Training Implicit Generative Models via an Invariant Statistical Loss

José Manuel de Frutos Universidad Carlos III
Pablo M. Olmos Universidad Carlos III
Manuel A. Vázquez Universidad Carlos III
Joaquín Míguez Universidad Carlos III

Abstract

Implicit generative models have the capability to learn arbitrary complex data distributions. On the downside, training requires telling apart real data from artificially-generated ones using adversarial discriminators, leading to unstable training and mode-dropping issues. As reported by Zahee et al. (2017), even in the one-dimensional (1D) case, training a generative adversarial network (GAN) is challenging and often suboptimal. In this work, we develop a discriminator-free method for training one-dimensional (1D) generative implicit models and subsequently expand this method to accommodate multivariate cases. Our loss function is a discrepancy measure between a suitably chosen transformation of the model samples and a uniform distribution; hence, it is invariant with respect to the true distribution of the data. We first formulate our method for 1D random variables, providing an effective solution for approximate reparameterization of arbitrary complex distributions. Then, we consider the temporal setting (both univariate and multivariate), in which we model the conditional distribution of each sample given the history of the process. We demonstrate through numerical simulations that this new method yields promising results, successfully learning true distributions in a variety of scenarios and mitigating some of the well-known problems that state-of-the-art implicit methods present.

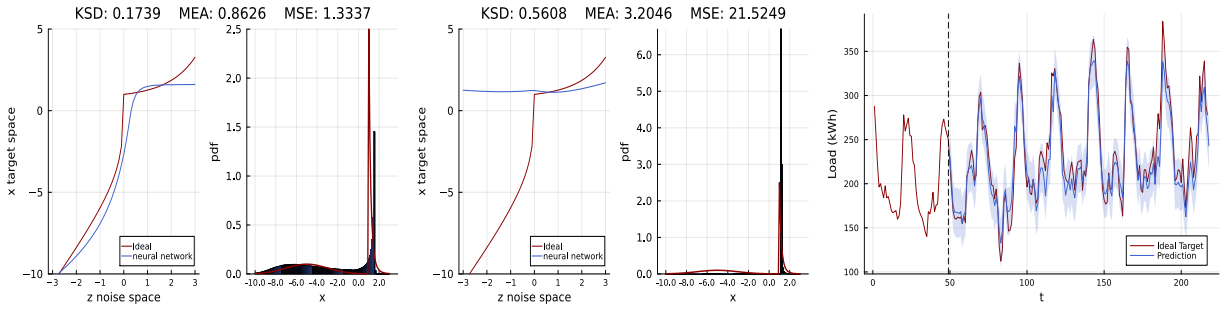
Proceedings of the 27[th] International Conference on Artificial Intelligence and Statistics (AISTATS) 2024, Valencia, Spain. PMLR: Volume 238.

1 Introduction

Implicit generative models employ an m-dimensional latent random variable (r.v.) $\mathbf{z}$ to simulate random samples from a prescribed n-dimensional target probability distribution. To be precise, the latent variable undergoes a transformation through a deterministic function g_θ, which maps $\mathbb{R}^m \mapsto \mathbb{R}^n$ using the parameter set θ. Given the model capability to generate samples with ease, various techniques can be employed for contrasting two sample collections: one originating from the genuine data distribution and the other from the model distribution. This approach essentially constitutes a methodology for the approximation of probability distributions via comparison. Generative adversarial networks (GANs) [Goodfellow et al., 2014], f-GANs [Nowozin et al., 2016], Wasserstein-GANs (WGANs) [Arjovsky et al., 2017], adversarial variational Bayes (AVB) [Mescheder et al., 2017], and maximum mean-miscrepancy (MMD) GANs [Li et al., 2017] are some popular methods that fall within this framework.

Approximation of 1-dimensional (1D) parametric distributions is a seemingly naive problem for which the above-mentioned models can perform below expectations. In [Zaheer et al., 2017], the authors report that various types of GANs struggle to approximate relatively simple distributions from samples, emerging with MMD-GAN as the most promising technique. However, the latter implements a kernelized extension of a moment-matching criterion defined over a reproducing kernel Hilbert space, and consequently, the objective function is expensive to compute.

In this work, we introduce a novel approach to train univariate implicit models that relies on a fundamental property of rank statistics. Let $r_1 < r_2 < \cdots < r_k$ be a ranked (ordered) sequence of independent and identically distributed (i.i.d.) samples from the generative model with probability density function (pdf) $\tilde{p}$, and let y be a random sample from a pdf p. If $\tilde{p} = p$, then $\mathbb{P}(r_{i-1} \leq y < r_i) = \frac{1}{K}$ for every $i = 1, \ldots, K+1$, with the convention that $r_0 = -\infty$ and $r_{K+1} = \infty$ (see, e.g., [Rosenblatt, 1952] or [Elvira et al., 2016a]



(a) ISL estimation (b) MMD-GAN estimation (c) 7-Day Electricity-C ISL Forecast

Figure 1: In (a), we show the ISL estimation of the pdf of a 1D random variable with a true distribution that is a mixture model with two components: a Gaussian $\mathcal{N}(-5, 2)$ distribution and a Pareto(5,1) distribution. In (b), we show the performance of MMD-GAN in [Zaheer et al., 2017] for the same target. In the left-side of both figures, we compare the optimal transformation from a $N(0, 1)$ to the target distribution versus the generator learned by ISL (a) and MMD-GAN (b). In (c), we show the temporal ISL-based 7-Day Forecast in the Electricity-C dataset [Trindade, 2015].

for a short explicit proof). This invariant property holds for any continuously distributed data, i.e., for any data with a pdf p. Consequently, even if p is unknown, we can leverage this invariance to construct an objective (loss) function. This objective function eliminates the need for a discriminator, directly measuring the discrepancy of the transformed samples with respect to (w.r.t.) the uniform distribution. The computational cost of evaluating this loss increases linearly with both K and N, allowing for low-complexity mini-batch updates. Moreover, the proposed criterion is invariant across true data distributions, hence we refer to the resulting objective function as invariant statistical loss (ISL). Because of this property, the ISL can be exploited to learn multiple modes in mixture models and different time steps when learning temporal processes. Additionally, considering the marginal distributions independently, it is straightforward to extend ISL to the multivariate case. In both scenarios, as illustrated in Figure 1, our approach enhances training robustness by avoiding the min-max optimization problem and outperforms GANs, including MMD-GAN and WGAN, and state-of-the-art deep temporal generative models.

2 Preliminaries

In this section we formally state the problem of density estimation, and introduce the main ideas that led to the design of the proposed loss function.

2.1 Problem statement

Let $y_1, \dots, y_N$ be real i.i.d. samples from a probability distribution with pdf p, let z be a random sample from a canonical distribution with pdf p_0 (e.g., standard Gaussian) and let $g_\theta : \mathbb{R} \mapsto \mathbb{R}$ be a real map parametrized by a $\theta \in \mathbb{R}^l$. The aim is to use the data $y_1, \dots, y_N$ to learn the parameters θ such that $\tilde{y} = g_\theta(z)$ is distributed according to the pdf of the data, p.

2.2 Implicit generative models

Implicit generative models train a *generator* neural network (NN), parametrized by θ and represented as g_θ, to transform samples $z \sim p_0$ into $\tilde{y} = g_\theta(z)$ with pdf $\tilde{p}$ such that, ideally, $p \approx \tilde{p}$. Throughout the training process, the emphasis is placed on computing the discrepancy, d, between p and $\tilde{p}$, which becomes the loss function of an optimization problem. This discrepancy reaches zero if and only if $p = \tilde{p}$. In this paper, we design a loss function d that measures the discrepancy between p and $\tilde{p}$, drawing upon the existence of a statistic that is uniform whenever the densities are equal. In particular, the proposed statistic is invariant w.r.t. p and it is based on the comparison between a set of i.i.d samples generated by our model and the real data.

3 Uniform rank statistics

In Section 3.1 we construct a simple rank statistic that can be proved to be uniform for any i.i.d. random sample from a continuous distribution with pdf p. Then, in Section 3.2 we prove that one can use data from an unknown pdf p to construct rank statistics that are approximately uniform when $\tilde{p} \approx p$ (and converge to uniform r.v.'s when $\tilde{p} \to p$, where $\tilde{p}$ is the pdf of the generative model).

Finally, in Section 3.3 we prove that when the proposed rank statistics are uniformly distributed then $p = \tilde{p}$ almost everywhere.

3.1 Discrete uniform rank statistics

Let $\tilde{y}_1, \ldots, \tilde{y}_K$ be a random sample from a univariate real distribution with pdf $\tilde{p}$ and let y be a single random sample independently drawn from another distribution with pdf p. We construct the set,

$$\mathcal{A}_K := \left\{ \tilde{y} \in \{\tilde{y}_k\}_{k=1}^K : \tilde{y} < y \right\},$$

and the statistic $A_K := |\mathcal{A}_K|$, i.e., A_K is the number of elements of $\mathcal{A}_K$. We note that A_K is a rank statistic that can be alternatively constructed by first finding indices $j_1, \ldots, j_k$ that sort the sample $\tilde{y}_1, \ldots, \tilde{y}_K$ in ascending order,

$$\tilde{y}_{j_1} < \tilde{y}_{j_2} < \ldots < \tilde{y}_{j_K},$$

and then setting

$$A_K = \sup \left\{ i \in \{1, \ldots, K\} : \tilde{y}_{j_i} < y \right\}. \tag{1}$$

Also note that almost surely $y \neq \tilde{y}_i$ for all i. The statistic A_K is a discrete r.v. that takes values on the set $\{0, \ldots, K\}$ and we denote its probability mass function (pmf) as $\mathbb{Q}_K : \{0, \ldots, K\} \mapsto [0, 1]$. This pmf satisfies the following fundamental result.

Theorem 1. *If $p = \tilde{p}$ then $\mathbb{Q}_K(n) = \frac{1}{K+1}$ $\forall n \in \{0, \ldots, K\}$, i.e., A_K is a discrete uniform r.v. on the set $\{0, \ldots, K\}$.*

Proof. See [Elvira et al., 2021] for an explicit proof. This is a basic result that appears under different forms in the literature, e.g., in [Rosenblatt, 1952] or [Djuric and Míguez, 2010]. □

3.2 Approximately uniform statistics

For $f : \mathbb{R} \mapsto \mathbb{R}$ a real function and p the pdf of an univariate real r.v. let us denote

$$(f, p) := \int_{-\infty}^{\infty} f(x) p(x) dx$$

and let $B_1(\mathbb{R}) := \left\{ (f : \mathbb{R} \to \mathbb{R}) : \sup_{x \in \mathbb{R}} |f(x)| \leq 1 \right\}$ be the set of real functions bounded by 1.

The quantity $\sup_{f \in B_1(\mathbb{R})} |(f, p) - (f, \tilde{p})|$ can be directly related to the total variation distance between the distributions with pdf's p and $\tilde{p}$. Hence, the following theorem states that if the generator output pdf, $\tilde{p}$, is close to p then the rank statistic A_K with pmf $\mathbb{Q}_K(n)$ constructed in Section 3.1 is approximately uniform.

Theorem 2. *If $\sup_{f \in B_1(\mathbb{R})} |(f, p) - (f, \tilde{p})| \leq \epsilon$ then,*

$$\frac{1}{K+1} - \epsilon \leq \mathbb{Q}_K(n) \leq \frac{1}{K+1} + \epsilon, \ \forall n \in \{0, \ldots, K\}.$$

The proof of this Theorem can be found in the Appendix.

3.3 A converse theorem

So far we have shown that if the pdf of the generative model, $\tilde{p}$, is close to the target pdf, p, then the statistic A_K is close to uniform. A natural question to ask is whether A_K displaying a uniform distribution implies that $\tilde{p} = p$. In this section, we prove that if A_K is uniform, i.e., $\mathbb{Q}_K(n) = \frac{1}{K+1}$, for all $n \in \{0, \ldots, K\}$ and all K, then $p = \tilde{p}$ almost everywhere.

To this end, we recall a key result from [Elvira et al., 2021].

Lemma 1. *Let p and $\tilde{p}$ be two univariate pdf's with associated cfd's F and $\tilde{F}$. If the rank statistic A_K constructed in Section 3 has a uniform distribution on the set $\{0, \ldots, K\}$ then $(\tilde{F}^n, p) = \frac{1}{n+1}$, $\forall n \in \{0, 1, \ldots, K\}$, where $\tilde{F}^n$ is the n-th power of $\tilde{F}$.*

Proof. See [Elvira et al., 2021, Theorem 2]. □

Based on Lemma 1 we can prove the result anticipated at the beginning of this section.

