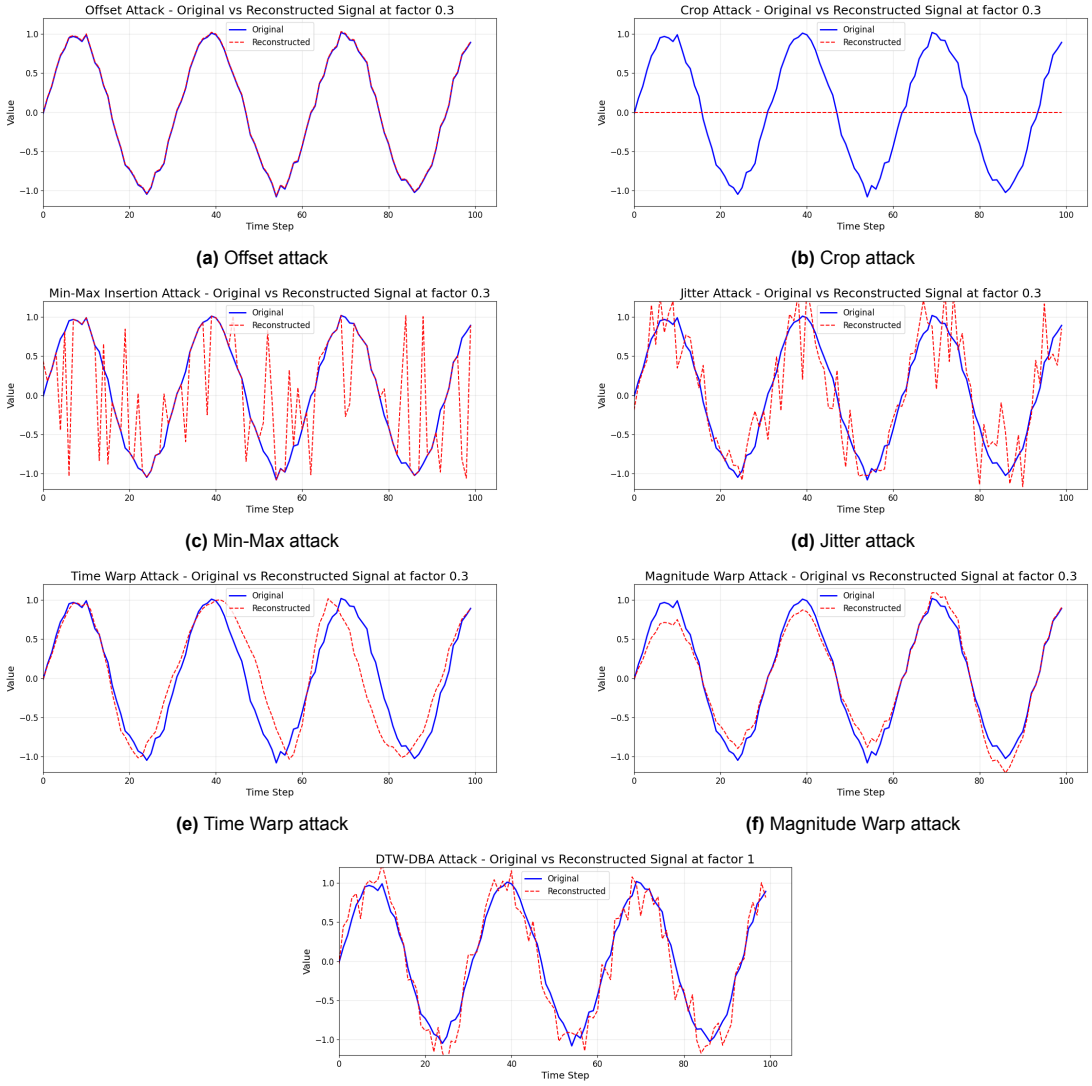


Figure 6.1: A taxonomy of propose time series attacks, grouped by the domain they target: time domain, frequency domain, or joint time-frequency domain.

- **Crop**: The crop attack emulates partial signal loss by removing a randomly selected region from the time series and padding the remainder with zeros. This form of cropping can disrupt local patterns in both time and feature space, challenging the watermark's spatial-temporal localization.
- **Min-max insert**: For each time series, a number of random time steps are selected according to the specified attack factor, and the values at those time steps are overwritten with random values drawn uniformly from the minimum to the maximum values observed in that feature. By injecting plausible but incorrect values into arbitrary locations, the insert attack targets the integrity of both local and global structures in the time series.
- **Jitter**: The jitter attack simulates random, high-frequency sensor noise by adding Gaussian-distributed noise to every point in the time series. This additive noise distorts fine-grained features of the time series without introducing large-scale structure changes, assessing sensitivity to small local changes.
- **Time warp** [64]: The time warp attack performs nonlinear temporal distortions by resampling the original time series along a smooth, warping curve. This attack tests the watermark's robustness to time misalignment and sampling irregularities.
- **Magnitude warp** [64]: This attack distorts the time series by adjusting its amplitude over time using a smooth multiplicative curve. Unlike the time warp attack, which perturbs the temporal axis, magnitude warping alters the time series measurements while preserving temporal alignment. It challenges watermarking techniques that depend on consistent amplitude or local magnitude.



(a) Offset attack

(b) Crop attack

(c) Min-Max attack

(d) Jitter attack

(e) Time Warp attack

(f) Magnitude Warp attack

(g) DTW-DBA attack

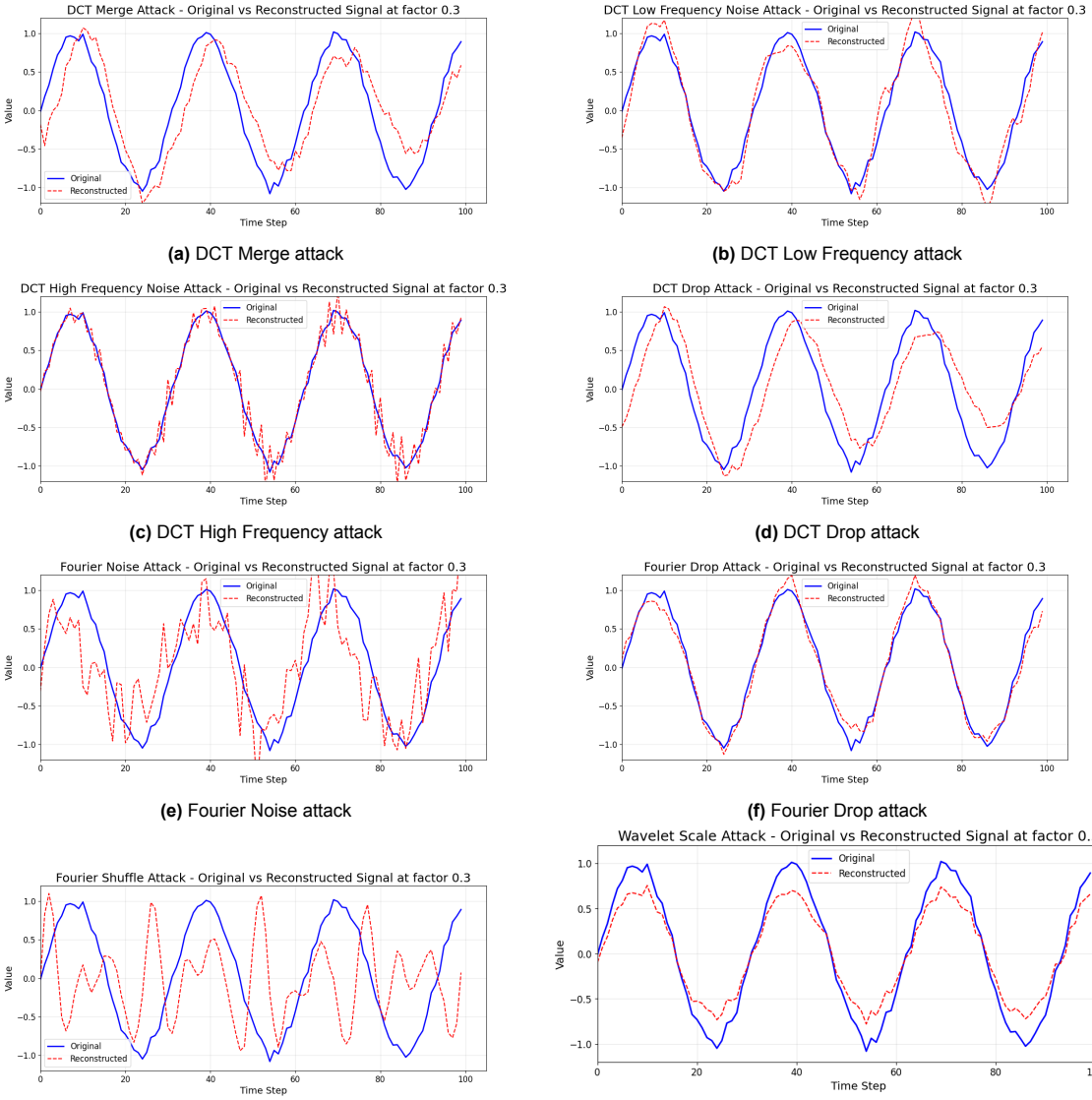
Figure 6.2: Time domain attacks at strength 0.3 on a low frequency sine signal.

