ECAI 2024
U. Endriss et al. (Eds.)

doi:10.3233/FAIA240835

VMFTransformer: An Angle-Preserving and Auto-Scaling Machine for Multi-Horizon Probabilistic Forecasting

Yunyi Zhou[a], Ruohan Gao[a], Xinping Zheng[a], Yuchen Huang[a] and Zhixuan Chu[b,*]

[a]Ant Group
[b]Zhejiang University

Abstract. As deep learning develops, the major research methodologies of time series forecasting can be divided into two categories, i.e., iterative and direct methods. In the iterative methods, since a small amount of error is produced at each time step, the recursive structure can potentially lead to large error accumulations over longer forecasting horizons. Although the direct methods can avoid this puzzle involved in the iterative methods, they face abuse of conditional independence among time points. This impractical assumption can also lead to biased models. To solve these challenges, we propose a direct approach for multi-horizon probabilistic forecasting, which can effectively characterize the dependence across future horizons. Specifically, we consider the multi-horizon target as a random vector. The direction of the vector embodies the temporal dependence, and the length of the vector measures the overall scale across each horizon. Therefore, we respectively apply the von Mises-Fisher (VMF) distribution and the truncated normal distribution to characterize the target vector's angle and magnitude in our model. Extensive results demonstrate the superiority of our framework over eight state-of-the-art methods.

1 Introduction

Time series forecasting has historically been a key area of academic research and industrial applications [5, 44, 6, 9, 17, 34, 39]. The common requirement of time series forecasting is measuring the uncertainty of the output by predicting its probability distribution, which is termed "probabilistic forecasting". In addition, the practical use of probabilistic forecasting generally requires forecasting more than one step, i.e., multi-horizon forecasting. Modern machine learning methods have been proposed for multi-horizon probabilistic forecasting, which can be divided into iterative and direct methods as follows.

Iterative approaches typically make use of autoregressive deep learning architectures, which produce multi-horizon forecasts by recursively feeding samples of the target into future time steps. Iterative approaches generally make use of the "chain rule" in the training stage, which decomposes $p(\mathbf{y}_{T+1:T+H}|\mathbf{y}_{0:T}, \mathbf{x})$ as $\prod_{t=T+1}^{T+H} p(y_t|\mathbf{y}_{0:t-1}, \mathbf{x})$ [31], where $\mathbf{y}_{0:T} = (y_0, y_1, \ldots, y_T)^T$ denotes a slice of the time series and $\mathbf{x}$ represents some external features. Due to the chain rule, iterative approaches transform the estimation of $p(\mathbf{y}_{T+1:T+H}|\mathbf{y}_{0:T}, \mathbf{x})$ into a one-step-ahead prediction in the training stage and feed the prediction of y_{t-1} back as ground truth to forecast y_t. However, as pointed out in [2, 21, 36], the discrepancy between actual data and estimates during prediction can lead to error accumulation. Since a small amount of error is produced at each time step, the recursive structure of iterative methods can potentially lead to large error accumulations over long forecasting horizons. Therefore, the iterative approaches are less robust and might lead to a biased model [7, 32, 8].

For the direct methods [36, 23, 12], they can alleviate the above-mentioned issues involved in iterative methods, which can directly forecast all targets by using all available inputs. They typically use sequence-to-sequence architecture, a type of network structure mapping sequences directly to sequences. These methods commonly choose the quantile loss function as the training objective. They try to jointly minimize the quantile loss at each future horizon, where a decoder structure propagates encoded historical information and processes external features that can be acquired in advance. However, minimizing a loss function at each horizon, which is equivalent to transforming $p(\mathbf{y}_{T+1:T+H}|\mathbf{y}_{0:T}, \mathbf{x})$ into $\prod_{h=1}^{H} p(y_{T+h}|\mathbf{y}_{0:T}, \mathbf{x})$, is somewhat abuse of conditional independence. The discrepancy between $p(\mathbf{y}_{T+1:T+H}|\mathbf{y}_{0:T}, \mathbf{x})$ and $\prod_{h=1}^{H} p(y_{T+h}|\mathbf{y}_{0:T}, \mathbf{x}, h)$ can also lead to biased models.

Taking all the aforementioned challenges into account, we propose a direct approach for multi-horizon probabilistic forecasting which can effectively capture the characteristics of the dependence across future horizons. We consider the multi-horizon target $\mathbf{y}_{T+1:T+H}$ as a random vector in an H-dimension vector space. When there exists a certain dependence mechanism among y_{T+h} $(1 \leq h \leq H)$, $\mathbf{y}_{T+1:T+H}$ is likely to be distributed around a specific direction, which can be characterized by the angles of a vector relative to an orthonormal basis of the vector space. Therefore, we apply the von Mises-Fisher (VMF) distribution, which is a probability distribution on the surface of a unit-sphere, to characterize the distribution of the direction of $\mathbf{y}_{T+1:T+H}$. Once the direction of $\mathbf{y}_{T+1:T+H}$ is learned, and suppose $||\mathbf{y}_{T+1:T+H}||_2$ is given, the forecast can be made by multiplying $||\mathbf{y}_{T+1:T+H}||_2$ with its direction. Hence, we normalize $\mathbf{y}_{T+1:T+H}$ by dividing its length and adopt a prior distribution on $||\mathbf{y}_{T+1:T+H}||_2$ to obtain a complete tractable likelihood function. Recall that the direction of $\mathbf{y}_{T+1:T+H}$ is defined via angles, and

* Corresponding Author. Email: zhixuanchu@zju.edu.cn. The author is with the State Key Laboratory of Blockchain and Data Security & Hangzhou High-Tech Zone (Binjiang) Institute of Blockchain and Data Security, Hangzhou, China.

its length is determined by the scale of each y_{T+h}, the similarity measurement of the attention module involved in our model is correspondingly modified, which is capable of evaluating the joint similarity between angles and scales, namely the "Angle&Scale" similarity. The key features of our method are to preserve the temporal dependence, or as we explained before, the angles of $\mathbf{y}_{T+1:T+H}$, and automatically project $||\mathbf{y}_{T+1:T+H}||_2$ to each y_{T+h} to estimate its scale. We summarize these features as "angle-preserving" and "auto-scaling", where auto-scaling is important for handling data with different magnitudes [29]. One remaining challenge is optimizing the likelihood function due to the Bessel function, an essential part of VMF distribution and commonly leading to underflow problems [20]. Without the sacrifice of accuracy, we estimate and alternatively optimize the upper bound of the Bessel function.