Theorem 3. *Let p and $\tilde{p}$ be pdf's of univariate real r.v.'s and let A_K be the rank statistic constructed in Section 3.1. If A_K has a discrete uniform distribution on $\{0, \ldots, K\}$ for every $K \in \mathbb{N}$, then $p = \tilde{p}$ almost everywhere.*

Proof. Let Y be a r.v. with pdf p and cdf F. From the inversion theorem (see, e.g., [Martino et al., 2018, Theorem 2.1]) we obtain that $U = F(Y)$ is a standard uniform r.v., i.e., $U \sim \mathcal{U}(0, 1)$ and as consequence,

$$\mathbb{E}[U^n] = (F^n, p) = \frac{1}{n+1}, \quad \forall n \in \mathbb{N}.$$

On the other hand, since A_K is uniformly distributed on $\{0, \ldots, K\}$, Lemma 1 implies that the r.v. $\tilde{U} = \tilde{F}(Y)$ satisfies

$$\mathbb{E}[\tilde{U}^n] = (\tilde{F}^n, p) = \frac{1}{n+1}, \quad \forall n \in \mathbb{N},$$

hence,

$$\tilde{U} = \tilde{F}(Y) \sim \mathcal{U}(0, 1) \tag{2}$$

as well. However, the relationship in Eq. (2) implies that $Y = \tilde{F}^{-1}(\tilde{U})$ has cdf $\tilde{F}$ (from the inversion theorem again), hence $F = \tilde{F}$, which implies that $p = \tilde{p}$ almost everywhere. □

4 The invariant statistical loss function

Taken together, Theorems 1, 2 and 3 imply that if we train a generative model with output pdf $\tilde{p}$ to make

the random statistics A_K uniform for sufficiently large K, then we can expect that $\tilde{p} \approx p$, where p is the actual pdf of the available data that has been used for training. This is true independently of the form of the pdf p and, in this sense, it enables us to construct loss functions which are themselves invariant w.r.t. p. In this section we introduce one such invariant loss and then propose a suitable surrogate which is differentiable w.r.t. the generative model parameters θ and, therefore, can be used for training using standard optimization tools.

4.1 The Invariant Statistic Loss

The training data set consists of a set of N i.i.d. samples $y_1, \dots, y_N$ from the true data distribution p. For each y_n, along with K i.i.d. samples from the generative model $\tilde{\mathbf{y}} = [\tilde{y}_1, \dots, \tilde{y}_K]^\top$, where $\tilde{y}_i = g_\theta(z_i)$, we can obtain one sample of the r.v. A_K, that we denote as $a_{K,n}$.

The great advantage of our method is that we simply have to verify the discrepancy between the discrete uniform distribution with $K+1$ outcomes and the empirical distribution of the set of statistics $a_{K,1}, a_{K,2}, \dots, a_{K,N}$, i.e., their associated histogram. On the downside, note that, by definition, the construction of the statistic A_K in Eq. (1) from samples of the generator does not yield a function that is differentiable w.r.t. its parameters θ.

We now introduce a new loss function, coined *invariant statistical loss* (ISL) that can be optimized w.r.t. θ and mimics the construction of a histogram from the statistics $a_{K,1}, a_{K,2}, \dots, a_{K,N}$. For this purpose, given a real data point y, we can tally how many of the K simulated samples in $\tilde{\mathbf{y}}$ are less than the observation y. Specifically, one computes

$$\tilde{a}_K(y) = \sum_{i=1}^{K} \sigma_\alpha(y - \tilde{y}_i) = \sum_{i=1}^{K} \sigma_\alpha(y - g_\theta(z_i)),$$

where z_i is a sample from a univariate standard Gaussian distribution, i.e., $z_i \sim p_0 \equiv \mathcal{N}(0,1)$, $\sigma(x) := 1/\left(1+\exp(-x)\right)$ is the sigmoid function and $\sigma_\alpha(x) := \sigma(\alpha x)$. We remark that $\tilde{a}_K(y)$ is a differentiable (w.r.t. θ) approximation of the actual statistic A_K for the observation y. The parameter α enables us to adjust the slope of the sigmoid function to better approximate the (discrete) 'counting' in the construction of A_K (other alternatives can certainly be chosen here). Sharper functions offer better approximation but unstable gradients. In our case, $\alpha \in [10, 20]$ has been effective in practice.

To construct a differentiable surrogate histogram from $\tilde{a}_K(y_1), \dots, \tilde{a}_K(y_N)$, we leverage a sequence of differentiable functions designed to mimic the bins around $k \in \{0, \dots, K\}$, replacing sharp bin edges with functions that mirror bin values at k and smoothly decay outside a neighborhood of k. In our particular case, we consider radial basis function (RBF) kernels $\{\psi_k\}_{k=0}^K$ centered at $k \in \{0, \dots, K\}$ with length-scale ν^2, i.e., $\psi_k(a) = \exp(-(a-k)^2/2\nu^2)$. Thus, the approximate normalized histogram count at bin k is given by

$$q[k] = \frac{1}{N} \sum_{i=1}^{N} \psi_k(\tilde{a}_K(y_i)), \tag{3}$$

for $k = 0, \dots, K$. The proposed ISL is now computed as ℓ-norm distance between the uniform pmf $\frac{1}{K+1}\mathbf{1}_{K+1}$ and the vector of empirical probabilities $\mathbf{q} = [q[0], q[1], \dots, q[K]]^\top$, namely,

$$\mathcal{L}_{ISL}(\theta, K, \alpha, \nu) = \left|\left| \frac{1}{K+1}\mathbf{1}_{K+1} - \mathbf{q} \right|\right|_\ell. \tag{4}$$

In Figure 2, we compare the surrogate ISL loss function and the theoretical counterpart for an exemplary case. The set of hyperparameters is listed in Table 1. Note that evaluating (4) requires $\mathcal{O}(NK)$ operations, but we can rely on mini-batch optimization to reduce it to $\mathcal{O}(MK)$, being M the mini-batch size. Algorithm 1 summarizes the training method.

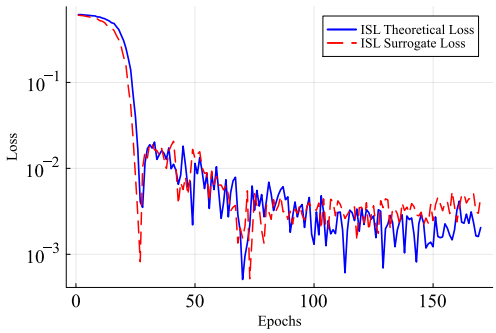


Figure 2: ISL surrogate loss function vs. ISL theoretical loss during the training of $\mathcal{N}(4,2)$, $K = 1000$, $N = 1000$, log-scale.

Table 1: ISL list of hyperparameters.

Hyperparameter	Value
Nº samples, K	tunable
Activation function	$\sigma(\alpha \cdot x)$
$\{\psi\}_{k=0}^K$	RBF Kernels, length-scale=ν^2

4.2 Progressively Training by Increasing K

The parameter K dictates the number of simulated samples per data point. As established in Section 3, larger values of K ensure a better approximation of

Algorithm 1 ISL Algorithm

Input Neural Network g_θ; Hyperparameter K, N; Number of Epochs; Batch Size M; Training Data $\{y_i\}_{i=1}^N$.
Output Trained Neural Network g_θ.
For $t = 1, \dots,$ epochs **do**
  For iteration $= 1, \dots, N/M$ **do**
    Select M samples from $\{y_j\}_{j=1}^N$ at random
    $\{z_i\}_{i=1}^K \sim \mathcal{N}(0,1)$
    $\mathbf{q} = \frac{1}{M}\sum_{j=1}^M \psi_k\Big(\sum_{i=1}^K \sigma_\alpha(y_j - g_\theta(z_i))\Big)$
    $\nabla_\theta \text{ loss} \leftarrow \nabla_\theta \left|\left|\frac{1}{K+1}\mathbf{1}_{K+1} - \mathbf{q}\right|\right|_2$
    Backpropagation$(g_\theta, \nabla_\theta \text{ loss})$
return g_θ

the true data distribution, albeit at the expense of an increased computational effort.

We have found that the training procedure can be made more efficient if it is performed in a sequence of increasing values of K, say $K^{(1)} < K^{(2)} < \cdots < K^{(I)}$, where I is the total number of stages in the process and $K^{(I)} = K_{max}$ is the maximum admissible value of K –a hyperparameter to be chosen depending on the available computing power. Hence, at the i-th stage we train the generative model using the ISL $\mathcal{L}_{ISL}(\theta, K^{(i)})$. During training, we periodically run a test (e.g., a Pearson χ^2 test against the discrete uniform distribution using the statistics $a_{K,1}, a_{K,2}, \dots, a_{K,N}$). When the hypothesis "$A_K$ is uniform" is accepted we increase $K^{(i)}$ to $K^{(i+1)}$ for the $(i+1)$-th stage and keep training. Note that the generative model parameters adjusted at the i-th stage serve as an initial condition for training at the $(i+1)$-th stage. We have followed this methodology for all the experiments presented in this paper.

5 Invariant Statistical Loss for Time Series Prediction

So far we have focused our discussion on 1D random variables, but our approach is amenable to be used on (scalar and multivariate) time series. Let $y[0], y[1], \dots, y[T]$ be a sample of a discrete-time random process between time[1] 0 and time T, and let $Y[t]$ be the random variable of the process at time t, with unknown conditional distribution $p_t = p(Y[t]|Y[0], Y[1], \cdots, Y[t-1])$.

Given the sequence $y[0], y[1], \dots, y[t-1]$, our goal is to train an autorregressive conditional implicit generator network $g_\theta(z, \mathbf{h}[t])$ that produces samples whose distribution, referred to as $\tilde{p}_t$, is close to the unknown pdf p_t when z is a sample drawn from a standard Gaussian. The new input to our generator NN, $\mathbf{h}[t]$, is an embedding of the sequence $y[0], y[1], \dots, y[t-1]$ provided by a suitable neural network. Specifically, we consider a simple recurrent neural network (RNN) connected to the generator NN as illustrated in Figure 3. During training, the newly-arrived observation at time t, $y[t]$, is fed into the RNN to collapse the process history into *hidden* state $\mathbf{h}[t]$. The latter is then exploited (along with input noise z) by generator NN $g_\theta(z, \mathbf{h}[t])$ to predict the next observation $\tilde{y}[t+1]$. During test/validation, forecasting is performed by feeding $\tilde{y}[t+1]$ back to the RNN.

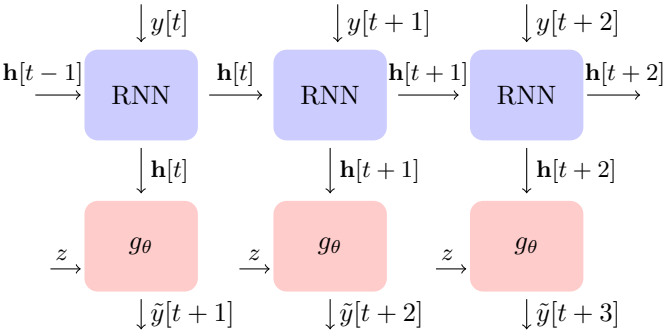


Figure 3: Conditional implicit generative model for time-series prediction.

Notice that in this new temporal setup, all the results established in the previous sections remain valid. However, while before we had a single statistic A_K, we now have one at every time instant, $A_K[t]$. The latter is still constructed according to Eq. (1), and will follow a uniform distribution if $\tilde{p}_t$ is close enough to p_t. This result is invariant w.r.t. t and p_t.

We use the sequence of observations $\mathbf{y} = [y[0], y[1], \dots, y[T]]$ to construct a sequence of statistics $a_K[0], a_K[1], \dots, a_K[T]$, whose aggregated empirical distribution should be approximately uniform if $\tilde{p}_t \approx p_t$ for $t = 0, \dots, T$.

To construct the ISL, we apply a similar procedure to obtain a differentiable histogram surrogate.

When observations are multivariate, i.e., we have a collection of time series[2], $\{\mathbf{y}_i\}_{i=1}^N$, we can essentially apply the same procedure. Specifically, for every time series, $\mathbf{y}_i$, we obtain a sequence of statistics

[1] We assume with no loss of generality that the origin of the process is at time $t = 0$.

[2] Assume for simplicity that they are of the same length T, but this is not a constraint here.

$\mathbf{a}_{i,K} = [a_{i,K}[0], a_{i,K}[1], \ldots, a_{i,K}[T]]$. If $\tilde{p}_t \approx p_t$ for $t = 0, \ldots, T$, then again the agreggated empirical distribution of $\mathbf{a}_{1,K} \bigcup \mathbf{a}_{2,K} \bigcup \ldots \bigcup \mathbf{a}_{T,K}$, should be approximately uniform. The ISL surrogate requires in this general scenario $\mathcal{O}(TNK)$ operations to evaluate, which can be reduced to $\mathcal{O}(WMK)$ if we use mini-batches of M data points and we consider a sliding window of length W.

6 Related Work

Among existing methods for training implicit generative models, most of the works in the literature consider the use of a discriminator/critic network that classifies whether data points are artificially generated or are real samples [Mohamed and Lakshminarayanan, 2016]. From basic GANs [Goodfellow et al., 2014], to WGANs [Arjovsky et al., 2017] and f-GANs [Nowozin et al., 2016], this approach is known to suffer from mode collapse and training instabilities [Arora and Zhang, 2017, Arora et al., 2017]. These issues can be alleviated with techniques such as mini-batch discrimination [Salimans et al., 2016] and spectral normalization [Miyato et al., 2018].

Among training methods for implicit models that do not require a discriminator, Maximum Mean Discrepancy (MMD) losses are a popular choice [Gretton et al., 2012, Li et al., 2015, Gouttes et al., 2021]. Incorporating MMD ensures that the distance between the real and generated data distributions is minimized in a Reproducing Kernel Hilbert Space, leading to more stable training and nuanced density approximations. However, due to the reformulation in terms of kernels, the complexity of evaluating the loss is large (scales quadratically with the amount of data), and approximations are needed.