- **DTW-DBA** [15]: The DTW-DBA (Dynamic Time Warping - Barycentric Averaging) attack constructs several noisy variants of the given time series and combines them using DTW-based barycentric averaging. This attack produces a smoothed, synthetic sequence that retains global similarity to the original but potentially eliminates subtle watermark signals.

Each attack was performed at two levels of intensity, 0.05 and 0.3. Thereby, providing insight into the watermarking method's sensitivity to varying degrees of perturbation. The only exception was the DTW-DBA attack, which does not have a tunable attack strength.

6.3.2. Domain Transformation Attacks

We continued our evaluation with domain transformation attacks. The initial TimeWak evaluation did not consider attacks in domains beyond the time domain. As a result, potential vulnerabilities in the frequency and time-frequency domains were overlooked. Similarly, the baseline watermarks we compare against, TreeRing and Gaussian Shading, have not been evaluated against such attacks either despite the viable avenue that other domains provide to adversaries. To address gaps in existing evaluations, we extended the attack suite with frequency domain and time-frequency domain attacks using the DCT,



(a) DCT Merge attack

(b) DCT Low Frequency attack

(c) DCT High Frequency attack

(d) DCT Drop attack

(e) Fourier Noise attack

(f) Fourier Drop attack

(g) Fourier Shuffle attack

(h) Wavelet Scale attack

Figure 6.3: Domain transformation attacks at strength 0.3 on a low frequency sine signal.

DFT and DWT. As with our time domain attacks, most attacks were derived fromdata augmentation techniques for time series. Naturally a robust watermark should remain detectable even in scenarios where no perturbations have been performed. The domain transformation attack suite consists of the following attacks:

- **DCT drop**: This attack removes spectral content by zeroing out a random selection of DCT coefficients. Setting these coefficients to zero removes their contribution to the reconstructed time series. By dropping specific frequency bands, this attack distorts the frequency composition of the time series, changing the ratio between low frequency measurements and high frequency measurements.
- **DCT high/low frequency noise**: The high frequency noise attack injects Gaussian noise into the high-frequency region of the DCT spectrum. This region corresponds to the smallest time-scale structures. The attack selectively targets sharp transitions between consecutive time steps and local features. In doing so, it can potentially degrade watermark patterns without affecting broader trends. In contrast to its high-frequency counterpart, the low frequency noise attack targets the low-frequency components of the DCT spectrum. These components correspond to slow-varying

trends or global signal shape. Adding noise to the low frequency components distorts the global time series structure while maintaining the fine-grained structures.

- **DCT merge** [17]: The DCT merge attack perturbs the signal in the frequency domain by selectively averaging randomly chosen frequency components. This flattens variations in the frequency spectrum, effectively removing fine-grained frequency distinctions.
- **Fourier drop**: This attack removes energy from specific frequencies by zeroing out symmetric pairs of FFT coefficients. The resulting time-domain time series is globally altered but remains smooth and continuous, muting detailed spectral patterns while preserving overall coherence.
- **Fourier noise** [16]: The Fourier noise attack perturbs both amplitude and phase information in the frequency domain. The input time series is transformed using the FFT, separating it into amplitude and phase components. Some selected frequency bins have their amplitudes replaced with normally distributed values. Other bins receive phase perturbations, altering both signal shape and timing.
- **Fourier shuffle** [78]: This attack selectively disrupts dominant spectral components by shuffling the frequencies with the highest magnitudes. This attack introduces unpredictable changes to dominant oscillatory patterns without altering the overall energy.
- **Wavelet scale**: The wavelet scale attack modifies the low-frequency content of a signal by manipulating its wavelet decomposition. This attack reduces the global amplitude of the time series while preserving detail at higher frequencies.

Each attack was conducted twice at an intensity of 0.05 and 0.3. For most attacks, the intensity was used to determine the number of samples to target, or the number of frequency coefficients to select.

6.4. Results

We evaluate the robustness of TimeWak, TreeRing and Gaussian Shading against our comprehensive attack suite spanning time-domain and domain transformation attacks. Each watermark was embedded in four time series datasets (ETTh, Energy, fMRI, Stocks) and subjected to attacks at two intensity levels (0.05 and 0.3). We measure robustness using Z-scores, where values above 1.64 indicate statistically significant watermark detection ($p < 0.05$). We also use critical difference diagrams to identify statistically significant performance differences.

Table 6.1: Z-scores (with standard deviations between parentheses) of time series for different Watermarking (WM) methods under simple temporal domain attacks. GS = Gaussian Shading, TR = TreeRing, TW = TimeWak. Attack intensities of 0.05 and 0.3 are shown. Best results are shown in bold.