Our contributions are three-fold: (1) We propose a probabilistic forecasting model (VMFTransformer) based on Transformer and VMF distribution, which captures the temporal dependence of multi-horizon targets. We also demonstrate our model's performance is state-of-the-art on real-world datasets; (2) We design a novel similarity measurement termed "Angle&Scale" similarity for the attention module; (3) We present a more efficient optimization method for the Bessel function in the VMF distribution without the sacrifice of accuracy.

2 Related work

Time Series Forecasting. Recent time series forecasting models based on deep learning (e.g., recurrent and convolutional neural networks) [29, 27, 36] provide a data-driven manner to deal with time series forecasting tasks and achieve great accuracy in most application fields. Due to complex dependencies over time of recurrent networks and the limits of convolutional filters, these methods have difficulties in modeling long-term and complex relations in the time series data. Recently, Transformers [33] based on self-attention mechanism [12] show promising performance in time series forecasting [22, 38, 43, 19]. Considering the dependencies of each time point in a sequence, Transformer-based methods [22] are proposed by assigning different importance to the different time points. Progresses have been made in reducing the computation complexity of the self-attention and enhancing the capacity of information extraction of the Encoder-Decoder structure [43, 38, 19]. In addition, Matrix factorization methods [40] and Bayesian methods that share information via hierarchical priors [4] are used to learn multiple related time series by leveraging hierarchical structure [16].

Von Mises-Fisher (VMF) distribution. The von Mises Fisher Distribution (VMF) is an important isotropic distribution for directional data that have direction as well as magnitudes, such as gene expression data, wind current directions, or measurements taken from compasses [11]. The von Mises–Fisher distribution is a probability distribution on directions in $\mathbb{R}^p$. It can be regarded as a distribution on the $(p-1)$-sphere of unit radius, which is on the surface of the D-ball of unit radius. If $p = 2$, the distribution reduces to the von Mises distribution on the circle. Recently, it has been successfully used in numerous machine learning tasks, such as unsupervised learning [1, 14], supervised learning [30], contrastive learning [35], natural language processing [20], computer vision [15, 42], and so on. Our work is the first to introduce the von Mises-Fisher distribution to the time series forecasting task.

Direction of Time Series Slice. Suppose $\mathbf{y}_{T+1:T+H}$ is a slice of a time series, the direction of $\mathbf{y}_{T+1:T+H}$, which is an H-dimension unit-vector, can be defined by the cosine of its angles relative to the orthonormal basis $\{e_h\}_{h=1}^{H}$[1] of the vector space. Technically, the direction of $\mathbf{y}_{T+1:T+H}$ is

$$\frac{\mathbf{y}_{T+1:T+H}}{||\mathbf{y}_{T+1:T+H}||_2} = \left(\frac{\langle \mathbf{y}_{T+1:T+H}, e_1 \rangle}{||\mathbf{y}_{T+1:T+H}||_2}, \ldots, \frac{\langle \mathbf{y}_{T+1:T+H}, e_H \rangle}{||\mathbf{y}_{T+1:T+H}||_2} \right)^T, \tag{1}$$

where $\langle , \rangle$ denotes the inner product and $||\mathbf{y}_{T+1:T+H}||_2 = \sqrt{\sum_{h=1}^{H} y_{T+h}^2}$ denotes the Euclidean norm of $\mathbf{y}_{T+1:T+H}$.

3 Methodology

The core idea of this work is to consider a multi-horizon target as a vector in a high-dimensional space, where its direction or angle relative to the orthonormal basis characterizes the dependence structure, and its length is determined by the scale of the targets at each horizon. Therefore, performing probabilistic forecasting requires first evaluating the probability distribution of directions. For this purpose, we appeal to the VMF distribution. Recall that the VMF distribution only applies to vectors distributed on a unit sphere; we, therefore, adopt a probability distribution on the length of the vector representing the multi-horizon target. We choose the truncated normal distribution for vector length, which turns out to be a prior distribution, and provide the detail in Section 3.2. We visualize this idea in Figure 1.

The rest of this section is organized into 4 subsections. We first formally define the problem of forecasting in Section 3.1, and then derive the loss function for model training in Section 3.2. Next, we introduce the model structure in Section 3.3. Finally, we present a random sampling method for performing probabilistic forecasts in section 3.4.

3.1 Problem Statement

We are given a dataset of N time series where the i-th time series consisting of tuples $(y_{i,t}, \mathbf{x}_{i,t}) \in \mathbb{R} \times \mathbb{R}^d, t = 0, \ldots, T$, where $y_{i,t}$ is the observation at time t and $\mathbf{x}_{i,t}$ are corresponding input covariates. Given a forecast horizon $H \in \mathbb{N}^+$, the goal is to estimate the joint predictive distribution of future observations

$$p(\{\mathbf{y}_{i,T+1:T+H}\}_{i=1}^{N} | \mathcal{D}),$$

where $\mathbf{y}_{i,t+1:t+H} = (y_{t+1}^i, y_{t+2}^i, \ldots, y_{t+H}^i)^T$, and $\mathcal{D} = \left\{ \{y_{i,t}\}_{t=0}^{T}, \{\mathbf{x}_{i,t}\}_{t=0}^{T+H} \right\}_{i=0}^{N}$ represents the history of the time series and all known input co-variates.