The literature on implicit temporal generative models is comparatively much shorter since the best-performing methods nowadays are based on prescribed conditional autoregressive models combined with sequential transformer architectures, conditional normalizing flows, or difussion models [Salinas et al., 2020, Li et al., 2019a, Moreno-Pino et al., 2023, Rasul et al., 2021b, Rasul et al., 2021a]. To our knowledge, due to the inherent invariance, ISL is the first method to adapt implicit generative models to the temporal domain in a straightforward manner. As we demonstrate, ISL achieves competitive performance when compared to prescribed autoregressive generative methods such as Phrophet [Taylor and Letham, 2018], TRMF [Yu et al., 2016], Deep State Space Models (DSSM) [Rangapuram et al., 2018], DeepAR [Salinas et al., 2020], and transformer architectures [Zeng et al., 2023, Wu et al., 2021, Zhou et al., 2021, Li et al., 2019a] with simpler network structures in both univariated and multivariated scenarios.

7 Experiments

Our experiments evaluate implicit models trained with the ISL in Eq. (4) for density estimation. We showcase its qualitative advantage over classical generative models for independent time settings, capturing multimodal and heavy-tailed distributions. ISL outperforms vanilla-GAN, Wasserstein-GAN, and MMD-GAN baselines. ISL also shows promise when compared to more modern methods such as diffusion models. In addition, we explore its potential as a GAN regularizer. Our experimental settings for this section are grounded in the study by [Zaheer et al., 2017]. Subsequently, we assess ISL's effectiveness in time series forecasting using synthetic and real datasets.

In the supplementary material, we extend the experimental validation by comparing different NN hyperparameters and several configurations of the proposed experimental settings.

7.1 Learning 1-D distributions

We start considering the same experimental setup as [Zaheer et al., 2017]. In particular, we aim at approximating different target distributions using 1000 i.i.d. samples.

The final three rows describe mixture models, each with equal weighting among their components,

Model_1 consists of a mixture model that combines two normal distributions, $\mathcal{N}(5, 2)$ and $\mathcal{N}(-1, 1)$. Model_2 is a mixture model that integrates three normal distributions $\mathcal{N}(5, 2)$, $\mathcal{N}(-1, 1)$ and $\mathcal{N}(-10, 3)$. Finally, Model_3 combines a normal distribution $\mathcal{N}(-5, 2)$ with a Pareto distribution $Pareto(5, 1)$.

As generator NN, we use a 4-layer multilayer perceptron (MLP) with $7, 13, 7$ and 1 units at the corresponding layers. In the supplementary material we compare the ISL performance with different architectures. As activation function we use a exponential linear unit (ELU). We train each setting up to 10^4 epochs with 10^{-2} learning rate using Adam. We compare ISL, GAN, WGAN, and MMD-GAN using KSD (Kolmogorov-Smirnov Distance), MAE (Mean Absolute Error), and MSE (Mean Squared Error) error metrics, defined as follows

Table 2: Comparison of ISL results with vanilla GAN, WGAN, and MMD-GAN under the hyperparameters: Noise $\sim \mathcal{N}(0,1)$, $K_{max} = 10$, epochs $= 1000$, and $N = 1000$.

	ISL			GAN			WGAN			MMD-GAN		
Target	**KSD**	**MAE**	**MSE**	**KSD**	**MAE**	**MSE**	**KSD**	**MAE**	**MSE**	**KSD**	**MAE**	**MSE**
$\mathcal{N}(4,2)$	$\mathbf{8.5e^{-3}}$	**0.1005**	**0.0348**	0.0154	0.4335	0.4015	0.0468	4.4644	55.1103	0.0238	4.2689	30.3614
$\mathcal{U}(-2,2)$	**0.0125**	0.1341	0.0258	0.0265	**0.1097**	**0.0180**	0.0790	0.5659	0.4025	0.0545	0.5322	0.4130
Cauchy(1, 2)	$\mathbf{6.3e^{-3}}$	**7.4957**	8177	0.0823	8.9628	8207	0.0302	10.5681	**8127**	0.0263	9.6798	57975
Pareto(1, 1)	**0.0822**	**4.7742**	7843	0.0987	12.6397	114970	0.4919	7.6435	7062	0.5008	9.023	10674
Model_1	0.0169	**0.2995**	**0.1437**	**0.0116**	0.4246	0.4537	0.0315	1.8831	4.8943	0.0450	1.6433	4.6988
Model_2	**0.0173**	0.6660	0.7757	0.0536	**0.6072**	**0.7506**	0.0459	4.2390	28.9843	0.0232	8.7218	112
Model_3	**0.1591**	0.8791	**1.5207**	0.1846	**0.4447**	1.6053	0.2986	3.0376	13.1256	0.5608	3.2046	21.5244

Table 3: ISL vs Diffusion model (1D case).

	ISL	Diff.
Target	**KSD**	**KSD**
$\mathcal{N}(4,2)$	**8.5e-3**	0.0867
$\mathcal{U}(-2,2)$	**0.0125**	0.2307
Cauchy(1, 2)	**6.3e-3**	0.0226
Pareto(1, 1)	0.0822	**0.0512**
Model_1	**0.0169**	0.1507
Model_2	**0.0173**	0.2217
Model_3	0.1591	**0.05475**

$$\text{KSD} = \sup_x |F(x) - \tilde{F}(x)|,$$

$$\text{MAE} = \int_{-\infty}^{\infty} |f(z) - f_\theta(z)|\, p_z(z) dz,$$

$$\text{MSE} = \int_{-\infty}^{\infty} \big(f(z) - f_\theta(z)\big)^2 p_z(z) dz,$$

where we have denoted by F, $\tilde{F}$, and F_Z the cdfs associated with the densities p, $\tilde{p}$, and p_Z respectively.

Observe in Table 2 that ISL outperforms the rest of methods in most scenarios, especially in the case of mixture models. Referring to [Zaheer et al., 2017, Theorem 1], it asserts that for a 1D distribution, there are at most two continuous transformations f such that if Z follows p_0 and X follows p, then $f(Z) = X$. These transformations can be derived using the probability integral transform. In Figure 1 we compare the generator function $g_\theta(z)$ with the true transformation function $f(z)$ for ISL (a) and MMD-GAN (b) for Model_3.

In Table 3 we compare ISL vs Diffusion model (1D case). The latter is based on an MLP with three layers of 100 units each, integrated with 100-dimensional positional encodings for time steps. It has been optimized for 20 epochs using MSE as a loss function, and linearly spaced betas (0.0004 to 0.06) across 100 denoising steps. For further information and details about parameters, refer to [Scotta and Messina, 2023].

7.2 GAN pre-training

Our method can be used in combination with a GAN pre-trained generator. While GANs often struggle with multimodal distributions, typically capturing only some modes [Arora et al., 2017, Arora and Zhang, 2017], our approach fully represents the support of the true distribution. If not, disparities would arise between fictional and real data. Especially with a large K, missed modes would cause fictional samples to diverge from real data modes, biasing the generated histogram. In this subsection, we now assess the ability of ISL to correct a potentially biased GAN construction of the generator $g_\theta(z)$.

Pre-training ISL with a GAN could potentially improve results if practitioners notice that ISL is suffering from a vanishing gradient problem due to poor parameter initialization. This strategy has only been used for the experiments in Table 4. For the rest of the 1D density approximations and time series forecasting experiments, no pre-training was applied.

We use the same parameters as before for the generator. For the GAN critics, following [Zaheer et al., 2017], we use a 4-layer MLP with 11, 29, 11, and 1 units. The ratio of updating critics and generators are searched in $\{2:1, 3:1, 4:1, 5:1\}$. We train each setting for 10^4 epochs.

Upon training the GAN, we train the generator using ISL with $K_{max} = 10$, 1000 epochs, and $N = 1000$ samples. The results are displayed in Table 4 and they can directly compare these results with the ones obtained in [Zaheer et al., 2017]. The comparison w.r.t. Table 2 shows that pre-training with a GAN can improve the performance of the ISL method.

7.3 Time Series

Experiments are performed using both synthetic and real-world datasets to demonstrate the enhanced forecasting capabilities of the proposed method.

Table 4: ISL combined with a GAN pre-trained generator: Left - Results after applying ISL method; Right - GAN results before ISL method. Hyperparameters: Noise $\sim \mathcal{N}(0,1)$, $K_{max} = 10$, epochs $= 1000$, $N = 1000$.

	GAN+ISL			GAN		
Target	**KSD**	**MAE**	**MSE**	**KSD**	**MAE**	**MSE**
$\mathcal{N}(23,1)$	$\mathbf{7.4e^{-3}}$	**0.0881**	**0.0119**	0.0170	0.1995	0.0549
$\mathcal{U}(22,24)$	**0.0287**	**0.1092**	**0.0327**	0.4488	0.3942	0.2221
Cauchy(23, 1)	$\mathbf{9.43e^{-3}}$	**6.9527**	**937.3329**	0.1278	34.8575	21154
Pareto(23, 1)	**0.0264**	**0.6328**	**0.8890**	0.0474	131.0215	21418626
Model_1	$\mathbf{5.63e^{-3}}$	0.6328	0.8890	0.0160	**0.4165**	**0.4584**
Model_2	**0.0161**	**0.4607**	**0.3350**	0.0399	0.8484	0.9670
Model_3	**0.0805**	**0.4507**	**0.3373**	0.1378	0.7628	1.2044

7.3.1 Synthetic Time Series

To begin, we conduct synthetic experiments to illustrate our method's proficiency in grasping the intrinsic probability distribution of the time series. We delve into the scenario involving autoregressive models with diverse parameters.

Consider a simple autoregressive process AR(p) of the form $X_t = \sum_{i=1}^{p} \phi_i X_{t-i} + \xi_t$. In Table 5 we consider different parameter settings for ϕ_i and the noise variance, ξ. We display the forecasting performance of both temporal ISL compared to DeepAR [Salinas et al., 2020] when the forecasting starts upon observing τ_0 samples of the process. As error metrics, we consider the Normal Deviation (ND), Root-mean-square deviation (RMSE), and Quantile Loss ($QL_{\rho=0.9}$) metrics (For further details, please refer to [Moreno-Pino et al., 2023]).

In this scenario, we have used for both methods (ISL and DeepAR) a two-layer RNN with 10 units each, utilizing a rectified linear unit (RELU) as activation function. This network will be connected with an MLP, consisting of two layers with 16 units each, applying RELU and identity activation functions. The learning rates will be ascertained within the range of $\{10^{-2}, 10^{-3}, 10^{-4}\}$ for the Adams optimizer. We train each model on 200 signals of 1000 elements each.

Table 5: Forecasting an Autoregressive Model AR(p), with noise $\xi = \mathcal{N}(0, 0.01)$.

Meths	τ_0	$\{\phi_i\}_i^p$	**ND**	**RMSE**	$\mathbf{QL_{\rho=0.9}}$
ISL	20	.5, .3, .2	$\mathbf{0.762 \pm 0.37}$	$\mathbf{3.788 \pm 1.84}$	$\mathbf{0.718 \pm 0.69}$
		.5, .2	$\mathbf{0.104 \pm 0.03}$	$\mathbf{0.551 \pm 0.14}$	$\mathbf{0.174 \pm 0.06}$
	50	.5, .3, .2	$\mathbf{0.850 \pm 0.23}$	$\mathbf{6.553 \pm 1.94}$	0.660 ± 0.60
		.5, .2	$\mathbf{0.155 \pm 0.09}$	$\mathbf{1.206 \pm 0.64}$	$\mathbf{0.039 \pm 0.02}$
DeepAR	20	.5, .3, .2	1.031 ± 0.05	4.970 ± 0.49	1.269 ± 0.80
		.5, .2	1.009 ± 0.03	5.537 ± 0.34	1.041 ± 0.49
	50	.5, .3, .2	1.000 ± 0.01	7.630 ± 0.61	$\mathbf{0.650 \pm 0.59}$
		.5, .2	1.006 ± 0.02	8.811 ± 0.35	1.012 ± 0.34

7.3.2 Real World datasets

We evaluated our model using several real-world datasets. The 'electricity-f' dataset captures 15-minute electricity consumption of 370 customers [Trindade, 2015]. The 'electricity-c' dataset aggregates this data into hourly intervals.

Training employed 2012-2013 data, and validation used 20 months starting from 15-April 2014. For this purpose, we have used a 2-layer RNN layers with 3 units each and a 2-layer MLP with 10 units each and a ReLU activation function. In Figure 1c we show the ISL 7-day forecast of one of the instances of the 'electricity-c' dataset. In Tables 6, 7 we compare the ISL performance with ARIMA [Wilson, 2016], Phrophet [Taylor and Letham, 2018], TRMF [Yu et al., 2016], a RNN based State Space Model, DSSM [Rangapuram et al., 2018], a transformer-based model (ConvTrans) [Li et al., 2019b], and DeepAR [Salinas et al., 2020]. As reported, with a relatively simple NN structure, ISL outperforms all baselines in all cases, except for ConvTrans for the 1-day forecast using the metric $\text{QL}_{\rho=0.5}$. Note, however that ConvTrans is much more complex than the proposed model, in terms of the NN structure and the number of parameters. This result proves that the inherent flexibility of the implicit model can compete with more complex prescribed models, usually applied in time-series forecasting. In Table 8, we make a comparison of ISL with different state-of-the-art transformer architectures on various databases (see [Zeng et al., 2023] for details). In this case, we have used a 1-layer RNN with 5 units, followed by a Batch Normalization layer, and a 2-layer MLP with 10 units in each layer. The MLP uses a ReLU activation function in the first layer, an identity activation function in the last layer, and a 5% dropout rate in the first layer. We observe that with a very simple architecture, ISL is competitive and often capable of surpassing transformer architectures for time series prediction.