Dataset	WM	Unattacked	Offset		Crop		Min-Max Insert		Jitter	
			0.05	0.3	0.05	0.3	0.05	0.3	0.05	0.3
Energy	GS	45.31 ± (0.80)	43.66 ± (0.76)	28.43 ± (0.88)	-2.60 ± (0.86)	**38.65** ± (1.37)	42.25 ± (0.89)	26.38 ± (1.00)	56.67 ± (0.77)	**47.98** ± (0.92)
	TR	-1.55 ± (0.03)	-1.56 ± (0.03)	-1.93 ± (0.03)	-110.62 ± (0.30)	-137.02 ± (0.25)	-11.48 ± (0.17)	-31.37 ± (0.17)	-2.21 ± (0.03)	-12.12 ± (0.03)
	TW	**224.34** ± (1.74)	**223.60** ± (1.67)	**187.66** ± (1.72)	**0.69** ± (1.01)	9.22 ± (0.84)	**184.55** ± (1.42)	**57.85** ± (1.11)	**145.56** ± (1.13)	27.78 ± (0.87)
ETTh	GS	107.23 ± (1.22)	104.07 ± (1.31)	84.40 ± (0.99)	**73.22** ± (1.18)	**24.53** ± (1.10)	**99.63** ± (0.96)	**66.25** ± (1.12)	**111.38** ± (1.21)	**109.66** ± (1.26)
	TR	-4.19 ± (0.08)	-4.12 ± (0.08)	-3.73 ± (0.09)	-14.07 ± (0.24)	-19.91 ± (0.15)	-7.91 ± (0.14)	-16.96 ± (0.28)	-3.88 ± (0.07)	-5.72 ± (0.06)
	TW	**128.24** ± (1.14)	**124.38** ± (1.05)	**112.85** ± (1.05)	34.15 ± (0.88)	4.31 ± (0.84)	97.50 ± (0.90)	19.48 ± (0.97)	108.83 ± (1.13)	33.87 ± (0.89)
fMRI	GS	365.62 ± (1.02)	365.94 ± (1.22)	366.67 ± (1.23)	**319.55** ± (0.96)	**179.20** ± (0.91)	**352.84** ± (1.27)	**248.61** ± (1.22)	359.04 ± (0.96)	321.18 ± (1.01)
	TR	4.37 ± (0.05)	4.41 ± (0.06)	4.49 ± (0.06)	3.67 ± (0.06)	3.70 ± (0.06)	3.29 ± (0.06)	-1.32 ± (0.07)	4.60 ± (0.06)	4.47 ± (0.06)
	TW	**386.13** ± (0.88)	**385.99** ± (0.83)	**382.79** ± (0.85)	277.53 ± (0.93)	75.12 ± (0.87)	333.56 ± (0.89)	135.68 ± (1.01)	**383.47** ± (0.93)	**344.86** ± (0.95)
Stocks	GS	75.79 ± (0.64)	75.64 ± (0.54)	73.28 ± (0.57)	-28.65 ± (0.53)	6.26 ± (0.59)	72.48 ± (0.62)	50.16 ± (0.64)	75.12 ± (0.55)	75.36 ± (0.73)
	TR	2.08 ± (0.02)	2.05 ± (0.02)	1.89 ± (0.02)	-53.30 ± (0.97)	-67.62 ± (1.18)	1.69 ± (0.03)	0.07 ± (0.05)	0.62 ± (0.03)	0.61 ± (0.03)
	TW	**182.48** ± (0.78)	**182.58** ± (0.85)	**181.63** ± (0.81)	**59.56** ± (1.03)	**11.32** ± (0.98)	**156.11** ± (1.01)	**56.57** ± (1.06)	**182.44** ± (0.79)	**181.35** ± (0.72)

The robustness evaluation reveals that across all attack types and domains, TreeRing exhibits the lowest robustness out of the watermarks, ranking significantly lower than TimeWak and Gaussian Shading (see Figure 6.4). TimeWak emerges as the most robust watermark, followed by Gaussian Shading.

TimeWak and Gaussian Shading demonstrate high Z-scores across all scenarios. Their Z-scores typically range in the hundreds in the unattacked scenario, and even after some attacks. While certain

Table 6.2: Z-scores (with standard deviations between parentheses) of time series watermarked with different watermarks (WM) under complex temporal domain attacks. GS = Gaussian Shading, TR = TreeRing, TW = TimeWak. Attack intensities of 0.05 and 0.3 are shown (except for DTW-DBA which has no intensity parameter). Best results are shown in bold.

Dataset	WM	Unattacked	Time Warp		Magnitude Warp		DTW-DBA
			0.05	0.3	0.05	0.3	
Energy	GS	45.31 ± (0.80)	40.68 ± (0.94)	23.47 ± (0.92)	**52.84** ± (1.07)	**28.34** ± (1.15)	**46.03** ± (0.86)
	TR	-1.55 ± (0.03)	-1.58 ± (0.03)	-1.84 ± (0.04)	-18.72 ± (0.18)	-31.19 ± (0.23)	-8.23 ± (0.03)
	TW	**224.34** ± (1.74)	**192.31** ± (1.43)	**48.52** ± (1.59)	7.28 ± (0.84)	-0.56 ± (0.92)	43.33 ± (1.04)
ETTh	GS	107.23 ± (1.22)	100.89 ± (1.44)	**71.25** ± (1.16)	100.00 ± (1.08)	22.32 ± (0.79)	**100.51** ± (1.10)
	TR	-4.19 ± (0.08)	-4.00 ± (0.08)	-4.39 ± (0.08)	-3.71 ± (0.07)	-1.33 ± (0.03)	-4.85 ± (0.07)
	TW	**128.24** ± (1.14)	**108.75** ± (1.10)	29.87 ± (1.19)	**108.82** ± (1.03)	**38.47** ± (1.11)	50.87 ± (0.93)
fMRI	GS	365.62 ± (1.02)	330.75 ± (1.22)	**196.25** ± (1.83)	326.88 ± (0.98)	189.86 ± (0.89)	336.44 ± (1.00)
	TR	4.37 ± (0.05)	4.95 ± (0.06)	3.59 ± (0.06)	4.91 ± (0.04)	2.34 ± (0.05)	4.37 ± (0.06)
	TW	**386.13** ± (0.88)	**340.34** ± (1.22)	95.59 ± (2.48)	**363.55** ± (0.77)	**298.11** ± (1.05)	**362.79** ± (0.75)
Stocks	GS	75.79 ± (0.64)	72.09 ± (0.53)	49.17 ± (0.68)	66.96 ± (0.63)	19.25 ± (0.74)	72.62 ± (0.66)
	TR	2.08 ± (0.02)	0.59 ± (0.03)	0.47 ± (0.03)	0.01 ± (0.03)	-5.07 ± (0.15)	0.61 ± (0.03)
	TW	**182.48** ± (0.78)	**163.56** ± (0.93)	**49.93** ± (1.28)	**97.59** ± (1.30)	**20.53** ± (1.07)	**181.80** ± (0.88)

Table 6.3: Z-scores (with standard deviations between parentheses) of time series watermarked with different watermarks (WM) under DCT-based domain transformation attacks. GS = Gaussian Shading, TR = TreeRing, TW = TimeWak. Attack intensities of 0.05 and 0.3 are shown. Best results are shown in bold.