3.2 Objective Function

This section derives the loss function based on negative log-likelihood (NLL).

VMF Distribution. The VMF distribution is a probability distribution on the surface of a unit-sphere. For a d dimensional random unit vector $\mathbf{y} = (y_1, \ldots, y_d)^T$ ($||\mathbf{y}||_2 = 1$), the probability density function of VMF distribution is defined as

$$p(\mathbf{y}; \boldsymbol{\mu}, \kappa) = C_d(\kappa) \exp(\kappa \boldsymbol{\mu}^T \times \mathbf{y}), \tag{2}$$

where $\boldsymbol{\mu}$ denotes the mean direction ($||\boldsymbol{\mu}||_2 = 1$), and κ denotes the concentration parameter. In other words, $\boldsymbol{\mu}$ locates the most likely direction of $\mathbf{y} = (y_1, \ldots, y_d)^T$, and κ controls the divergence of $\mathbf{y} = (y_1, \ldots, y_d)^T$ from $\boldsymbol{\mu}$. The greater the value of κ, the stronger

[1] $e_h = (0, ..., 1, ..., 0)$, the h-th element is 1 while the rest are 0.

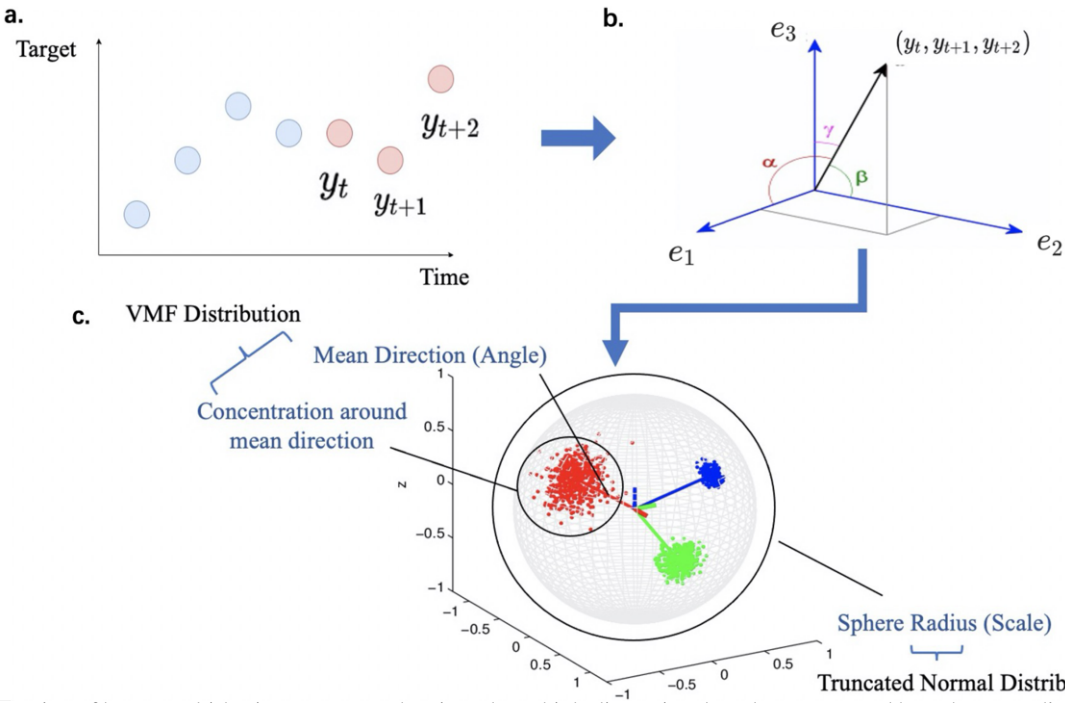

Figure 1: Illustration of how a multi-horizon target can be viewed as a high-dimensional random vector and how the VMF distribution is applied to describe its distribution. As shown in Figure 1.a, the example shows when performing a 3-step forecasting, the target can be packed together into a 3-dimensional vector (y_t, y_{t+1}, y_{t+2}) where its direction is a unit-vector, i.e. $(\cos\alpha, \cos\beta, \cos\gamma)$, determined by its angles relative to the orthonormal basis $\{e_1, e_2, e_3\}$ in Figure 1.b. When there exists dependency among y_t, y_{t+1}, and y_{t+2}, the random vector (y_t, y_{t+1}, y_{t+2}) is likely to be distributed around a certain direction, described by the VMF distribution for random vectors on unit-sphere in Figure 1.c. The scale or length of (y_t, y_{t+1}, y_{t+2}) is equivalent to sphere radius, whose distribution is described by the Truncated Normal Distribution in Figure 1.c. A forecast can be made by multiplying a direction generated from the estimated VMF distribution with a length generated from the estimated Truncated Normal Distribution.

concentration of $\mathbf{y} = (y_1, \ldots, y_d)^T$ around $\boldsymbol{\mu}$. The normalization constant $C_d(\kappa)$ in Equation (2) is defined as

$$C_d(\kappa) = \frac{\kappa^{d/2-1}}{(2\pi^{d/2}) I_{d/2-1}(\kappa)},$$

where $I_{d/2-1}(\kappa)$ [2] is the modified Bessel function of the first kind.