Table 6: Evaluation summary, using $\mathbf{QL}_{\rho=0.5}/\mathbf{QL}_{\rho=0.9}$ metrics, on electricity-c datasets, forecast window 1 and 7 days.

Method	electricity-$\mathbf{c}_{1d}$	electricity-$\mathbf{c}_{7d}$
ARIMA	0.154/0.102	0.283/0.109
Prophet	0.108/0.099	0.160/0.154
ETS	0.101/0.077	0.121/0.101
TRMF	0.084/–	0.087/–
DSSM	0.083/0.056	0.085/0.052
DeepAR	0.075/0.040	0.082/0.053
ConvTras	**0.059**/0.034	0.076/0.037
ISL	0.062/**0.0189**	**0.063**/**0.021**

Table 7: Evaluation summary, using $\mathbf{QL}_{\rho=0.5}/\mathbf{QL}_{\rho=0.9}$ metrics, on electricity-f datasets, forecast window 1 day.

Method	electricity-$\mathbf{f}_{1d}$
TRMF	0.094/–
DeepAR	0.082/0.063
ConvTrans	0.074/0.042
ISL	**0.050**/**0.036**

7.3.3 Multivariate time series

We have extended our model to incorporate multivariate scenarios. As described in Section 5, our methodology entails independently fitting the marginal distribution for each respective output and training the average ISL. Comprehensive results of this approach, comparing ISL versus transformer techniques in multidimensional cases, are presented in Table 9. In this case, we have use the same architecture as the one described for the results of the Table 8.

As we observe, with an RNN-based structure and a simple MLP implicit model, ISL ($\approx 25K$ parameters) achieves forecasting accuracy better than many state-of-the-art models, which are based on much more complex structures with millions of parameters such as transformers [Zeng et al., 2023].

8 Conclusion and Future Work

We have introduced a novel criterion to train univariate implicit generator functions that simply requires checking uniformity on the statistic defined by Eq. (1). We designed a *surrogate* loss function to optimize the neural network models. In experiments, we showcased the method's ability to capture complex 1D distributions, including multimodal cases and heavy tails, distinguishing it from traditional generative models that frequently face challenges such as mode collapse. Additionally, our model demonstrated effectiveness in handling univariate and multivariate time series data with complex underlying density functions.

Table 8: Univariate long sequence time-series forecasting results on four datasets.

DB	τ	Autoformer		Informer		LogTrans		ISL	
		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	**0.071**	**0.206**	0.193	0.377	0.283	0.468	0.113	0.239
	192	**0.114**	**0.262**	0.217	0.395	0.234	0.409	0.168	0.294
	336	**0.107**	**0.258**	0.202	0.381	0.386	0.546	0.208	0.333
	720	**0.126**	**0.283**	0.183	0.355	0.475	0.629	0.234	0.369
ETTh2	96	**0.153**	**0.306**	0.213	0.373	0.217	0.379	0.184	0.335
	192	0.204	0.351	0.227	0.387	0.281	0.429	**0.177**	0.333
	336	0.246	0.389	0.242	0.401	0.293	0.437	**0.175**	**0.334**
	720	0.268	0.409	0.291	0.439	0.218	0.387	**0.181**	**0.342**
ETTm1	96	**0.056**	0.183	0.109	0.277	0.029	**0.171**	0.065	0.208
	192	**0.081**	**0.216**	0.151	0.310	0.157	0.317	0.146	0.276
	336	**0.076**	**0.218**	0.427	0.591	0.289	0.459	0.107	0.242
	720	0.110	0.267	0.438	0.586	0.430	0.579	**0.092**	**0.224**
Exch.	96	0.241	0.387	0.591	0.615	0.279	0.441	**0.193**	**0.298**
	192	0.273	0.403	1.183	0.912	1.950	1.048	**0.198**	**0.300**
	336	0.508	0.539	1.367	0.984	2.438	1.262	**0.201**	**0.301**
	720	1.111	0.860	1.872	1.072	2.010	1.247	**0.201**	**0.299**

Table 9: Multivariate long sequence time-series forecasting results on four datasets.

DB	τ	Autoformer		Informer		LogTrans		ISL	
		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	0.449	0.459	0.865	0.713	0.878	0.740	**0.384**	**0.459**
	192	0.500	**0.482**	1.008	0.792	1.037	0.824	**0.380**	0.511
	336	0.521	0.496	1.107	0.809	1.238	0.932	**0.334**	**0.392**
	720	**0.514**	**0.512**	1.181	0.865	1.135	0.852	0.747	0.667
ETTh2	96	0.358	**0.397**	3.755	1.525	2.116	1.197	**0.305**	0.435
	192	0.456	**0.452**	5.602	1.931	4.315	1.635	**0.449**	0.525
	336	0.482	**0.486**	4.721	1.835	1.124	1.604	**0.410**	0.511
	720	0.515	0.511	3.647	1.625	3.188	1.540	**0.400**	**0.499**
ETTm1	96	0.672	0.571	0.543	0.510	0.600	0.546	**0.408**	**0.508**
	192	0.795	0.669	**0.557**	**0.537**	0.837	0.700	0.573	0.623
	336	1.212	0.871	0.754	**0.655**	1.124	0.832	**0.702**	0.670
	720	1.166	0.823	0.908	0.724	1.153	0.820	**0.768**	**0.686**
ETTm2	96	**0.255**	**0.339**	0.365	0.453	0.768	0.642	0.276	0.443
	192	**0.281**	**0.340**	0.533	0.563	0.989	0.757	0.335	0.463
	336	0.339	**0.372**	1.363	0.887	1.334	0.872	**0.305**	0.455
	720	0.433	**0.432**	3.379	1.338	3.048	1.328	**0.427**	0.546

Several potential research paths exist. A straightforward direction is the use of random projections to advance ISL capabilities for generating multidimensional data, such as images [Deshpande et al., 2018]. Additionally, a significant research trajectory would involve accurately establishing the order of convergence of the method in relation with its parameters K and N.

Acknowledgments

This work has been supported by the the Office of Naval Research (award N00014-22-1-2647) and Spain's *Agencia Estatal de Investigación* (refs. PID2021-125159NB-I00 TYCHE and PID2021-123182OB-I00 EPiCENTER). Pablo M. Olmos also acknowledges the support by Comunidad de Madrid under grants IND2022/TIC-23550 and ELLIS Unit Madrid.

References

[Alexandrov et al., 2020] Alexandrov, A., Benidis, K., Bohlke-Schneider, M., Flunkert, V., Gasthaus, J., Januschowski, T., Maddix, D. C., Rangapuram,

S., Salinas, D., Schulz, J., et al. (2020). Gluonts: Probabilistic and neural time series modeling in python. *The Journal of Machine Learning Research*, 21(1):4629–4634.

[Arjovsky et al., 2017] Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR.

[Arora et al., 2017] Arora, S., Ge, R., Liang, Y., Ma, T., and Zhang, Y. (2017). Generalization and equilibrium in generative adversarial nets (GANs). In *International conference on machine learning*, pages 224–232. PMLR.

[Arora et al., 2018] Arora, S., Risteski, A., and Zhang, Y. (2018). Do GANs learn the distribution? some theory and empirics. In *International Conference on Learning Representations*.

[Arora and Zhang, 2017] Arora, S. and Zhang, Y. (2017). Do GANs actually learn the distribution? an empirical study. *arXiv preprint arXiv:1706.08224*.

[Billingsley, 1986] Billingsley, P. (1986). *Probability and Measure*. John Wiley and Sons, second edition.

[de Frutos, 2023] de Frutos, J. M. (2023). Training implicit generative models via an invariant statistical loss (isl). https://github.com/josemanuel22/ISL.

[Deshpande et al., 2018] Deshpande, I., Zhang, Z., and Schwing, A. G. (2018). Generative modeling using the sliced wasserstein distance. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3483–3491.

[Devroye et al., 2017] Devroye, L., Györfi, L., Lugosi, G., and Walk, H. (2017). On the measure of voronoi cells. *Journal of Applied Probability*, 54(2):394–408.

[Djuric and Míguez, 2010] Djuric, P. M. and Míguez, J. (2010). Assessment of nonlinear dynamic models by kolmogorov–smirnov statistics. *IEEE transactions on signal processing*, 58(10):5069–5079.

[Dziugaite et al., 2015] Dziugaite, G. K., Roy, D. M., and Ghahramani, Z. (2015). Training generative neural networks via maximum mean discrepancy optimization. *arXiv preprint arXiv:1505.03906*.

[Elvira et al., 2016a] Elvira, V., Míguez, J., and Djurić, P. M. (2016a). Adapting the number of particles in sequential monte carlo methods through an online scheme for convergence assessment. *IEEE Transactions on Signal Processing*, 65(7):1781–1794.

[Elvira et al., 2016b] Elvira, V., Míguez, J., and Djurić, P. M. (2016b). Online adaptation of the number of particles of smc methods. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4378–4382. IEEE.

[Elvira et al., 2021] Elvira, V., Miguez, J., and Djurić, P. M. (2021). On the performance of particle filters with adaptive number of particles. *Statistics and Computing*, 31:1–18.

[Goodfellow et al., 2014] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. *Advances in neural information processing systems*, 27.

[Gouttes et al., 2021] Gouttes, A., Rasul, K., Koren, M., Stephan, J., and Naghibi, T. (2021). Probabilistic time series forecasting with implicit quantile networks. *arXiv preprint arXiv:2107.03743*.

[Gretton et al., 2012] Gretton, A., Borgwardt, K. M., Rasch, M. J., Schölkopf, B., and Smola, A. (2012). A kernel two-sample test. *The Journal of Machine Learning Research*, 13(1):723–773.

[Kingma and Ba, 2014] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[Li et al., 2017] Li, C.-L., Chang, W.-C., Cheng, Y., Yang, Y., and Póczos, B. (2017). MMD GAN: Towards deeper understanding of moment matching network. *Advances in neural information processing systems*, 30.

[Li et al., 2019a] Li, S., Jin, X., Xuan, Y., Zhou, X., Chen, W., Wang, Y.-X., and Yan, X. (2019a). Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. *Advances in neural information processing systems*, 32.

[Li et al., 2019b] Li, S., Jin, X., Xuan, Y., Zhou, X., Chen, W., Wang, Y.-X., and Yan, X. (2019b). *Enhancing the Locality and Breaking the Memory Bottleneck of Transformer on Time Series Forecasting*. Curran Associates Inc., Red Hook, NY, USA.

[Li et al., 2015] Li, Y., Swersky, K., and Zemel, R. (2015). Generative moment matching networks. In *International conference on machine learning*, pages 1718–1727. PMLR.

[Martino et al., 2018] Martino, L., Luengo, D., and Míguez, J. (2018). *Independent random sampling methods*. Springer.

[Mescheder et al., 2017] Mescheder, L., Nowozin, S., and Geiger, A. (2017). Adversarial variational bayes: Unifying variational autoencoders and generative adversarial networks. In *International conference on machine learning*, pages 2391–2400. PMLR.

[Miyato et al., 2018] Miyato, T., Kataoka, T., Koyama, M., and Yoshida, Y. (2018). Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*.

[Mohamed and Lakshminarayanan, 2016] Mohamed, S. and Lakshminarayanan, B. (2016). Learning in implicit generative models. *arXiv preprint arXiv:1610.03483*.

[Moreno-Pino et al., 2023] Moreno-Pino, F., Olmos, P. M., and Artés-Rodríguez, A. (2023). Deep autoregressive models with spectral attention. *Pattern Recognition*, 133:109014.

[Nowozin et al., 2016] Nowozin, S., Cseke, B., and Tomioka, R. (2016). f-GAN: Training generative neural samplers using variational divergence minimization. *Advances in neural information processing systems*, 29.

[Rangapuram et al., 2018] Rangapuram, S. S., Seeger, M. W., Gasthaus, J., Stella, L., Wang, Y., and Januschowski, T. (2018). Deep state space models for time series forecasting. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

[Rasul et al., 2021a] Rasul, K., Seward, C., Schuster, I., and Vollgraf, R. (2021a). Autoregressive Denoising Diffusion Models for Multivariate Probabilistic Time Series Forecasting. In Meila, M. and Zhang, T., editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8857–8868. PMLR.

[Rasul et al., 2021b] Rasul, K., Sheikh, A.-S., Schuster, I., Bergmann, U., and Vollgraf, R. (2021b). Multivariate Probabilistic Time Series Forecasting via Conditioned Normalizing Flows. In *International Conference on Learning Representations 2021*.