Dataset	WM	Unattacked	DCT Drop		DCT High Freq. Noise		DCT Low Freq. Noise		DCT Merge	
			0.05	0.3	0.05	0.3	0.05	0.3	0.05	0.3
Energy	GS	45.31 ± (0.80)	**43.62** ± **(0.72)**	**41.74** ± **(0.85)**	52.49 ± (0.77)	**47.52** ± (0.85)	53.89 ± (0.84)	56.97 ± (0.97)	**47.60** ± (0.84)	**44.83** ± (1.06)
	TR	-1.55 ± (0.03)	-33.87 ± (0.24)	-50.54 ± (0.19)	-2.80 ± (0.04)	-6.01 ± (0.03)	-2.02 ± (0.03)	-5.13 ± (0.03)	-52.61 ± (0.76)	-74.86 ± (0.28)
	TW	**224.34** ± (1.74)	16.76 ± (0.99)	15.00 ± (0.94)	**98.31** ± (1.06)	47.18 ± (0.90)	**153.39** ± (1.25)	**59.91** ± (0.91)	14.96 ± (0.87)	6.76 ± (1.00)
ETTh	GS	107.23 ± (1.22)	**102.20** ± (1.34)	**67.94** ± (1.32)	**105.73** ± (1.11)	**113.14** ± (1.12)	106.93 ± (1.02)	**107.84** ± (1.08)	**91.21** ± (1.20)	**61.75** ± (0.99)
	TR	-4.19 ± (0.08)	-2.10 ± (0.11)	-6.95 ± (0.15)	-3.34 ± (0.07)	-5.04 ± (0.07)	-2.70 ± (0.07)	-1.96 ± (0.06)	-4.11 ± (0.14)	-13.21 ± (0.21)
	TW	**128.24** ± (1.14)	42.62 ± (0.98)	13.17 ± (0.93)	87.47 ± (1.04)	55.70 ± (1.06)	**112.79** ± (0.97)	67.22 ± (0.90)	46.49 ± (1.26)	4.97 ± (0.90)
fMRI	GS	365.62 ± (1.02)	234.88 ± (0.85)	**228.22** ± (1.03)	360.17 ± (1.10)	356.34 ± (0.87)	366.15 ± (1.21)	342.25 ± (1.13)	303.75 ± (0.99)	**227.37** ± (0.98)
	TR	4.37 ± (0.05)	4.49 ± (0.05)	2.70 ± (0.07)	4.18 ± (0.05)	4.28 ± (0.05)	4.60 ± (0.05)	5.44 ± (0.05)	4.72 ± (0.06)	2.83 ± (0.06)
	TW	**386.13** ± (0.88)	**293.82** ± (0.95)	163.43 ± (0.87)	**377.27** ± (0.69)	**361.76** ± (0.92)	**380.29** ± (0.97)	**370.21** ± (0.77)	**326.96** ± (0.93)	174.95 ± (0.86)
Stocks	GS	75.79 ± (0.64)	31.62 ± (1.43)	-0.83 ± (1.10)	75.35 ± (0.64)	73.77 ± (0.62)	76.27 ± (0.67)	76.39 ± (0.63)	24.72 ± (1.29)	-3.12 ± (1.03)
	TR	2.08 ± (0.02)	-8.34 ± (0.31)	-20.34 ± (0.39)	0.61 ± (0.03)	0.61 ± (0.04)	0.61 ± (0.03)	0.60 ± (0.03)	-11.42 ± (0.60)	-32.19 ± (0.81)
	TW	**182.48** ± (0.78)	**102.58** ± (1.32)	**46.72** ± (1.13)	**182.10** ± (0.74)	**182.30** ± (0.73)	**182.39** ± (0.81)	**182.08** ± (0.85)	**77.95** ± (1.50)	**9.91** ± (1.00)

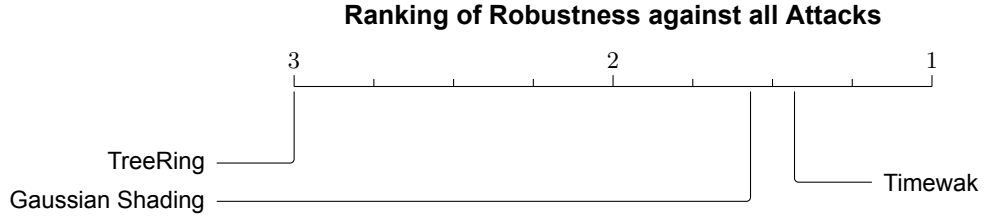


Figure 6.4: Critical Difference diagram of robustness rankings across all attacks of evaluated methods. Lower rank is better.

attacks manage to substantially reduce these scores, the post-attack Z-scores frequently remain far above the detectability threshold of 1.64. Notably, TimeWak achieves its robustness while also maintaining superior quality metrics compared to Gaussian Shading. This suggests TimeWak possesses the ability to introduce a subtle watermark that preserves detectability under adversarial attacks.

TreeRing's performance shows substantial variation across datasets. Even in unattacked scenarios, TreeRing fails to achieve detectable watermark signals for two datasets (ETTh and Energy). As a result, post-attack detection is impossible for these cases. Among datasets where TreeRing does embed detectable watermarks, its performance is inconsistent. The watermark remains poorly robust when embedded in the Stocks dataset, becoming undetectable after nearly every attack regardless of intensity. Conversely, TreeRing demonstrates better resilience when applied to the fMRI dataset, surviving most attacks with detectable Z-scores. This dataset-dependent behavior appears unrelated to the number of features, as TreeRing's best performing datasets (fMRI with 49 features and Stocks

Table 6.4: Z-scores (with standard deviations between parentheses) of time series watermarked with different watermarks (WM) under Fourier and Wavelet domain transformation attacks. GS = Gaussian Shading, TR = TreeRing, TW = TimeWak. Attack intensities of 0.05 and 0.3 are shown. Best results are shown in bold.

Dataset	WM	Unattacked	Fourier Drop		Fourier Noise		Fourier Shuffle		Wavelet Scale	
			0.05	0.3	0.05	0.3	0.05	0.3	0.05	0.3
Energy	GS	45.31 ± (0.80)	40.13 ± (0.88)	**38.15** ± (0.81)	**55.64** ± (0.94)	**42.61** ± (1.09)	45.02 ± (0.83)	**40.04** ± (1.08)	56.08 ± (0.83)	63.78 ± (0.84)
	TR	-1.55 ± (0.03)	-7.26 ± (0.06)	-14.71 ± (0.10)	-38.30 ± (0.12)	-54.81 ± (0.14)	-1.55 ± (0.04)	-136.27 ± (0.44)	-1.56 ± (0.03)	-2.91 ± (0.03)
	TW	**224.34** ± (1.74)	**61.78** ± (0.91)	31.32 ± (1.01)	0.98 ± (0.98)	1.86 ± (1.06)	**224.40** ± (1.73)	2.38 ± (0.94)	**221.92** ± (1.36)	**147.17** ± (1.27)
ETTh	GS	107.23 ± (1.22)	**100.73** ± (1.04)	**75.39** ± (1.14)	**38.00** ± (1.09)	**62.07** ± (1.14)	107.31 ± (1.13)	**49.14** ± (0.97)	107.39 ± (1.17)	94.98 ± (0.98)
	TR	-4.19 ± (0.08)	-4.23 ± (0.08)	-6.36 ± (0.12)	-3.69 ± (0.07)	-11.63 ± (0.14)	-4.18 ± (0.08)	-19.54 ± (0.22)	-4.23 ± (0.08)	-4.54 ± (0.08)
	TW	**128.24** ± (1.14)	50.37 ± (0.99)	14.88 ± (0.97)	11.60 ± (1.02)	1.71 ± (0.85)	**128.31** ± (1.08)	-0.91 ± (0.93)	**130.60** ± (1.00)	**136.20** ± (0.99)
fMRI	GS	365.62 ± (1.02)	**303.29** ± (1.03)	**245.56** ± (1.15)	**302.28** ± (0.96)	**202.49** ± (0.75)	**365.85** ± (0.91)	**218.29** ± (1.14)	368.57 ± (1.01)	304.93 ± (0.83)
	TR	4.37 ± (0.05)	4.29 ± (0.05)	3.28 ± (0.06)	4.49 ± (0.05)	1.06 ± (0.05)	4.38 ± (0.05)	3.51 ± (0.06)	4.28 ± (0.05)	3.53 ± (0.06)
	TW	**386.13** ± (0.88)	290.90 ± (0.84)	155.72 ± (0.86)	290.55 ± (0.92)	158.24 ± (0.86)	386.31 ± (0.89)	193.94 ± (0.97)	**387.38** ± (0.83)	**361.88** ± (0.88)
Stocks	GS	75.79 ± (0.64)	58.07 ± (0.74)	**48.98** ± (0.68)	**12.96** ± (0.66)	**16.77** ± (0.66)	75.83 ± (0.55)	**6.16** ± (0.62)	75.66 ± (0.61)	70.56 ± (0.56)
	TR	2.08 ± (0.02)	0.49 ± (0.03)	0.25 ± (0.04)	-18.57 ± (0.28)	-23.50 ± (0.42)	0.62 ± (0.03)	-56.06 ± (0.84)	0.62 ± (0.03)	0.63 ± (0.03)
	TW	**182.48** ± (0.78)	**117.16** ± (1.25)	47.66 ± (1.20)	6.63 ± (0.98)	0.66 ± (0.96)	**182.30** ± (0.84)	-0.44 ± (0.98)	**182.69** ± (0.81)	**170.23** ± (0.84)

with 6 features respectively) sit at opposite ends of the spectrum. On the other hand, the number of time steps may play a role in TreeRing's performance. Datasets in which TreeRing is undetectable (ETTh and Energy) contain nearly twice as many time steps as fMRI. The performance difference between these datasets suggest that TreeRing may have an optimal number of time steps where it can effectively disperse its signal without becoming diluted.