Negative Log-Likelihood. For a slice of the i-th time series $\mathbf{y}_{i,t+1:t+H}$, recall that by Equation (1), its direction is $\frac{\mathbf{y}_{i,t+1:t+H}}{\sigma_{i,t}}$, where $\sigma_{i,t} = ||\mathbf{y}_{i,t+1:t+H}||_2$. We define the distribution of the direction as a VMF distribution, which is

$$p\left(\frac{\mathbf{y}_{i,t+1:t+H}}{\sigma_{i,t}}; \boldsymbol{\mu}_{i,t}, \kappa_{i,t}\right) = C_H(\kappa_{i,t}) \exp\left(\frac{\kappa_{i,t}\boldsymbol{\mu}_{i,t}^T \mathbf{y}_{i,t+1:t+H}}{\sigma_{i,t}}\right), \tag{3}$$

where $\sigma_{i,t}$ is a random variable. We introduce a prior distribution $p(\sigma_{i,t})$ on $\sigma_{i,t}$, and obtain a tractable likelihood function of $\mathbf{y}_{i,t+1:t+H}$, which is

$$p(\mathbf{y}_{i,t+1:t+H}) = p\left(\mathbf{y}_{i,t+1:t+H} | \sigma_{i,t}; \boldsymbol{\mu}_{i,t}, \kappa_{i,t}\right) p(\sigma_{i,t}), \tag{4}$$

Let $\mathcal{T}$ denote the set of all forecast times selected for generating training data, by taking the logarithm of Equation (4) and multiplying it with -1, we have derived the NLL function as

$$\begin{aligned}\mathcal{L} &= -\sum_{i=1}^{N}\sum_{t\in\mathcal{T}} \log p\left(\mathbf{y}_{i,t+1:t+H} | \sigma_{i,t}; \boldsymbol{\mu_{i,t}}, \kappa_{i,t}\right) - \sum_{i=1}^{N}\sum_{t\in\mathcal{T}} \log p(\sigma_{i,t}) \\ &= \mathcal{L}_{VMF} + \mathcal{L}_{Prior\ on\ \sigma}. \end{aligned} \tag{5}$$

[2] $I_n(z)$ is defined as $(z/2)^n \sum_{k=0}^{+\infty} \frac{(z/2)^{2k}}{k!\Gamma(n+k+1)}$, where $\Gamma(\cdot)$ is the Gamma function.

Remark 3.1. *In this work, the horizon H should be at least 2, otherwise, the forecasting target is a scalar and hence has no direction.*

Bound of Bessel Function. Specifically, omitting the constant terms, $\mathcal{L}_{VMF}$ is

$$\begin{aligned}\mathcal{L}_{VMF} = &\underbrace{-\sum_{i=1}^{N}\sum_{t\in\mathcal{T}} \frac{\kappa_{i,t}\boldsymbol{\mu}_{i,t}^T \mathbf{y}_{i,t+1:t+H}}{\sigma_{i,t}}}_{loss\ of\ angle} \\ &\underbrace{-\sum_{i=1}^{N}\sum_{t\in\mathcal{T}} (\frac{H}{2}-1)\log\kappa_{i,t} + \sum_{i=1}^{N}\sum_{t\in\mathcal{T}} \log I_{\frac{H}{2}-1}(\kappa_{i,t})}_{penalty\ on\ \kappa_{i,t}}, \end{aligned} \tag{6}$$

where the first part computes the cosine between the forecast average direction $\boldsymbol{\mu}_{i,t}$ and the direction of $\mathbf{y}_{i,t+1:t+H}$, and the second part penalizes the magnitude of the forecast concentration parameter $\kappa_{i,t}$.

The NLL $\mathcal{L}_{VMF}$ is not directly differentiable because the Bessel function $I_{H/2-1}(\cdot)$ cannot be written in a closed form [20, 10]. In addition, optimizing the Bessel function may cause an underflow problem when d is large or x is small [20]. Therefore, we alternatively minimize the upper bound of $\sum_{i=1}^{N}\sum_{t\in\mathcal{T}} \log I_{H/2-1}(\kappa_{i,t})$.

We evaluate both the lower and upper bounds of the function $\log I_d(x) (d \in \mathbb{R} \text{ and } d \geq 0,\ x \in \mathbb{R})$, and summarize the result in Proposition 3.1.

We also visualize the difference between the upper and lower bounds in Figure 3, which vividly illustrates the range of the approximation error.

Proposition 3.1. *Let $I_d(x)$ be the modified Bessel function of the first kind, and $m = d - \lfloor d \rfloor$, then*

$$\log\left(I_d(x)\right) < \log\left(I_m(\kappa)\right) + \sum_{v=1}^{\lfloor d \rfloor} \log \frac{\kappa}{v + m - 1 + \sqrt{(v+m+1)^2 + \kappa^2}},$$

and

$$\log\left(I_d(x)\right) > \log\left(I_m(\kappa)\right) + \sum_{v=1}^{\lfloor d \rfloor} \log \frac{\kappa}{v+m-\frac{1}{2}+\sqrt{(v+m+\frac{1}{2})^2+\kappa^2)}}.$$

Proof. Apparently,

$$\begin{aligned}\log\left(I_d(x)\right) &= \log\left(I_m(x)\prod_{v=1}^{\lfloor d \rfloor}\frac{I_{m+v}(x)}{I_{m+v-1}(x)}\right)\\ &= \log\left(I_m(x)\right) + \sum_{v=1}^{\lfloor d \rfloor}\log\left(\frac{I_{m+v}(x)}{I_{m+v-1}(x)}\right).\end{aligned}$$

By the Theorem 4 of Diego et al. 2016 [28], for any $d \geq 0$

$$\frac{x}{d-\frac{1}{2}+\sqrt{(d+\frac{1}{2})^2+x^2)}} < \frac{I_d(x)}{I_{d-1}(x)} < \frac{x}{d-1+\sqrt{(d+1)^2+x^2)}}.$$

We complete the proof. □

By Proposition 3, we use the following approximation of $\sum_{i=1}^{N}\sum_{t\in\mathcal{T}}\log I_{H/2-1}(\kappa_{i,t})$ to compute $\mathcal{L}_{VMF}$ for model training,

$$\sum_{i=1}^{N}\sum_{t\in\mathcal{T}}\left[\log\left(I_m(\kappa_{i,t})\right) + \sum_{v=1}^{\lfloor \frac{H}{2}-1 \rfloor}\log\frac{\kappa_{i,t}}{v+m-1+\sqrt{(v+m+1)^2+\kappa_{i,t}^2)}}\right], \tag{7}$$

where m is 0.5 when H is odd and 0 the otherwise.