[Rodríguez-Santana et al., 2022] Rodríguez-Santana, S., Zaldivar, B., and Hernandez-Lobato, D. (2022). Function-space inference with sparse implicit processes. In *International Conference on Machine Learning*, pages 18723–18740. PMLR.

[Rosenblatt, 1952] Rosenblatt, M. (1952). Remarks on a multivariate transformation. *The annals of mathematical statistics*, 23(3):470–472.

[Salimans et al., 2016] Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., and Chen, X. (2016). Improved techniques for training GANs. *Advances in neural information processing systems*, 29.

[Salinas et al., 2020] Salinas, D., Flunkert, V., Gasthaus, J., and Januschowski, T. (2020). Deepar: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, 36(3):1181–1191.

[Santos et al., 2019] Santos, C. N. d., Mroueh, Y., Padhi, I., and Dognin, P. (2019). Learning implicit generative models by matching perceptual features. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4461–4470.

[Scotta and Messina, 2023] Scotta, S. and Messina, A. (2023). Understanding and contextualising diffusion models. *arXiv preprint arXiv:2302.01394*.

[Sohier, cess] Sohier, D. (Year of Access). 30 years of european wind generation. https://www.kaggle.com/datasets/sohier/30-years-of-european-wind-generation.

[Taylor and Letham, 2018] Taylor, S. J. and Letham, B. (2018). Forecasting at scale. *The American Statistician*, 72(1):37–45.

[Trindade, 2015] Trindade, A. (2015). ElectricityLoadDiagrams20112014. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C58C86.

[Uppal et al., 2019] Uppal, A., Singh, S., and Póczos, B. (2019). Nonparametric density estimation & convergence rates for GANs under besov ipm losses. *Advances in neural information processing systems*, 32.

[Wilson, 2016] Wilson, G. T. (2016). Time Series Analysis: Forecasting and Control, 5th Edition , by George E. P. Box , Gwilym M. Jenkins , Gregory C. Reinsel and Greta M. Ljung , 2015 . Published by John Wiley and Sons Inc. , Hoboken, N. *Journal of Time Series Analysis*, 37(5):709–711.

[Wu et al., 2021] Wu, H., Xu, J., Wang, J., and Long, M. (2021). Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in Neural Information Processing Systems*, 34:22419–22430.

[Yu et al., 2016] Yu, H.-F., Rao, N., and Dhillon, I. S. (2016). Temporal regularized matrix factorization for high-dimensional time series prediction. In Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc.

[Zaheer et al., 2017] Zaheer, M., Li, C.-l., Póczos, B., and Salakhutdinov, R. (2017). Gan connoisseur: Can gans learn simple 1D parametric distributions. In *Proceedings of the 31st Conference on Neural Information Processing Systems*, pages 1–6.

[Zeng et al., 2023] Zeng, A., Chen, M., Zhang, L., and Xu, Q. (2023). Are transformers effective for time series forecasting? In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pages 11121–11128.

[Zhou et al., 2021] Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H., and Zhang, W. (2021). Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 11106–11115.

9 Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. **Yes. This analysis has been carried out in sections 2 and 3 of the paper.**
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. **Yes. This analysis has been carried out in sections 2 and 3 of the paper.**
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. **Yes. We have included our anonymized repository. The package/project dependencies and compatibility constraints are listed in the project file.**
2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. Yes
   (b) Complete proofs of all theoretical results. **Yes**
   (c) Clear explanations of any assumptions. **Yes**
3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). **Yes. See appendix, Section 3.**
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). **Yes. The details of the hyperparameters are described in detail in each section where the experiment is discussed.**
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). **Yes**
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). **Yes. It is included in the appendix in Section 3.**
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. **Yes**
   (b) The license information of the assets, if applicable. **Yes**
   (c) New assets either in the supplemental material or as a URL, if applicable. **Yes**
   (d) Information about consent from data providers/curators. **Not Applicable**
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. **Not Applicable**
5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. **Not Applicable**
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. **Not Applicable**
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. **Not Applicable**

Appendix

Contents

1 MISSING PROOFS

For $f : \mathbb{R} \mapsto \mathbb{R}$ a real function and p the pdf of an univariate real r.v. let us denote,

$$(f, p) := \int_{-\infty}^{\infty} f(x)p(x)dx$$

and let $B_1(\mathbb{R}) := \left\{(f : \mathbb{R} \to \mathbb{R}) : \sup_{x\in\mathbb{R}} |f(x)| \leq 1\right\}$ be the set of real functions bounded by 1.

Let us also denote the indicator function as

$$I_A(x) = \begin{cases} 1, & \text{if } x \in A, \\ 0, & \text{if } x \notin A, \end{cases}$$

for $A \subset \mathbb{R}$ and $x \in \mathbb{R}$.

Theorem 2. *If* $\sup_{f\in B_1(\mathbb{R})} |(f, p) - (f, \tilde{p})| \leq \epsilon$ *then,*

$$\frac{1}{K+1} - \epsilon \leq \mathbb{Q}_K(n) \leq \frac{1}{K+1} + \epsilon, \ \forall n \in \{0, \ldots, K\}.$$

Proof. Choose $y_0 \in \mathbb{R}$. The univariative distribution function (cdf) asociated to the pdf p is

$$F(y_0) = \int_{-\infty}^{y_0} p(y)dy = \big(I_{(-\infty,y_0)}, p\big),$$

and for $\tilde{p}$ we obtain the cdf

$$\tilde{F}(y_0) = \int_{-\infty}^{y_0} \tilde{p}(y)dy = \big(I_{(-\infty,y_0)}, \tilde{p}\big).$$

Now, the rank statistic can be interpreted in terms of a binomial r.v. To be specific, choose some $y_0 \in \mathbb{R}$ and then run K Bernoulli trials where, for the i-th trial, we draw $\tilde{y}_i \sim \tilde{p}$ and declare a success if, and only if, $\tilde{y}_i < y_0$. It is apparent that the probability of success for these trials is $\tilde{F}(y_0)$ and the probability of having exactly n successes is

$$h_n(y_0) = \binom{K}{n} (\tilde{F}(y_0))^n (1 - \tilde{F}(y_0))^{K-n}.$$

If we now let $y_0 \sim p$ and integrate $h_n(y_0)$ with respect to $p(y)$ we obtain the probability of the event $A_K = n$, i.e.

$$\mathbb{Q}_K(n) = (h_n, p), \quad \forall n \in \{0, \ldots, K\}.$$

Now, since $h_n \in B_1(\mathbb{R})$, the assumption in the statement of Theorem 2 yields

$$|(h_n, \tilde{p}) - (h_n, p)| \leq \epsilon,$$

which, in turn, implies

$$(h_n, \tilde{p}) - \epsilon \leq (h_n, p) \leq (h_n, \tilde{p}) + \epsilon.$$

However, $(h_n, p) = \mathbb{Q}_K(n)$ by construction. On the other hand, if we let $y_0 \sim \tilde{p}$ and integrate with respect to $\tilde{p}$ Theorem 1 yields that $(h_n, \tilde{p}) = \dfrac{1}{K+1}$, hence we obtain the inequality

$$\frac{1}{K+1} - \epsilon \leq \mathbb{Q}_K(n) \leq \frac{1}{K+1} + \epsilon,$$

and conclude the proof. $\square$

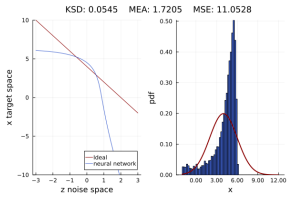
2 ADDITIONAL EXPERIMENTS

In this section of the appendix, we expand upon the experiments conducted. In the first part, we will study how the results of the ISL method for learning 1-D distributions change depending on the chosen hyperparameters. Later, in a second part, we will examine how these evolve over time. Finally, we will extend the experiments conducted with respect to time series, considering a mixture time series case. We will also provide additional results for the 'electricity-c' and 'electricity-f' series, and lastly, we will include the results obtained for a new time series.

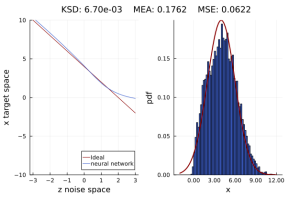
2.1 The Effect of Parameter K_{max}

We will first analyze the effects of the evolution of the hyperparameter K_{max} for different target distributions. For training, the number of epochs is 1000, the learning rate 10^{-2}, and the weights are updated using Adam optimizer. Also, the number of (ground-truth) observations is, in every case, $N = 1000$. First, we consider a standard normal distribution, $\mathcal{N}(0, 1)$, as the source from which we sample random noise to generate the synthetic data. As generator, we use a 4-layer multilayer perceptron (MLP) with $7, 13, 7$ and 1 units at the corresponding layers. As activation function we use a exponential linear unit (ELU). We specify the target distribution and K_{max} in each subcaption.

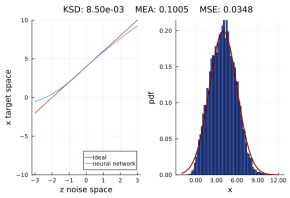
At the sight of the results (see Figure 1 and 2) we observe that a higher value of K_{max} generally leads to more accurate results compared to the true distribution. However, this is not always the case because, as demonstrated in the example of the Pareto distribution, the value of K achieved during training is lower (specifically 6) than the maximum limit set for K. In such cases, K_{max} does not have an effect.