Experiment Summary: We evaluate the robustness of three watermarking methods (TimeWak, TreeRing, and Gaussian Shading) against a comprehensive attack suite. Each watermark is embedded in four time series datasets (ETTh, Energy, fMRI, Stocks) and subjected to attacks at two intensity levels (0.05 and 0.3). Robustness is measured using Z-scores, where values above 1.64 indicate statistically significant detection.

Key Takeaway: TimeWak emerges as the most robust watermark overall, followed by Gaussian Shading. TreeRing exhibits the lowest robustness. TimeWak maintains superior quality metrics while achieving high robustness, suggesting it can introduce subtle watermarks that preserve detectability under adversarial conditions.

6.4.1. Time Domain Attacks

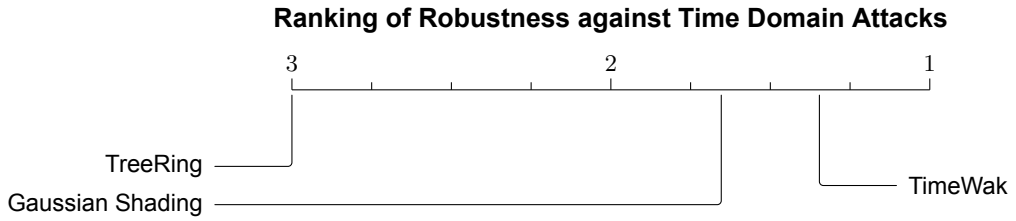


Figure 6.5: Critical Difference diagram of robustness rankings after temporal domain attacks. Lower rank is better.

To better understand the specific approaches that pose the greatest threats to each watermarking method, we analyze robustness within individual domains. We begin by analyzing the impact of time-domain attacks, which represent the most direct form of attack. When considering each domain in isolation, TimeWak demonstrates the greatest robustness in the time domain. It achieves significantly higher Z-scores than the runner up Gaussian Shading as shown in Figure 6.5. Despite its strong performance, TimeWak falters against Cropping attacks and Magnitude Warp attacks on watermarked Energy data. The watermark fails to remain detectable after facing these attacks at strengths 0.05 and 0.3 respectively (Table 6.1 and Table 6.2). Gaussian Shading also becomes undetectable after 0.05 strength Cropping attacks on watermarked Energy data, but does survive the Magnitude Warp attack.

Surprisingly, in some datasets, mild Cropping attacks lead to larger Z-score drops than more aggressive

variations for the TimeWak and Gaussian Shading watermarks. Such behavior may be attributed to how watermark verification is performed. For Gaussian Shading, evaluation relies on whether the signs of recovered noise match a fixed random bit sequence. For TimeWak, the bit sequence of every second row is compared to its predecessor after an inverse permutation. Following heavy cropping, a row might consists entirely of zeros. The outcome then becomes invariant to permutation, artificially boosting the match rate. Similarly, if the latent bits in Gaussian Shading contain many zeros, zeroed regions might coincide with expected values, leading to higher scores.

TreeRing fails against most attacks, only consistently withstanding attacks when applied to fMRI. We hypothesize that the temporal length of fMRI allows TreeRing to embed its signal well enough to withstand attacks. We hypothesize that fMRI contains sufficient temporal length for TreeRing to embed a robust watermark signal. This length enables TreeRing to withstand attacks, while being short enough to prevent the watermark from becoming too dispersed for detection. In contrast to the other watermarks, it even fails to withstand the simple attacks like Min-Max Insert and Jitter, dropping below the detectability threshold on synthetic Stocks data. Out of all time domain attacks, Cropping, Time Warp and Magnitude Warp cause the largest drops in Z-score. However, with the exception of attacks performed on synthetic Energy data, Gaussian Shading and especially TimeWak remain far above the detectability threshold after these attacks.

Experiment Summary: We analyze the impact of time-domain attacks on watermark detectability. These attacks represent the most direct form of manipulation, operating on raw temporal values.

Key Takeaway: TimeWak demonstrates the greatest robustness in the time domain, achieving significantly higher Z-scores than Gaussian Shading. However, both watermarks falter against Cropping attacks on Energy data. TreeRing fails against most time-domain attacks. Cropping, Time Warp, and Magnitude Warp cause the largest Z-score drops, yet TimeWak and Gaussian Shading typically remain far above the detectability threshold.

6.4.2. Domain Transformation Attacks

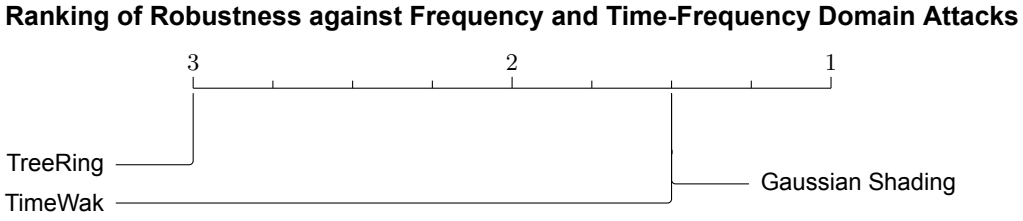


Figure 6.6: Critical Difference diagram of robustness rankings after frequency and joint time-frequency domain attacks. Lower rank is better.

In the frequency domain, TimeWak and Gaussian Shading display roughly equivalent levels of robustness (Figure 6.6). Same as with the preceding attacks, TreeRing again exhibits a poor resistance against these attacks. Even though TimeWak yields higher Z-scores on the unattacked datasets than Gaussian Shading, these higher Z-scores do not appear to translate to increased robustness to frequency attacks. Notably, Fourier-based attacks prove especially devastating to TimeWak. Both the Fourier Noise and Fourier Shuffle attacks succeed in completely removing the watermark in some datasets, whereas Gaussian Shading maintains detectability under all Fourier-related attacks (Table 6.4). We hypothesize that this could be because these methods disrupt the temporal dependencies on which TimeWak relies. Fourier Noise does so by altering both the phase and amplitude, effectively scrambling temporal relationships between consecutive rows. Fourier Shuffle achieves a similar result by reordering the frequencies most responsible for the dominant structural components of the time series. Both approaches break the sequential and spatial consistency required for the TimeWak watermark to be correctly recovered. In contrast, Gaussian Shading shows greater resilience to these types

of attacks, particularly the Fourier Shuffle attack. This resilience could be attributed to its detection implementation, which is based on sign comparisons. Gaussian Shading's detection mechanism does not rely on the precise order or temporal structure of the values. Instead, it only depends on whether their signs match a seeded pattern. Therefore, Fourier-based reordering might fail to push a sufficient number of values across the sign threshold. This prevents significant degradation of detection accuracy. In essence, Gaussian Shading gains a degree of inherent robustness from its threshold-based evaluation strategy.