Prior on $\boldsymbol{\sigma}$. The support of the prior distribution of $\sigma_{i,t}$ should be non-negative. We choose the truncated normal distribution [3] as prior distribution for $\sigma_{i,t}$ to ensure non-negativity, and substituting it into $\mathcal{L}_{Prior\,on\,\sigma}$. We show next that one advantage of choosing truncated normal distribution is to reduce $\mathcal{L}_{Prior\,on\,\sigma}$ to a mean squared loss with penalty.

The truncated normal distribution is an extension of the normal distribution, which compress the range of a random variable from $(-\infty,+\infty)$ into an open interval $(a,b)\,(-\infty \leq a < b \leq +\infty)$. A truncated normal distribution is determined by four parameters m,γ,a,b. The parameters m,γ denote the location and shape parameters, respectively, while a,b denote the lower and upper bound of the random variable. In our case, $a=0$ and $b=+\infty$. Technically,

$$p(\sigma_{i,t};m_{i,t},\gamma_{i,t}) = \frac{\exp\left[-\frac{(\sigma_{i,t}-m_{i,t})^2}{2\gamma_{i,t}^2}\right]}{\sqrt{2\pi}\gamma_{i,t}\left[1-\Phi\left(-\frac{m_{i,t}}{\gamma_{i,t}}\right)\right]}, (\sigma_{i,t}\geq 0) \tag{8}$$

where $\Phi(\cdot)$ denotes the cumulative distribution function of the standard normal distribution. Based on Equation (8) and omitting the constant terms,

$$\mathcal{L}_{Prior\,on\,\sigma} = \underbrace{\sum_{i=1}^{N}\sum_{t\in\mathcal{T}}\frac{(\sigma_{i,t}-m_{i,t})^2}{\gamma_{i,t}}}_{meaan\,squared\,loss\,of\,scale} + \underbrace{\sum_{i=1}^{N}\sum_{t\in\mathcal{T}}\left[\gamma_{i,t}+\Phi\left(\frac{m_{i,t}}{\gamma_{i,t}}\right)\right]}_{penalty\,on\,m_{i,t},\gamma_{i,t}}, \tag{9}$$

where the essential part of $\mathcal{L}_{Prior\,on\,\sigma}$ for being used as a loss function, is the computation of the squared difference between the forecast average vector length $m_{i,t}$ and $\sigma_{i,t}$.

Training Loss. A forecasting model with model parameters θ is a multi-output function

$$f_\theta(\mathbf{z}_{i,t}) = (f_\theta^\mu(\mathbf{z}_{i,t}), f_\theta^\kappa(\mathbf{z}_{i,t}), f_\theta^m(\mathbf{z}_{i,t}), f_\theta^\gamma(\mathbf{z}_{i,t})),$$

where $\mathbf{z}_{i,t}$ represents the model input for forecasting $\mathbf{y}_{i,t+1:t+H}$, and $f_\theta^\mu(\mathbf{z}_{i,t})$, $f_\theta^\kappa(\mathbf{z}_{i,t})$, $f_\theta^m(\mathbf{z}_{i,t})$, and $f_\theta^\gamma(\mathbf{z}_{i,t})$ are model outputs corresponding to $\boldsymbol{\mu}_{i,t}$, $\kappa_{i,t}$, $m_{i,t}$ and $\gamma_{i,t}$ respectively. Based on Equation (7), the ultimate training loss is

$$\mathcal{L}(\theta) = \overline{\mathcal{L}_{VMF}}(\theta) + \mathcal{L}_{Prior\,on\,\sigma}(\theta), \tag{10}$$

where

$$\begin{aligned}&\overline{\mathcal{L}_{VMF}}(\theta) \qquad\qquad (11)\\ &= -\sum_{i=1}^{N}\sum_{t\in\mathcal{T}}\frac{f_\theta^\kappa(\mathbf{z}_{i,t})f_\theta^\mu(\mathbf{z}_{i,t})^T\mathbf{y}_{i,t+1:t+H}}{||\mathbf{y}_{i,t+1:t+H}||_2}\\ &-\sum_{i=1}^{N}\sum_{t\in\mathcal{T}}(\frac{H}{2}-1)\log f_\theta^\kappa(\mathbf{z}_{i,t}) + \sum_{i=1}^{N}\sum_{t\in\mathcal{T}}\log I_m(f_\theta^\kappa(\mathbf{z}_{i,t}))\\ &+\sum_{i=1}^{N}\sum_{t\in\mathcal{T}}\sum_{v=1}^{\lfloor \frac{H}{2}-1 \rfloor}\log\frac{f_\theta^\kappa(\mathbf{z}_{i,t})}{v+m-1+\sqrt{(v+m+1)^2+f_\theta^\kappa(\mathbf{z}_{i,t})^2)}}, \qquad (12)\end{aligned}$$

and

$$\begin{aligned}&\mathcal{L}_{Prior\,on\,\sigma}(\theta)\\ &=\sum_{i=1}^{N}\sum_{t\in\mathcal{T}}\frac{(||\mathbf{y}_{i,t+1:t+H}||_2 - f_\theta^m(\mathbf{z}_{i,t}))^2}{f_\theta^\gamma(\mathbf{z}_{i,t})}\\ &+\sum_{i=1}^{N}\sum_{t\in\mathcal{T}}\left[f_\theta^\gamma(\mathbf{z}_{i,t}) + \Phi\left(\frac{f_\theta^m(\mathbf{z}_{i,t})}{f_\theta^\gamma(\mathbf{z}_{i,t})}\right)\right]. \qquad (13)\end{aligned}$$

3.3 Model

The seq2seq architecture is based on Transformer [33], which adopts an encoder-decoder structure with a multi-head self-attention mechanism to capture both long- and short-term dependencies. This encoder-decoder structure has been demonstrated a significant contribution to time series forecasting [38, 23, 22]. We visualize our model architecture in Figure 2.