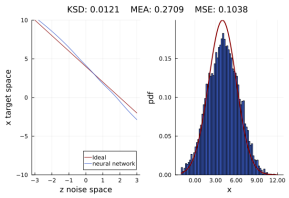
$\mathcal{N}(4,2),\ K_{max}=2$



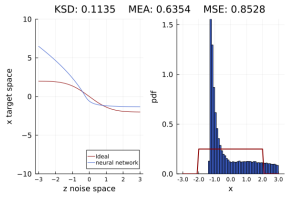
$\mathcal{N}(4,2),\ K_{max}=5$



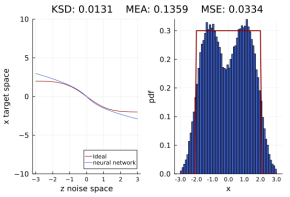
$\mathcal{N}(4,2),\ K_{max}=10$



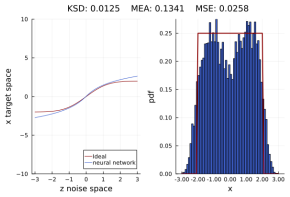
$\mathcal{N}(4,2),\ K_{max}=20$



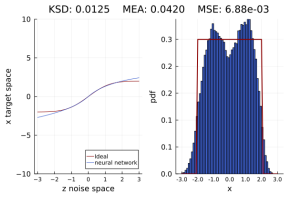
$\mathcal{U}(-2,2),\ K_{max}=2$



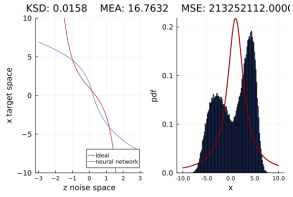
$\mathcal{U}(-2,2),\ K_{max}=5$



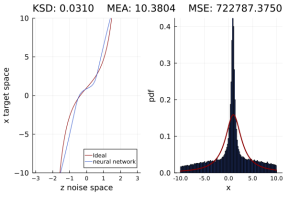
$\mathcal{U}(-2,2),\ K_{max}=10$



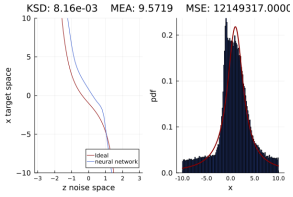
$\mathcal{U}(-2,2),\ K_{max}=20$



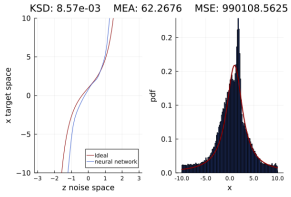
Cauchy$(1,2),\ K_{max}=2$



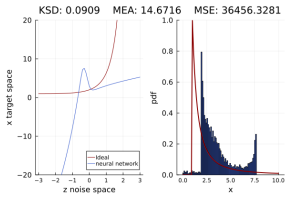
Cauchy$(1,2),\ K_{max}=5$



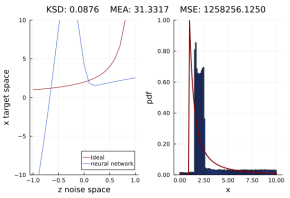
Cauchy$(1,2),\ K_{max}=10$



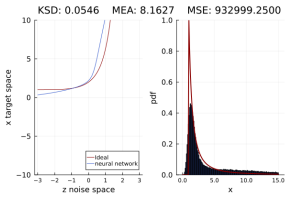
Cauchy$(1,2),\ K_{max}=20$



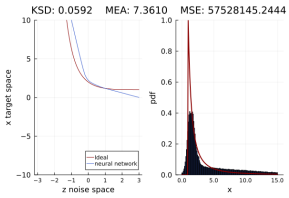
Pareto$(1,1),\ K_{max}=2$



Pareto$(1,1),\ K_{max}=5$

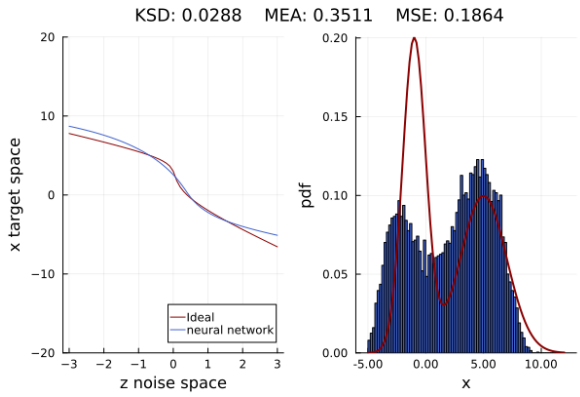


Pareto$(1,1),\ K_{max}=10$

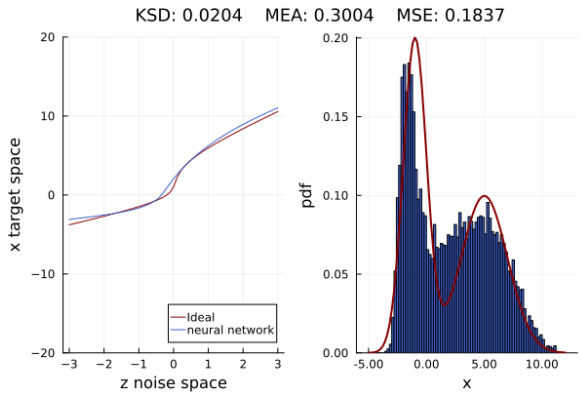


Pareto$(1,1),\ K_{max}=20$

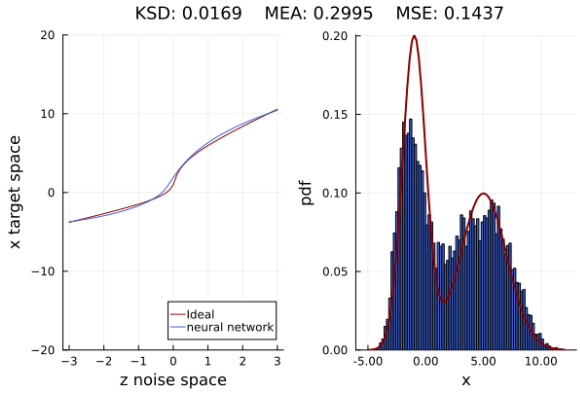
Figure 1: Results obtained for varying K_{max} for different target distributions. Global parameters: $N = 1000$, $Epochs = 1000$, $LearningRate = 10^{-2}$, and Initial Distribution=$\mathcal{N}(0,1)$.



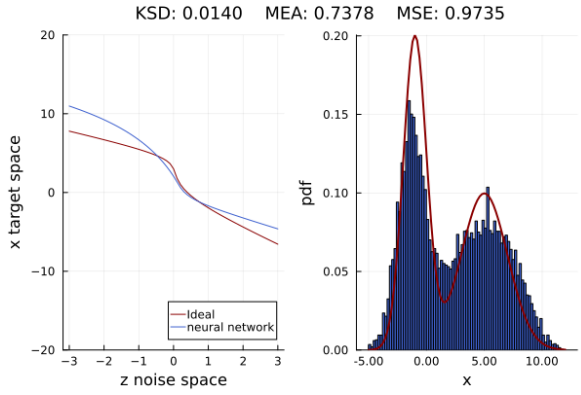
$\text{Model}_1,\ K_{max} = 2$



$\text{Model}_1,\ K_{max} = 5$



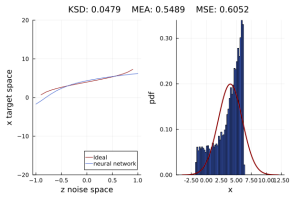
$\text{Model}_1,\ K_{max} = 10$



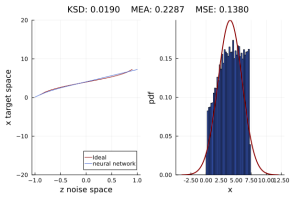
$\text{Model}_1,\ K_{max} = 20$

Figure 2: Results obtained for varying K_{max} for Target Distributions = Model_1.
Global parameters: $N = 1000$, $Epochs = 1000$, $LearningRate = 10^{-2}$, and Initial Distribution=$\mathcal{N}(0, 1)$.

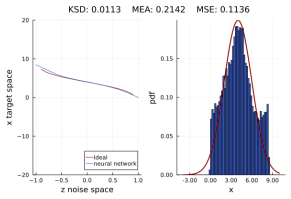
As before, we are analyzing the effects of the evolution of the hyperparameter K_{max} for different distributions. In this case, we consider the random noise originating from an uniform distribution $\mathcal{U}(-1, 1)$. The other settings are identical to the $\mathcal{N}(0, 1)$ case.



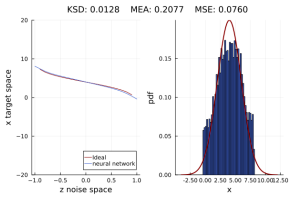
$\mathcal{N}(4, 2),\ K_{max} = 2$



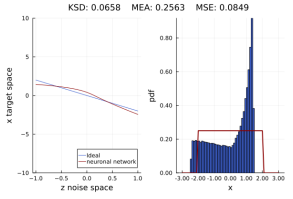
$\mathcal{N}(4, 2),\ K_{max} = 5$



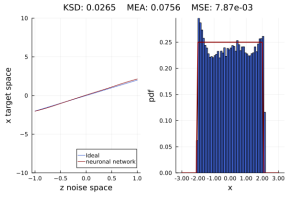
$\mathcal{N}(4, 2),\ K_{max} = 10$



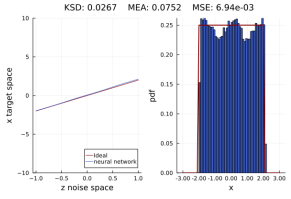
$\mathcal{N}(4, 2),\ K_{max} = 20$



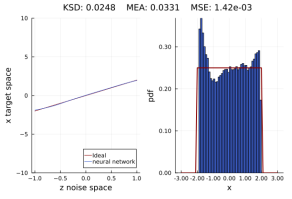
$\mathcal{U}(4, 2),\ K_{max} = 2$



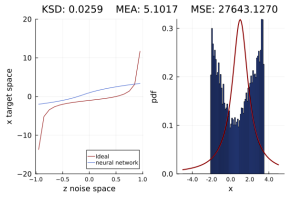
$\mathcal{U}(4, 2),\ K_{max} = 5$



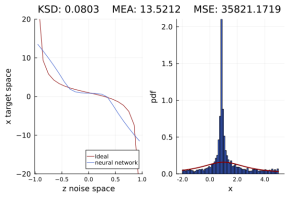
$\mathcal{U}(4, 2),\ K_{max} = 10$



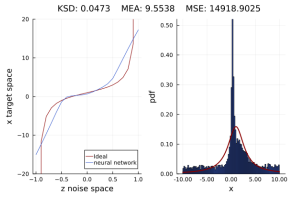
$\mathcal{U}(4, 2),\ K_{max} = 20$



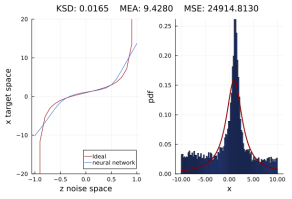
Cauchy$(1, 2),\ K_{max} = 2$



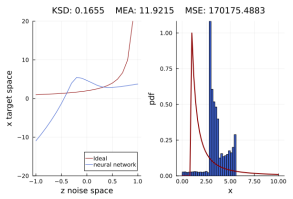
Cauchy$(1, 2),\ K_{max} = 5$



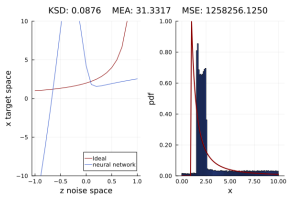
Cauchy$(1, 2),\ K_{max} = 10$



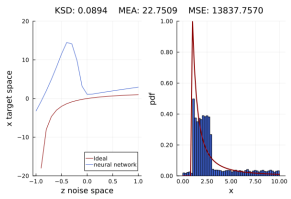
Cauchy$(1, 2),\ K_{max} = 20$



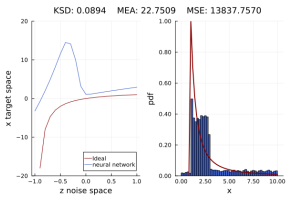
Pareto$(1, 1),\ K_{max} = 2$



Pareto$(1, 1),\ K_{max} = 5$

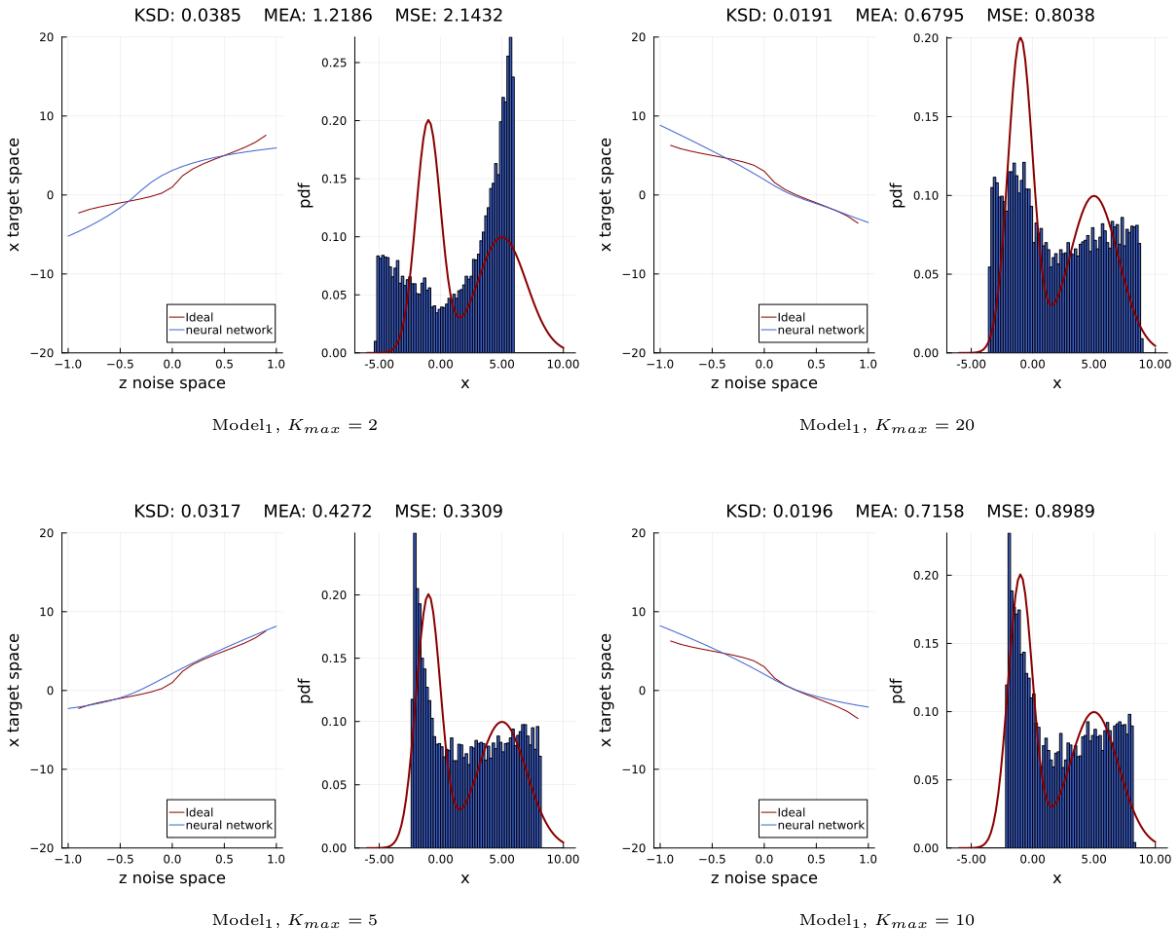


Pareto$(1, 1),\ K_{max} = 10$



Pareto$(1, 1),\ K_{max} = 20$

Figure 3: Results obtained for varying K_{max} for different target distributions. Global parameters: $N = 1000$, $Epochs = 1000$, Learning Rate $= 10^{-2}$, and Initial Distribution $= \mathcal{U}(-1, 1)$.



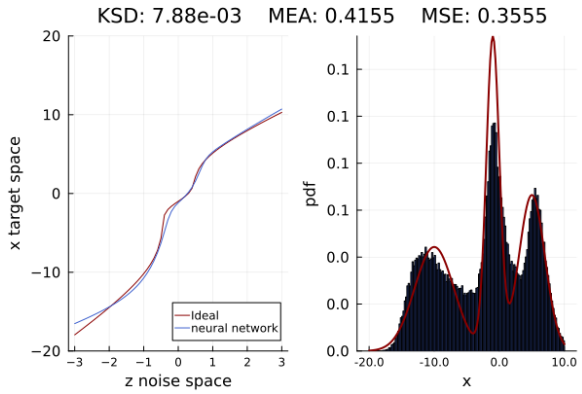
Model_1, $K_{max} = 2$ Model_1, $K_{max} = 20$

Model_1, $K_{max} = 5$ Model_1, $K_{max} = 10$

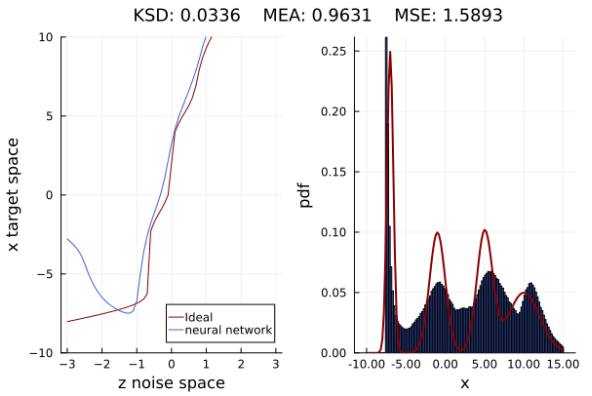
Figure 4: Results obtained for varying K_{max} for Target Distributions = Model_1. Global parameters: $N = 1000$, $Epochs = 1000$, Learning Rate = 10^{-2}, and Initial Distribution = $\mathcal{U}(0, 1)$.