DCT-based attacks affect both TimeWak and Gaussian Shading similarly, as illustrated in Table 6.3. While Gaussian Shading generally exhibits better robustness against DCT Merge attacks, it does fail under the strong variant of this attack on the Stocks dataset. The watermark is also removed by strong DCT Drop attacks on the Stocks dataset. In these particular scenarios, it is less robust than TimeWak. In general, we observe that some DCT-based attacks demonstrate greater success on the Stocks dataset than on other datasets. One possible explanation is that the Stocks time series may contain more abrupt transitions or high-frequency fluctuations. These properties lead to a less concentrated energy distribution in the DCT domain. Under such conditions, randomly removing or merging coefficients can cause meaningful signal distortion. This occurs even when targeting the higher frequency range. Once the inverse transformation is applied, this distortion can then potentially degrade or remove embedded watermarks. This effect was especially pronounced for the Gaussian Shading watermark. In these cases, the attack likely pushed many watermarked values across the median threshold of the Gaussian distribution they were sampled from. Consequently, these values had their sign flipped, thereby reducing detection accuracy.

Another consistent trend observed across datasets is the relative robustness of all watermarking methods against noise injection in the DCT frequency domain. Regardless of whether low or high-frequency coefficients were targeted, the impact of added noise is generally limited. Such limited impact contrasts sharply with the Fourier-based noise attack, which proved to be one of the most powerful overall. A plausible explanation lies in the structural differences between the two transforms: while the DCT operates solely in terms of signal amplitude, the DFT captures both amplitude and phase. Since the phase of a signal plays a critical role in determining its temporal structure, injecting noise into both phase and amplitude is more disruptive than perturbations only targeting the amplitude. This distinction likely explains why DCT noise attacks fail to fully eliminate watermarks, whereas Fourier-based noise injection often succeeds.

When subjected to a joint time-frequency attack using DWT, TimeWak is the most robust method. It consistently achieves the highest Z-scores across all datasets for this attack (see Table 6.4), demonstrating its ability to withstand attacks in joint domains. Gaussian Shading also proves its robustness in this setting, while TreeRing ranks last. We observe that the DWT attack is the least destructive of all the domain transformation attacks. The attack often fails to drastically decrease the Z-score, and occasionally even improves the detectability of the watermarks. The DWT attack only targets the coefficients at the lowest index, leaving the other coefficients intact. The low impact of the DWT attack suggests that rather than the low frequency component that controls overall trend and signal energy, the watermark signal is embedded in the higher frequency components of the DWT domain.

Experiment Summary: We evaluate robustness against frequency-domain (Fourier and DCT) and time-frequency domain (DWT) attacks. These attacks target different frequency components or combine temporal and frequency information.

Key Takeaway: TimeWak and Gaussian Shading display roughly equivalent robustness in frequency domains overall, though each has distinct vulnerabilities. TimeWak seems especially vulnerable to Fourier-based attacks. DCT-based attacks affect TimeWak and Gaussian Shading similarly. DWT attacks prove least destructive overall, with TimeWak showing the highest resilience. TreeRing again ranks last across all transformation attacks.

6.5. Conclusion

Our results confirm the superior robustness of TimeWak to a variety of attacks when compared to TreeRing and Gaussian Shading. While Gaussian Shading demonstrates competitive performance, especially under Fourier and DCT-based attacks, it is generally less consistent across domains. TreeRing, in contrast, exhibits limited robustness and fails to withstand most attacks. Taken together, the findings confirm that TimeWak provides a more resilient watermark for time series data while also preserving data quality. It strikes a better balance in the quality-detectability-robustness trade-off than both Gaussian Shading and TreeRing. Gaussian Shading provides resilience against attacks at the cost of data quality, while TreeRing preserves quality at the cost of detectability and robustness. TimeWak is the only watermark that does not incur any debt to provide robustness, particularly in scenarios involving time domain attacks. However, the results also highlight the importance of a diverse attack suite. The novel attacks uncovered new vulnerabilities in not only the TimeWak watermark but all evaluated watermarks. The frequency domain attacks in particular emerge as effective perturbation approaches. These attacks consistently cause large drops in Z-score, in some cases even removing the watermark completely.

While the present results affirm TimeWak's robustness in many scenarios, its vulnerability to specific frequency-domain transformations indicates room for improvement. Strengthening resistance to DCT and especially Fourier-based attacks will help to enhance the watermark's real-world applicability. Such improvements are particularly important in settings where adversarial or preprocessing-related modifications are common.

7

Conclusion

This thesis examines synthetic time series generation and watermarking along three dimensions: the choice of generative model, the suitability of latent diffusion models for watermarking time series, and the robustness of current watermarking methods against realistic attacks.

The first question we answer is: Which architecture generates higher quality synthetic time series? Our comparative evaluation of transformer-based and diffusion-based generative models demonstrates a decisive advantage for diffusion models. Current transformer models struggle to maintain long-range coherence and capture global statistical structure. As a result, their synthetic sequences tend to share little similarities with the data they are meant to emulate. In contrast, diffusion models succeed in generating synthetic sequences that are both statistically faithful and practically useful. They achieve better discriminative scores, predictive scores, context-FID scores and correlational scores across all datasets. This makes diffusion the more appropriate backbone for time series synthesis at the time of writing. Their superior generation quality also make them a more suitable candidate for watermarking.

The success of standard diffusion models raises the question: To what extent can latent diffusion models offer a viable approach to watermarking time series? Our emperical evaluation of both data and latent diffusion models highlights key limitations of the latter. Due to their limitations, at present LDMs do not translate well to the time series domain. While LDMs perform adequately on small datasets with homogeneous numerical features, their effectiveness deteriorates on datasets with heterogeneous features. Moreover, high-dimensional datasets prove challenging as well. On average, they yield higher discriminative scores, context-FID scores and correlational scores than low-dimensional datasets. Through our ablations, we identify the VAE component of the LDM as the primary bottleneck. We provide empirical evidence that the VAE fails to adequately enforce a standard Gaussian distribution on the latent space of datasets with heterogeneous features. Using this evidence, we demonstrate that divergence from the Gaussian prior correlates with higher discriminative scores for the synthetic data. The VAE also presents a roadblock to sampling-time watermarking. Currently, decoder inversion is too inexact to effectively retrieve the diffusion latent space, leading to poor watermark detectability. Standard diffusion models operating directly on data proved more reliable for both synthesis quality and watermark performance. Hence, LDMs remain impractical until more capable autoencoders are developed for time series data. Future work could also explore different loss functions to improve the decoder inversion process.

After establishing standard diffusion models as the most suitable candidate for watermarking, we proceed to a closer examination of the watermarks themselves. Here we focus on the following question: How robust are diffusion watermarks against sophisticated post-processing attacks? We introduced a comprehensive attack framework covering time, frequency, and joint time-frequency domains. Using this framework, we evaluated three watermarking methods: TimeWak, TreeRing, and Gaussian Shading. The watermarking robustness evaluation demonstrates TimeWak's advantages over competing methods. TreeRing sacrifices robustness for quality, whereas Gaussian Shading sacrifices quality for robustness. Unlike these methods, TimeWak achieves strong performance across all three dimensions

of the quality-detectability-robustness trade-off. However, our novel frequency-domain attacks expose significant vulnerabilities in all evaluated watermarks, with DCT and Fourier transformations proving particularly effective at degrading or removing watermarks. These results emphasize that watermark evaluation must include diverse attack types to attain a complete picture of a watermark's robustness. Future research could focus on developing watermarks explicitly designed to withstand transformations across both time and frequency domains.