Encoder. We adopt a similar structure of encoder as the vanilla Transformer [33], where all attention modules are modified to better suit $\mathcal{L}(\theta)$, and detailedly explained later. The encoder focuses on the history of time series, of which time-index features are processed via implicit neural representation (INR) as suggested in [37]. Different from the vanilla Transformer, our encoder is composed of a stack of one input layer as the first layer, and N identical layer namely "encoding layer". Each layer has two sub-layers, of which the first is a multi-head attention mechanism, and the second is a position-wise fully connected feed-forward network. We also employ a residual connection around each sub-layer followed by layer normalization. Both the time series and time-index features are directly fed into the attention layer of the input layer, while only the history of the time series is used for the residual connection around the attention layer. Sequentially, the output of the former layer and the history of the time series are fed into each of the encoding layers (Figure 2).

Remark 3.2. *This design enhances the impact of the original time series to capture its direction as a random vector better, while the attention layers extract significant time-index features.*

Decoder. Different from the encoder, the decoder structure is very similar to the vanilla Transformer decoder [33], and is composed of a stack of N identical layers but also with the modified multi-head self-attention mechanism. Each of the identical layers consists of 3 sub-layers. The inputs of the decoder consist of the time-index features of the future target processed via INR and the output of the

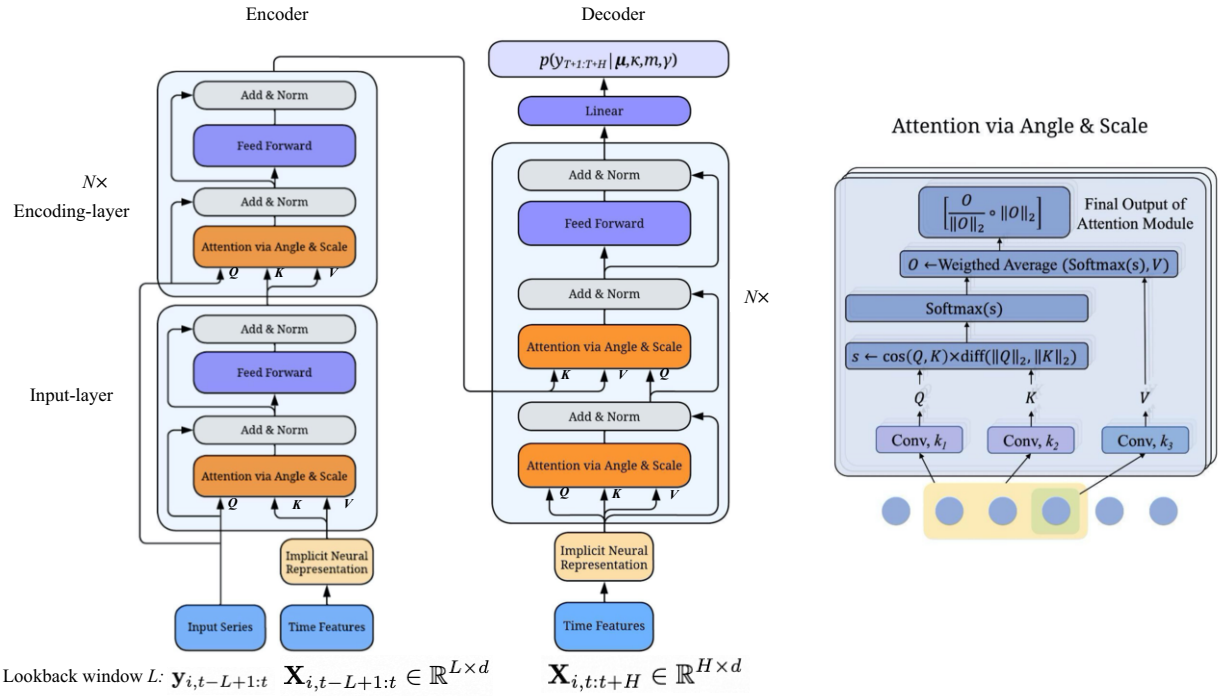

Figure 2: Model architecture. The encoder is composed of an input layer and N identical encoding layer, while the decoder is composed of N identical layer (Left panel). The attention mechanism first passes the input sequence into convolution kernels for computing queries ($\boldsymbol{Q}$), keys ($\boldsymbol{K}$), and values ($\boldsymbol{V}$), where "Conv, k_1", "Conv, k_2", and "Conv, k_3" mean convolution of kernel size k_1, k_2, and k_3 with stride 1 respectively. The similarity between $\boldsymbol{Q}$ and $\boldsymbol{K}$ is measured by multiplying the co- sine of the angle by the difference of their norms.

encoder which is fed to the third sub-layer of each decoder (Figure 2). In addition, we do not conduct masking on the decoder as we are performing direct forecasting and hence have no concern about using future outputs [33].

Attention Mechanism. An attention function is to map a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors [33]. To compute the attention function on a set of queries simultaneously, all queries are usually packed into a query matrix $\boldsymbol{Q}$. The keys and values are also packed into matrices $\boldsymbol{K}$ and $\boldsymbol{V}$ respectively [33].

The conventional attention function measures the similarities between queries and keys, which are row vectors of $\boldsymbol{Q}$ and $\boldsymbol{K}$ respectively, by a 'dot-product' operation and feeds the similarities into a softmax function to normalize the similarities summing to 1 [33]. Afterward, the output of the attention function is computed as the weighted average of the values, where the weights are the outputs of the softmax function. We adopt the **multi-head convolutional self-attention mechanism**, which has effectively enhanced awareness of local context, e.g. local shapes of time series [22].