2.2 Experiments with large hyperparameters K, N and number of epochs

In this section, we investigate the capacity of ISL to learn complex distributions in scenarios where the hyperparameters K and N are set to large values. In each case, we specify the values of the hyperparameters. As we can see, in these two examples with large values of K_{max}, N, and allowing for sufficient training time (number of epochs is large), the method is capable of learning complex multimodal distributions.



(a) $N = 10000$, $K_{max} = 10$, epochs=10000, lr $= 10^{-2}$, Target Distribution = Model$_3$.



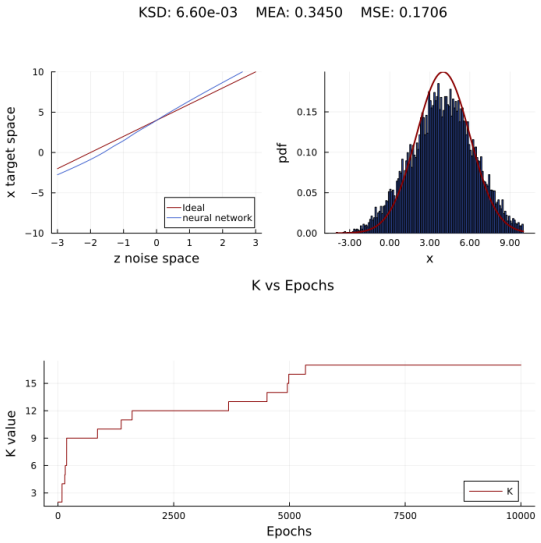
(b) $N = 5000$, $K_{max} = 50$, epochs=10000, lr $= 10^{-2}$ Target Distribution = MixtureModel$\big(\mathcal{N}(-7, 0.5), \mathcal{N}(-1, 1), \mathcal{N}(4, 1), \mathcal{N}(10, 2)\big)$

Figure 5: Complex Distribution Learning with High Values of Hyperparameters: K, N, and Epochs.

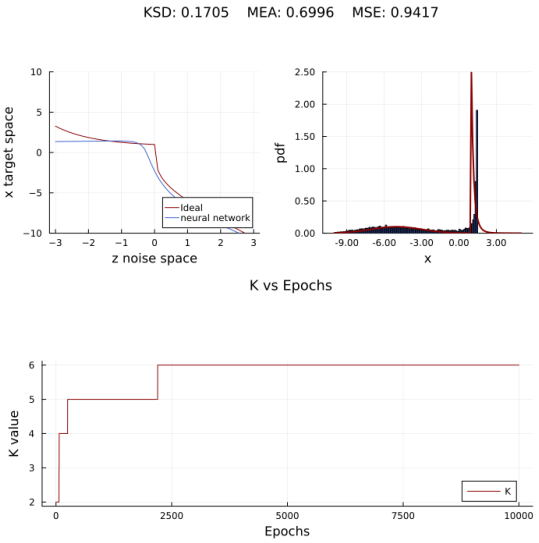
2.3 Evolution of the hyperparameter K

Here, we study the evolution of hyperparameter K during training. It is progressively adapted according to the algorithm explained in Section 4.2 of the paper. In every experiment we use a learning rate of 10^{-2} and train for 10000 epochs. The number of ground-truth observations is $N = 2,000$. In this case, we haven't imposed any restrictions on K_{max}, and hence K can grow as required. In the subcaption, the target distribution is specified. The initial distribution is a $\mathcal{N}(0, 1)$ in all cases.

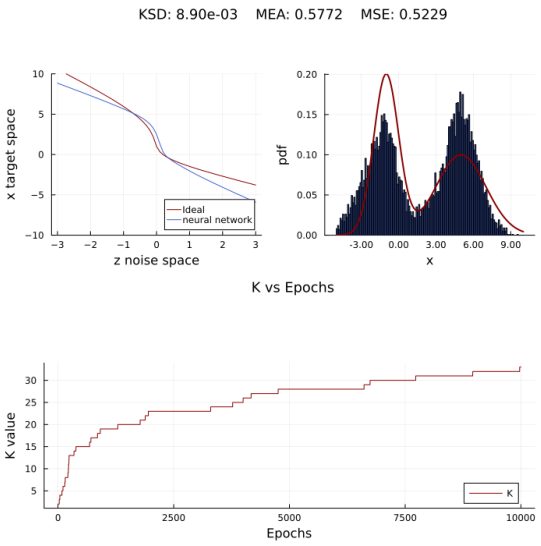
We notice that initially, as the number of epochs increases, the value of K grows very rapidly, but it eventually levels off after reaching a certain point. The observed data (see Figure 6) leads us to infer that obtaining a high K value requires a progressively more effort as compared to attaining a lower value, as evidenced by the incremental increases over successive training epochs. This complexity seems to exhibit a logarithmic pattern. Therefore, there is an advantage in initially setting smaller values of K and gradually increasing them.



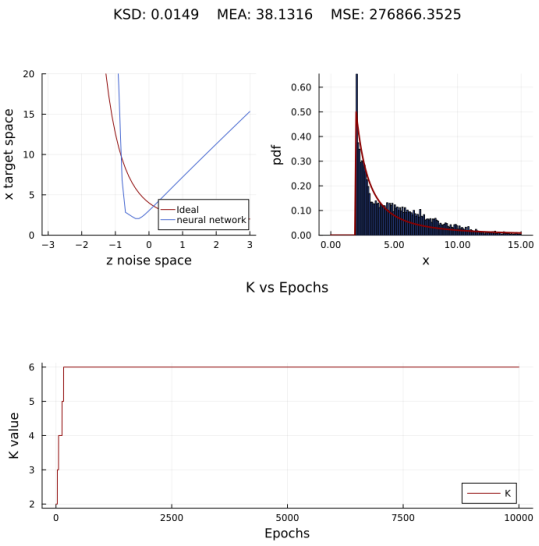
(a) $\mathcal{N}(4, 2)$



(c) Model_3



(b) Model_1



(d) $\text{Pareto}(1, 2)$, $N = 2000$

Figure 6: Evolution of the hyperparameter K as a function of the epochs. Figure top left: We compare the optimal transformation from a $\mathcal{N}(0, 1)$ to the target distribution versus the generator learned by ISL, Figure top right: The resulting distribution, Figure at the bottom: The evolution of K as a function of epochs.

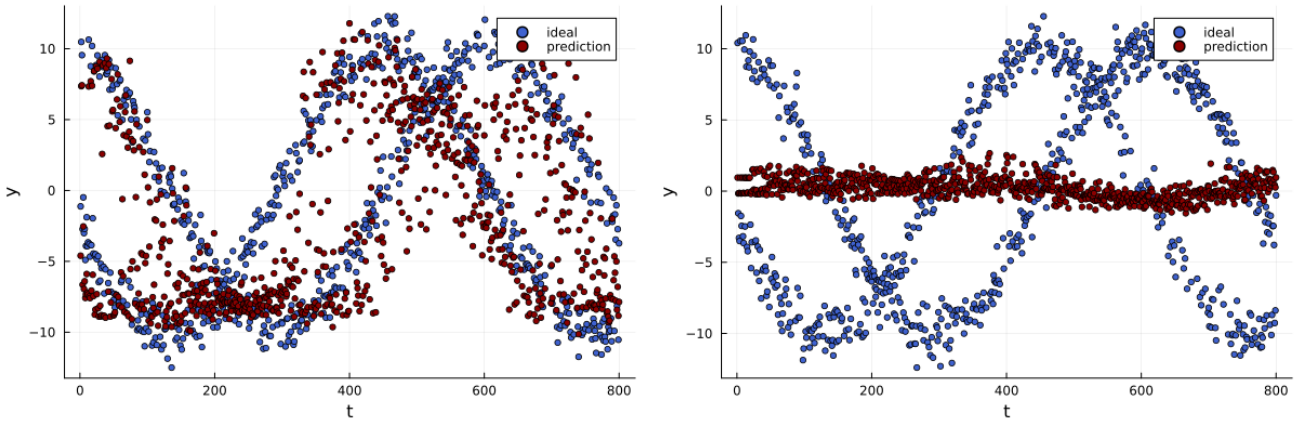
2.4 Mixture time series

We now consider the following composite time series, such that the underlying density function is multimodal. For each time t, a sample is taken from functions,

$$y_1(t) = 10\cos(t - 0.5) + \xi \quad \text{or,} \quad y_2(t) = 10\cos(t - 0.5) + \xi,$$

based on a Bernoulli distribution with parameter $1/2$. Noise, denoted by ξ, follows a standard Gaussian distribution, $\mathcal{N}(0, 1)$. Our temporal variable, t, will take values in the interval $(-4, 4)$ where each point will be equidistantly spaced by 0.1 units apart (when we represent it, we use the conventional notation $t \geq 0$). The neural network comprises two components: an RNN with three layers featuring 16, 16 and finally 32 units; and a feedforward neural network with three layers containing 32, 64 units and 64 units. Activation functions include ELU for the RNN layers and ELU for the MLP layers, except for the final layer, which employs an identity activation function. The chosen learning rate is 10^{-3} in an Adam optimizer.

In the Figure 7, we display the predictions obtained for the neural network trained using ISL and DeepAR, with a prediction window of 20 steps.



(a) ISL forecasting

(b) DeepAR forecasting

Figure 7: Mixture time series, forecasting 20-step

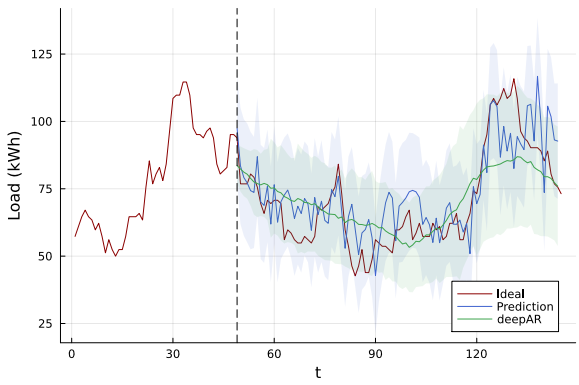
2.5 More experiments with real world datasets

In this section, we delve deeper into the experiments carried out with real world datasets. Previously, we provided a brief overview of the results pertaining to 'electric-f' and 'electric-c.' Within this section, we will present comparative graphs for these two datasets and introduce the results related to the wind dataset.

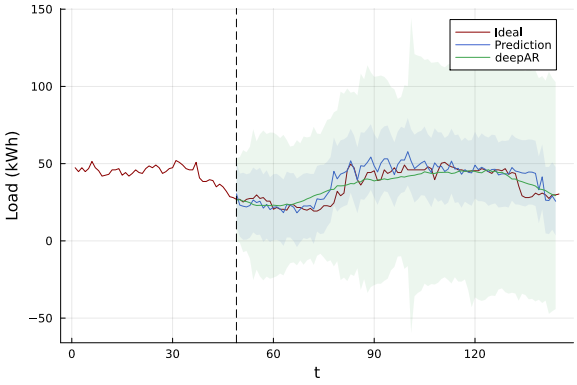
2.5.1 Electricity-f time series

We present comparative graphs of the results achieved by ISL using the 'electricity-f' dataset. The training phase utilized data from 2012 to 2013, while validation encompassed a span of 20 months, commencing on April 15, 2014. For this purpose, we employed a model architecture consisting of 2 layers-deep RNN with 3 units each, as well as a 2-layer MLP with 10 units each and a ReLU activation function.

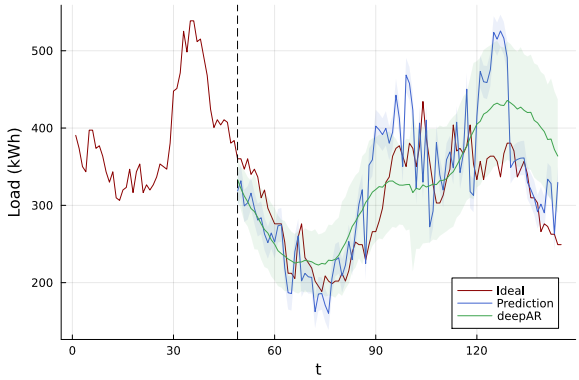
In the case of training with DeepAR, we utilized the GluonTS library [Alexandrov et al., 2020] with its default settings, which entail 2 recurrent layers, each comprising 40 units. The model was trained over the course of 100 epochs. The results presented in Figure 8 relate to the prediction of the final day within the series for different users, with the shaded area indicating the range of plus or minus one standard deviation from the predicted value.