Through systematic evaluation across models, architectures, and attack scenarios, this thesis advances our understanding of watermarking time series. We demonstrate that standard diffusion models outperform both transformer-based models and LDMs in generating high quality synthetic time series. Our novel multi-domain attack suite reveals vulnerabilities overlooked by existing evaluations. These findings highlight that watermark design must consider attack diversity beyond simple time-domain perturbations. As synthetic data becomes increasingly prevalent, robust watermarking grows more critical to prevent misuse. The robustness framework and findings presented here provide a foundation for developing stronger, more resilient watermarking techniques.

References

[1] Yihao Ang et al. “Tsgbench: Time series generation benchmark”. In: *arXiv preprint arXiv:2309.03755* (2023).

[2] Abdul Fatir Ansari et al. “Chronos: Learning the language of time series”. In: *arXiv preprint arXiv:2403.07815* (2024).

[3] John Asafu-Adjaye. “The relationship between energy consumption, energy prices and economic growth: time series evidence from Asian developing countries”. In: *Energy economics* 22.6 (2000), pp. 615–625.

[4] Andreas Blattmann et al. “Align your latents: High-resolution video synthesis with latent diffusion models”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2023, pp. 22563–22575.

[5] George EP Box and David A Pierce. “Distribution of residual autocorrelations in autoregressive-integrated moving average time series models”. In: *Journal of the American statistical Association* 65.332 (1970), pp. 1509–1526.

[6] Karla L Caballero Barajas and Ram Akella. “Dynamically modeling patient’s health state from electronic medical records: a time series approach”. In: *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*. 2015, pp. 69–78.

[7] Ngai Hang Chan. *Time series: applications to finance*. John Wiley & Sons, 2004.

[8] Xinyun Chen et al. “Refit: a unified watermark removal framework for deep learning systems with limited data”. In: *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security*. 2021, pp. 321–335.

[9] Yunzhuo Chen et al. *Image Watermarking of Generative Diffusion Models*. 2025. arXiv: 2502.10465 [eess.IV]. URL: https://arxiv.org/abs/2502.10465.

[10] Jui-Sheng Chou and Duc-Son Tran. “Forecasting energy consumption time series using machine learning techniques based on usage patterns of residential householders”. In: *Energy* 165 (2018), pp. 709–726.

[11] Hai Ci et al. “Ringid: Rethinking tree-ring watermarking for enhanced multi-key identification”. In: *European Conference on Computer Vision*. Springer. 2025, pp. 338–354.

[12] William T Cochran et al. “What is the fast Fourier transform?” In: *Proceedings of the IEEE* 55.10 (1967), pp. 1664–1674.

[13] Benjamin F Crabtree et al. “The individual over time: time series applications in health care research”. In: *Journal of clinical epidemiology* 43.3 (1990), pp. 241–260.

[14] Abhyuday Desai et al. “Timevae: A variational auto-encoder for multivariate time series generation”. In: *arXiv preprint arXiv:2111.08095* (2021).

[15] Hassan Ismail Fawaz et al. “Data augmentation using synthetic data for time series classification with deep residual networks”. In: *arXiv preprint arXiv:1808.02455* (2018).

[16] Jingkun Gao et al. “Robusttad: Robust time series anomaly detection via decomposition and convolutional neural networks”. In: *arXiv preprint arXiv:2002.09545* (2020).

[17] Zijun Gao, Haibao Liu, and Lingbo Li. “Data augmentation for time-series classification: An extensive empirical study and comprehensive survey”. In: *arXiv preprint arXiv:2310.10060* (2023).

[18] Azul Garza and Max Mergenthaler-Canseco. “TimeGPT-1”. In: *arXiv preprint arXiv:2310.03589* (2023).

[19] Luigi Genovese et al. “Daubechies wavelets as a basis set for density functional pseudopotential calculations”. In: *The Journal of Chemical Physics* 129.1 (July 2008). ISSN: 1089-7690. DOI: 10.1063/1.2949547. URL: http://dx.doi.org/10.1063/1.2949547.

[20] Ian Goodfellow et al. “Generative adversarial networks”. In: *Communications of the ACM* 63.11 (2020), pp. 139–144.

[21] Hengzhi He et al. *Watermarking Generative Tabular Data*. 2024. arXiv: 2405.14018 [cs.CR]. URL: https://arxiv.org/abs/2405.14018.

[22] Christopher E Heil and David F Walnut. “Continuous and discrete wavelet transforms”. In: *SIAM review* 31.4 (1989), pp. 628–666.

[23] Jonathan Ho, Ajay Jain, and Pieter Abbeel. “Denoising diffusion probabilistic models”. In: *Advances in neural information processing systems* 33 (2020), pp. 6840–6851.

[24] Jonathan Ho et al. “Cascaded diffusion models for high fidelity image generation”. In: *Journal of Machine Learning Research* 23.47 (2022), pp. 1–33.

[25] Fatih Onur Hocaoglu and Fatih Karanfil. “A time series-based approach for renewable energy modeling”. In: *Renewable and Sustainable Energy Reviews* 28 (2013), pp. 204–214.

[26] Seongmin Hong et al. *On Exact Inversion of DPM-Solvers*. 2023. arXiv: 2311.18387 [cs.CV]. URL: https://arxiv.org/abs/2311.18387.

[27] Paul Jeha et al. “PSA-GAN: Progressive self attention GANs for synthetic time series”. In: *The Tenth International Conference on Learning Representations*. 2022.

[28] Young Shin Kim et al. “Time series analysis for financial market meltdowns”. In: *Journal of Banking & Finance* 35.8 (2011), pp. 1879–1891.

[29] Diederik P Kingma and Max Welling. “Auto-encoding variational bayes”. In: *arXiv preprint arXiv:1312.6114* (2013).

[30] John Kirchenbauer et al. “A watermark for large language models”. In: *International Conference on Machine Learning*. PMLR. 2023, pp. 17061–17084.

[31] Gebhard Kirchgässner, Jürgen Wolters, and Uwe Hassler. *Introduction to modern time series analysis*. Springer Science & Business Media, 2012.

[32] Rohith Kuditipudi et al. “Robust distortion-free watermarks for language models”. In: *arXiv preprint arXiv:2307.15593* (2023).

[33] Solomon Kullback. *Information theory and statistics*. Courier Corporation, 1997.

[34] Sang Gyu Kwak and Jong Hae Kim. “Central limit theorem: the cornerstone of modern statistics”. In: *Korean journal of anesthesiology* 70.2 (2017), p. 144.

[35] Klaus Lehnertz. “Non-linear time series analysis of intracranial EEG recordings in patients with epilepsy—an overview”. In: *International Journal of Psychophysiology* 34.1 (1999), pp. 45–52.

[36] Liangqi Lei et al. “Diffusetrace: A transparent and flexible watermarking scheme for latent diffusion model”. In: *arXiv preprint arXiv:2405.02696* (2024).

[37] Linyang Li et al. *Watermarking LLMs with Weight Quantization*. 2023. arXiv: 2310.11237 [cs.CL]. URL: https://arxiv.org/abs/2310.11237.

[38] Yuxuan Liang et al. “Foundation models for time series analysis: A tutorial and survey”. In: *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2024, pp. 6555–6565.

[39] Shujian Liao et al. “Conditional sig-wasserstein gans for time series generation”. In: *arXiv preprint arXiv:2006.05421* (2020).

[40] Lequan Lin et al. “Diffusion models for time-series applications: a survey”. In: *Frontiers of Information Technology & Electronic Engineering* 25.1 (2024), pp. 19–41.