We modify the similarity measurement between queries and keys to better suit our objective function. We measure the similarity between a query vector and a key vector by computing the cosine of the angle between them and multiplying it by the difference between the lengths. For the rest of this article, we refer to this similarity measurement as the "Angle&Scale" similarity. The softmax function is then applied to obtain the weights of the values.

Technically, let L be the input length or lookback window [37], the **encoder attention mechanism** of the input layer transforms $\mathbf{y}_{i,t-L+1:t}$ via a convolution kernel into a query matrix $\boldsymbol{Q} = \mathbf{y}_{i,t-L+1:t} * \boldsymbol{W}^Q$ ($\boldsymbol{W}^Q \in \mathbb{R}^{k\times 1\times d_q}, \boldsymbol{Q} \in \mathbb{R}^{L\times d_q}$), and transforms time-index features $\mathbf{X}_{i,t-L+1:t} = (\mathbf{x}_{i,t-L+1}, \ldots, \mathbf{x}_{i,t})$ ($\mathbf{X}_{i,t-L+1:t} \in \mathbb{R}^{L\times d}$) into key matrix $\boldsymbol{K} = \mathbf{X}_{i,t-L+1:t} * \boldsymbol{W}^K$ ($\boldsymbol{W}^K \in \mathbb{R}^{k\times d\times d_q}, \boldsymbol{K} \in \mathbb{R}^{L\times d_q}$) and value matrix $\boldsymbol{V} = \mathbf{X}_{i,t-L+1:t} * \boldsymbol{W}^V$ ($\boldsymbol{W}^K \in \mathbb{R}^{k\times d\times d_v}, \boldsymbol{V} \in \mathbb{R}^{L\times d_v}$) respectively, where $*$ denotes the convolution operator. The first two dimensions of convolution kernels $\boldsymbol{W}^Q$, $\boldsymbol{W}^K$, and $\boldsymbol{W}^V$ indicate the kernel shape, while the last dimension indicates the number of kernels. All inputs are convoluted with the first two dimensions of convolution kernels to generate a scalar output and combined together along the last dimension.

Let $\mathbf{q}_i$ and $\mathbf{k}_j$ be the i-th row vector of $\boldsymbol{Q}$ and the j-th row vector of $\boldsymbol{K}$ respectively. The similarity between $\mathbf{q}_i$ and $\mathbf{k}_j$ is defined as

$$s_{i,j} = \frac{\mathbf{q}_i^T \mathbf{k}_j}{||\mathbf{q}_i||_2 ||\mathbf{k}_j||_2} \times \exp\left[-\left(||\mathbf{q}_i||_2 - ||\mathbf{k}_j||_2\right)^2\right]. \tag{14}$$

It should be noted that in Equation (14), the difference between $||\mathbf{q}_i||_2$ and $||\mathbf{k}_j||_2$ is passed into a Gaussian kernel with a radial parameter of 1. This operation is to ensure the difference ranging from 0 to 1, such that the magnitude of the cosine and difference would be at the same level.

Each row vector of the output of the attention layer $\boldsymbol{O} = [\mathbf{o}_1, \ldots, \mathbf{o}_t]^T \in \mathbb{R}^{L\times d_v}$, is defined as the normalized weighted average of the row vectors of the value matrix $\boldsymbol{V}$, concatenating its norm. By Equation (15), we formalize this definition as

$$\mathbf{o}_i = \left[\frac{\sum_{j=1}^t \text{softmax}(s_{i,j})\mathbf{v}_j^T}{||\sum_{j=1}^t \text{softmax}(s_{i,j})\mathbf{v}_j^T||_2} \circ ||\sum_{j=1}^t \text{softmax}(s_{i,j})\mathbf{v}_j^T||_2\right]. \tag{15}$$

It should be noted that all subsequent encoding layers take the outputs of the former layer to compute $\boldsymbol{K}$ and $\boldsymbol{V}$, but still use $\mathbf{y}_{i,t-L+1:t}$ to compute $\boldsymbol{Q}$. The attention layers of the decoder also perform the Angle&Scale similarity but use the same input to generate $\boldsymbol{Q}$, $\boldsymbol{K}$ and $\boldsymbol{V}$ via convolution kernels as shown in Figure 2.

Table 1: Comparison of performance of VMFTransformer with competitive models on three public datasets. The best results are marked in bold (lower is better).