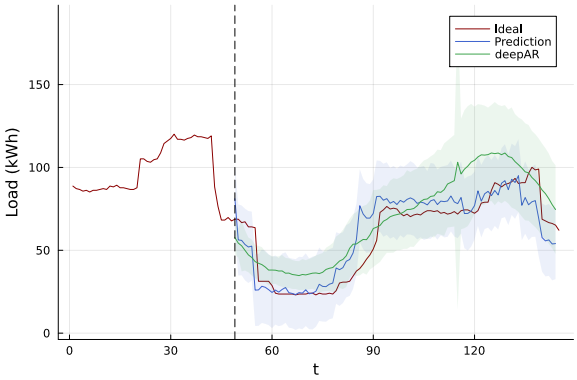
(a) user 5



(c) user 169



(b) user 8

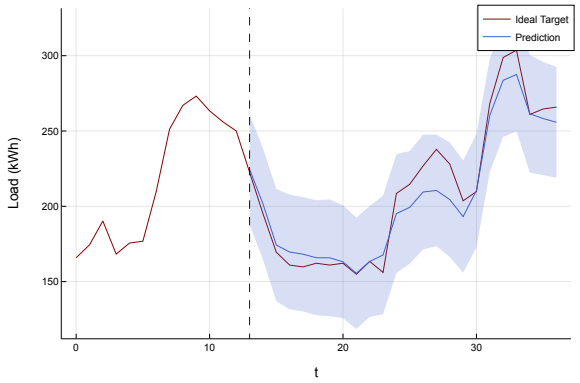


(d) user 335

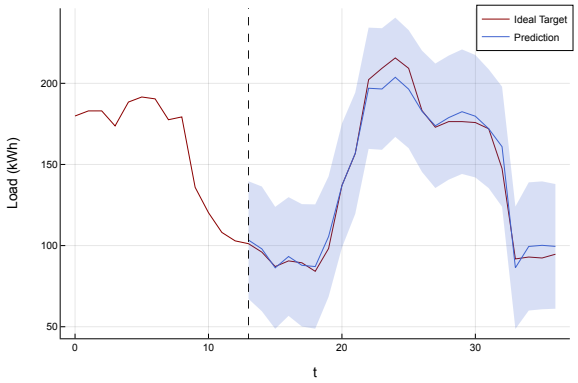
Figure 8: 1-day prediction of 'electricity-f' consumption. Blue: 1-day prediction using ISL, Green: 1-day prediction using DeepAR.

2.5.2 Electricity-c time series

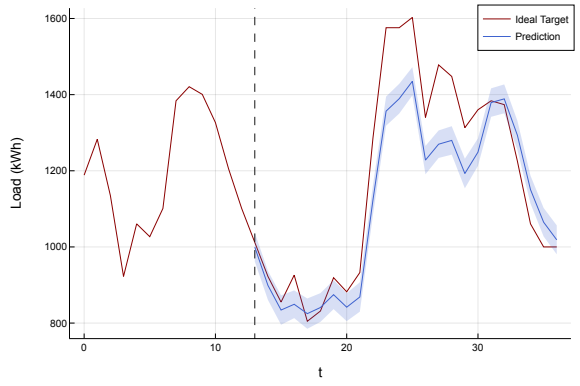
We present comparative graphs of the results achieved by ISL using the 'electricity-c' dataset. We used the same settings for both training and validation, as well as the same neural network as described for the 'electricity-f' dataset. Below, we present various prediction plots for different users, both for a 1-day (Figure 9) and a 7-day forecast (Figure 10).



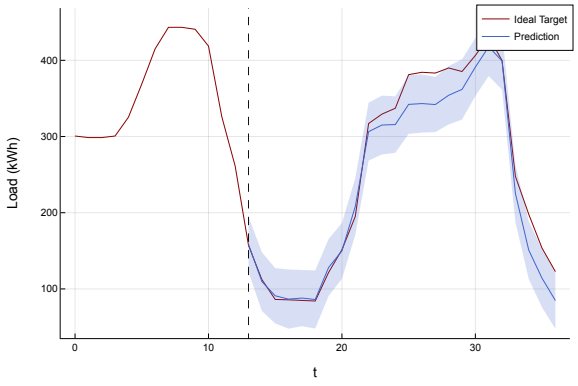
(a) user 5



(c) user 169

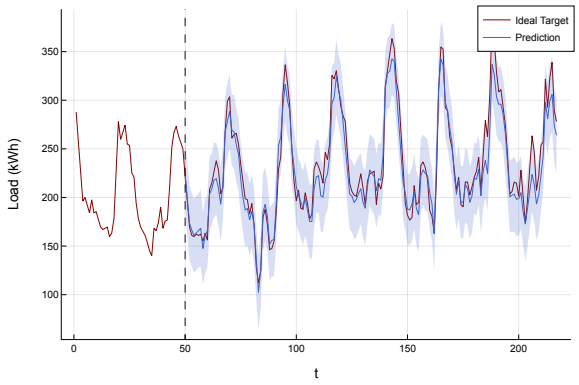


(b) user 8

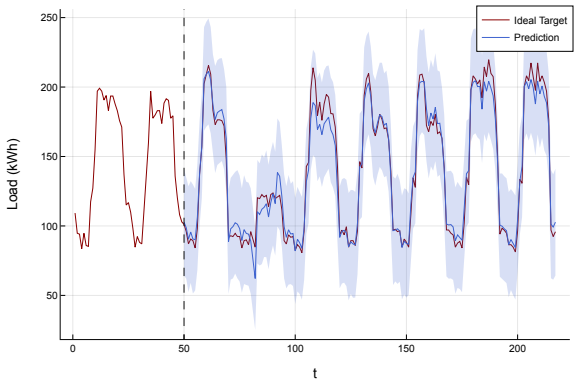


(d) user 335

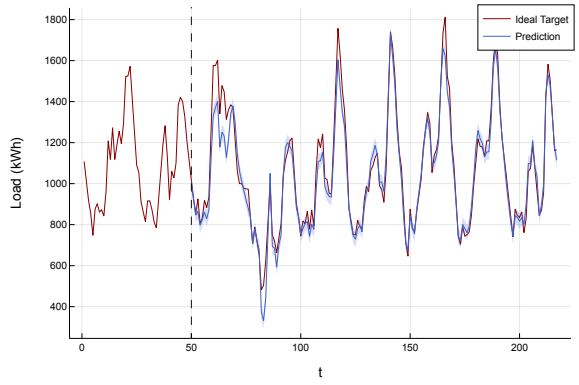
Figure 9: 1-day forecast for 'electricity-c' consumption



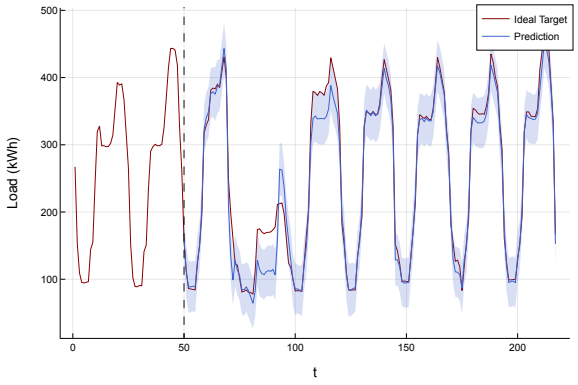
(a) user 5



(c) user 169



(b) user 8

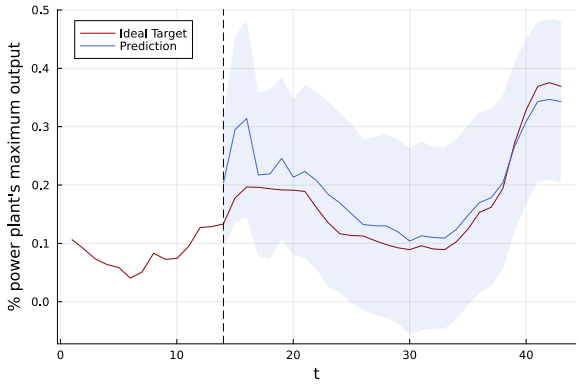


(d) user 335

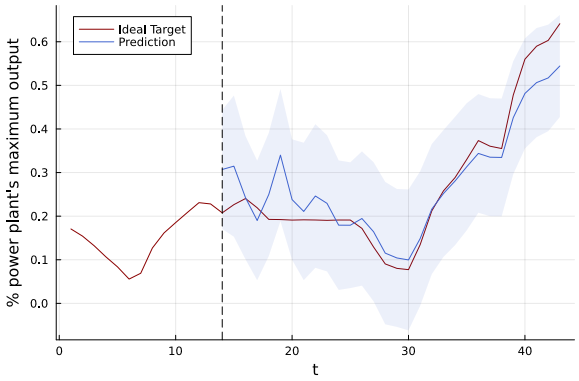
Figure 10: 7-day forecast for 'electricity-c' consumption

2.5.3 Wind time series

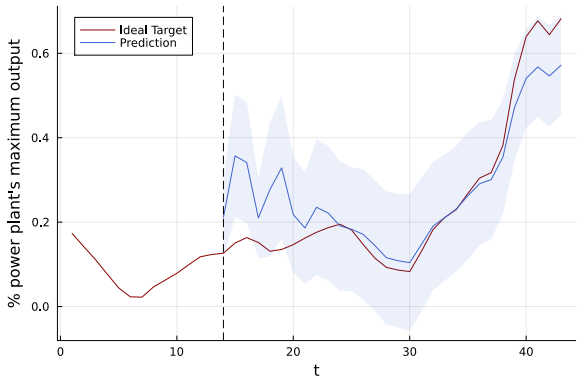
The wind [Sohier, cess] dataset contains daily estimates of the wind energy potential for 28 countries over the period from 1986 to 2015, expressed as a percentage of a power plant's maximum output. The training was conducted using data from a 3-year period (1986-1989), and validation was performed on three months of data from 1992. Below, we present the results obtained for 30-day and 180-day predictions for different time series. For this purpose, we used a 3-layer RNN with 32 units in each layer and a 2-layer MLP with 64 units in each layer, both with ReLU activation functions. The final layer employs an identity activation function.



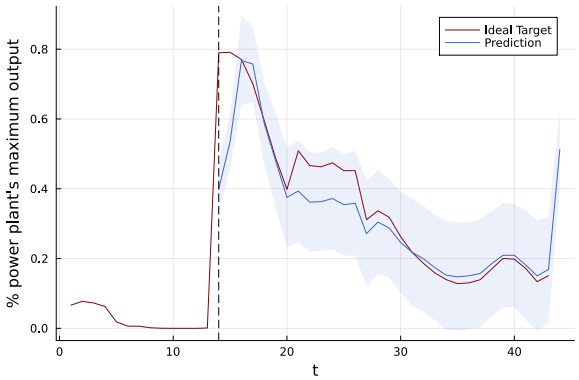
(a) FR22



(c) FR51

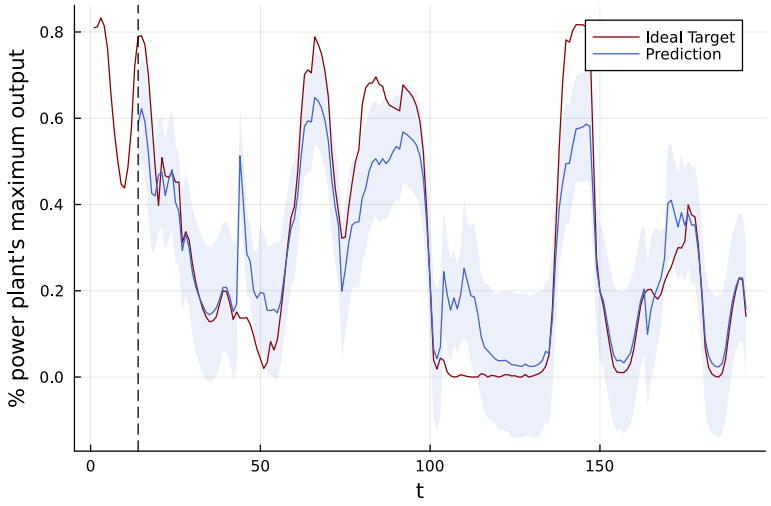


(b) FR25



(d) AT31

Figure 11: 30 days forecast prediction



(a) FR82

Figure 12: 180 days forecast prediction

Table 1: Evaluation summary, using $\mathbf{QL}_{\rho=0.5}/\mathbf{QL}_{\rho=0.9}$ metrics, on wind datasets, forecast window 30 day

Method	Wind
Prophet	0.305/0.275
TRMF	0.311/−
N-BEATS	0.302/0.283
DeepAR	0.286/0.116
ConvTras	0.287/**0.111**
ISL	**0.204**/0.245

3 Experimental Setup

All experiments were conducted using a personal computer with the following specifications: a MacBook Pro with macOS operating system version 13.2.1, an Apple M1 Pro CPU, and 16GB of RAM. The specific hyperparameters for each experiment are detailed in the respective sections where they are discussed. The repository with the code can be found in at the following link https://github.com/josemanuel22/ISL.