[41] Haomiao Ni et al. “Conditional image-to-video generation with latent flow diffusion models”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2023, pp. 18444–18455.

[42] Alex Nichol and Prafulla Dhariwal. *Improved Denoising Diffusion Probabilistic Models*. 2021. arXiv: 2102.09672 [cs.LG]. URL: https://arxiv.org/abs/2102.09672.

[43] Yuqi Nie et al. *A Time Series is Worth 64 Words: Long-term Forecasting with Transformers*. 2023. arXiv: 2211.14730 [cs.LG]. URL: https://arxiv.org/abs/2211.14730.

[44] Alexander Nikitin, Letizia Iannucci, and Samuel Kaski. “TSGM: A Flexible Framework for Generative Modeling of Synthetic Time Series”. In: *arXiv preprint arXiv:2305.11567* (2023).

[45] Vasileios C Pezoulas et al. “Synthetic data generation methods in healthcare: A review on open-source tools and methods”. In: *Computational and structural biotechnology journal* 23 (2024), pp. 2892–2910.

[46] Walter HL Pinaya et al. “Brain imaging generation with latent diffusion models”. In: *MICCAI workshop on deep generative models*. Springer. 2022, pp. 117–126.

[47] Walter HL Pinaya et al. “Brain imaging generation with latent diffusion models”. In: *MICCAI workshop on deep generative models*. Springer. 2022, pp. 117–126.

[48] Dustin Podell et al. “Sdxl: Improving latent diffusion models for high-resolution image synthesis”. In: *arXiv preprint arXiv:2307.01952* (2023).

[49] Jian Qian et al. “Timeldm: Latent diffusion model for unconditional time series generation”. In: *arXiv preprint arXiv:2407.04211* (2024).

[50] Alec Radford et al. “Improving language understanding by generative pre-training”. In: (2018).

[51] Kashif Rasul et al. *Autoregressive Denoising Diffusion Models for Multivariate Probabilistic Time Series Forecasting*. 2021. arXiv: 2101.12072 [cs.LG]. URL: https://arxiv.org/abs/2101.12072.

[52] *Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation)*. Apr. 27, 2016. URL: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32016R0679 (visited on 02/29/2024).

[53] Ahmad Rezaei et al. “Lawa: Using latent space for in-generation image watermarking”. In: *arXiv preprint arXiv:2408.05868* (2024).

[54] Robin Rombach et al. *High-Resolution Image Synthesis with Latent Diffusion Models*. 2022. arXiv: 2112.10752 [cs.CV]. URL: https://arxiv.org/abs/2112.10752.

[55] Zhi Wen Soi et al. *TimeWak: Temporal Chained-Hashing Watermark for Time Series Data*. 2025. arXiv: 2506.06407 [cs.CR]. URL: https://arxiv.org/abs/2506.06407.

[56] Jiaming Song, Chenlin Meng, and Stefano Ermon. *Denoising Diffusion Implicit Models*. 2022. arXiv: 2010.02502 [cs.LG]. URL: https://arxiv.org/abs/2010.02502.

[57] Yang Song et al. *Score-Based Generative Modeling through Stochastic Differential Equations*. 2021. arXiv: 2011.13456 [cs.LG]. URL: https://arxiv.org/abs/2011.13456.

[58] Gilbert Strang. “The discrete cosine transform”. In: *SIAM review* 41.1 (1999), pp. 135–147.

[59] Zbigniew R. Struzik and Arno Siebes. “The Haar Wavelet Transform in the Time Series Similarity Paradigm”. In: *Proceedings of the Third European Conference on Principles of Data Mining and Knowledge Discovery*. PKDD ’99. Berlin, Heidelberg: Springer-Verlag, 1999, pp. 12–22. ISBN: 3540664904.

[60] Namjoon Suh et al. “TimeAutoDiff: Combining Autoencoder and Diffusion model for time series tabular data synthesizing”. In: *arXiv preprint arXiv:2406.16028* (2024).

[61] Tomonori Takahashi and Takayuki Mizuno. “Generation of synthetic financial time series by diffusion models”. In: *Quantitative Finance* (2025), pp. 1–10.

[62] Paul W Talbot et al. “Correlated synthetic time series generation for energy system simulations using Fourier and ARMA signal processing”. In: *International Journal of Energy Research* 44.10 (2020), pp. 8144–8155.

[63] Ruey S Tsay. *Analysis of financial time series*. John wiley & sons, 2005.

[64] Terry T Um et al. “Data augmentation of wearable sensor data for parkinson’s disease monitoring using convolutional neural networks”. In: *Proceedings of the 19th ACM international conference on multimodal interaction*. 2017, pp. 216–220.

[65] Ashish Vaswani et al. “Attention is all you need”. In: *Advances in neural information processing systems* 30 (2017).

[66] Qingsong Wen et al. “Time series data augmentation for deep learning: A survey”. In: *arXiv preprint arXiv:2002.12478* (2020).

[67] Yuxin Wen et al. “Tree-ring watermarks: Fingerprints for diffusion images that are invisible and robust”. In: *arXiv preprint arXiv:2305.20030* (2023).

[68] Gerald Woo et al. “Unified Training of Universal Time Series Forecasting Transformers”. In: *arXiv preprint arXiv:2402.02592* (2024).

[69] Haixu Wu et al. *Autoformer: Decomposition Transformers with Auto-Correlation for Long-Term Series Forecasting*. 2022. arXiv: 2106.13008 [cs.LG]. URL: https://arxiv.org/abs/2106.13008.

[70] Zijin Yang et al. “Gaussian Shading: Provable Performance-Lossless Image Watermarking for Diffusion Models”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024, pp. 12162–12171.

[71] Gokul Yenduri et al. *Generative Pre-trained Transformer: A Comprehensive Review on Enabling Technologies, Potential Applications, Emerging Challenges, and Future Directions*. 2023. arXiv: 2305.10435 [cs.CL]. URL: https://arxiv.org/abs/2305.10435.

[72] Jinsung Yoon, Daniel Jarrett, and Mihaela Van der Schaar. “Time-series generative adversarial networks”. In: *Advances in neural information processing systems* 32 (2019).

[73] Xinyu Yuan and Yan Qiao. “Diffusion-ts: Interpretable diffusion for general time series generation”. In: *arXiv preprint arXiv:2403.01742* (2024).

[74] Guoqiang Zhang, Jonathan P Lewis, and W Bastiaan Kleijn. “Exact diffusion inversion via bidirectional integration approximation”. In: *European Conference on Computer Vision*. Springer. 2024, pp. 19–36.

[75] Jie Zhang et al. *Deep Model Intellectual Property Protection via Deep Watermarking*. 2021. arXiv: 2103.04980 [cs.CV]. URL: https://arxiv.org/abs/2103.04980.

[76] Lijun Zhang et al. “Attack-Resilient Image Watermarking Using Stable Diffusion”. In: *The Thirty-eighth Annual Conference on Neural Information Processing Systems*. 2024.

[77] Xinyang Zhang et al. “Text2layer: Layered image generation using latent diffusion model”. In: *arXiv preprint arXiv:2307.09781* (2023).

[78] Kai Zhao et al. “Dominant Shuffle: A Simple Yet Powerful Data Augmentation for Time-series Prediction”. In: *arXiv preprint arXiv:2405.16456* (2024).

[79] Shihao Zhao et al. “Uni-controlnet: All-in-one control to text-to-image diffusion models”. In: *Advances in Neural Information Processing Systems* 36 (2023), pp. 11127–11150.

[80] Daquan Zhou et al. “Magicvideo: Efficient video generation with latent diffusion models”. In: *arXiv preprint arXiv:2211.11018* (2022).