		Electricity				Solar				Traffic			
H	Models	MAE	MSE	q-50-loss	q-90-loss	MAE	MSE	q-50-loss	q-90-loss	MAE	MSE	q-50-loss	q-90-loss
24	PatchTST	0.0610	0.0081	0.1840	0.2291	**0.1938**	**0.0936**	0.5611	0.5678	0.2746	0.2251	0.4690	0.4367
	DLinear	0.1069	0.0309	0.1161	0.1180	0.2110	0.1336	0.2319	0.2129	0.2317	0.2720	0.3418	0.3462
	DeepAR	0.0167	0.0187	0.0734	0.0580	0.2619	0.3243	0.4626	0.3097	0.1667	0.2042	0.1526	0.1082
	SimpleFeedForward	0.0152	0.0113	0.0784	0.0433	0.3023	0.4213	0.5341	0.3591	0.2751	0.3775	0.2518	0.2011
	TFT	0.0251	0.0363	0.1107	0.0588	0.2748	0.3189	0.4855	0.2471	0.2164	0.2912	0.1981	0.1478
	Transformer	0.0138	0.0144	0.0789	0.0500	0.3031	0.4502	0.5354	0.3502	0.1796	0.2050	0.1644	0.1087
	Autoformer	0.1236	0.0252	0.1224	0.0653	0.2106	0.1684	0.2233	0.2112	0.2612	0.1541	0.2156	0.1454
	Informer	0.0844	0.0144	0.2167	0.1816	0.2607	0.1823	0.3456	0.2480	0.4620	0.6446	0.6055	0.6649
	VMFTransformer	**0.0116**	**0.0077**	**0.0722**	**0.0427**	0.2092	0.1333	**0.2125**	**0.1371**	**0.1567**	**0.1185**	**0.1490**	**0.0935**
168	PatchTST	0.0957	**0.0169**	0.1171	0.1189	**0.1836**	**0.0942**	0.3462	0.3976	**0.0898**	**0.0179**	0.3041	0.3019
	DLinear	0.1559	0.0595	0.1750	0.1874	0.2129	0.1250	0.2458	0.2326	0.2633	0.3096	0.3879	0.3968
	DeepAR	0.0224	0.0300	0.1463	0.0862	0.3076	0.3806	0.5613	0.2349	0.1712	0.2115	0.1553	**0.1012**
	SimpleFeedForward	0.0210	0.0281	0.1397	0.0730	0.3228	0.4122	0.5890	0.2557	0.2590	0.2901	0.2348	0.1360
	TFT	0.0286	0.0468	0.2242	0.1122	0.3335	0.5125	0.6085	0.2030	0.1670	0.1952	0.1514	0.1019
	Transformer	0.0198	0.0295	0.1172	0.0670	0.3463	0.4903	0.6318	0.3115	0.1918	0.2081	0.1739	0.1063
	Autoformer	0.1197	0.0241	0.1342	0.0790	0.2784	0.1843	0.2354	0.1588	0.2231	0.1746	0.2559	0.1948
	Informer	0.1460	0.0335	0.3753	0.2440	0.2274	0.1616	0.5434	0.2960	0.5549	0.8068	0.7194	0.6800
	VMFTransformer	**0.0128**	0.0204	**0.0839**	**0.0597**	0.2136	0.1390	**0.2318**	**0.1473**	0.1567	0.1185	**0.1225**	0.1099
720	PatchTST	0.0843	0.0623	0.4222	0.2998	0.3112	0.2975	0.5221	0.2723	0.5836	0.4726	0.7957	0.4704
	DLinear	0.2529	0.1401	0.2802	0.3319	0.2249	**0.1322**	**0.2598**	0.2319	0.4537	0.5847	0.6692	0.7092
	DeepAR	0.0323	0.0897	0.1805	0.1762	0.2220	0.2004	0.3543	0.1451	0.2958	0.4112	0.2724	0.1883
	SimpleFeedForward	0.0239	0.0491	0.1336	0.0854	0.2342	0.2226	0.3738	0.1603	0.2890	0.3501	0.2661	0.1579
	TFT	0.0313	0.0809	0.1748	0.0975	0.2027	0.2141	0.3234	**0.1084**	0.2069	0.3467	0.1906	0.1266
	Transformer	0.0173	0.0321	0.1246	0.0829	0.2170	0.2536	0.3463	0.1600	**0.2047**	0.3921	**0.1885**	**0.1226**
	Autoformer	0.3857	0.2345	0.3384	0.3138	0.4082	0.2574	0.6827	0.5837	0.3897	0.3277	0.3751	0.2969
	Informer	0.1698	0.0457	0.3897	0.2496	0.2647	0.2599	0.5601	0.3097	0.6165	0.8749	0.8969	0.7540
	VMFTransformer	**0.0147**	**0.0136**	**0.1028**	**0.0798**	**0.2018**	0.2198	0.2556	0.5455	0.2583	**0.3029**	0.5738	0.3917

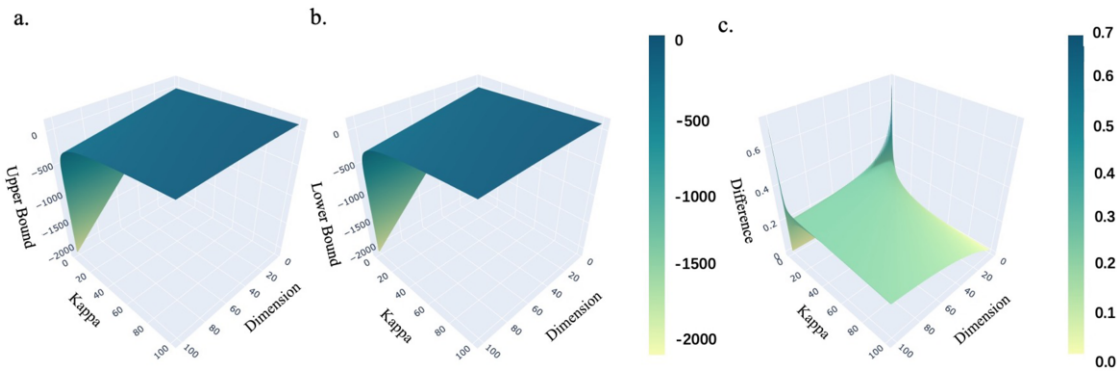

Figure 3: Upper (a) and lower (b) bounds of the Bessel Function. The difference between the upper and lower bounds is shown in (c).

Model Output. Recall that in Section 3.2, Equation (10), (11) and (13) require that there are 4 outputs corresponding to $\boldsymbol{\mu}_{i,t}$, $\kappa_{i,t}$, $m_{i,t}$ and $\gamma_{i,t}$ respectively. Let $\mathbf{h}_{i,t}$ denote the output of the decoder for forecasting $\mathbf{y}_{i,t+1:t+H}$. We apply a fully-connected layer to $\mathbf{h}_{i,t}$ to obtain $\boldsymbol{\mu}$,

$$\boldsymbol{\mu}_{i,t} = \frac{\boldsymbol{W}_{\boldsymbol{\mu}}\mathbf{h}_{i,t} + \mathbf{b}_{\boldsymbol{\mu}}}{||\boldsymbol{W}_{\boldsymbol{\mu}}\mathbf{h}_{i,t} + \mathbf{b}_{\boldsymbol{\mu}}||_2}.$$

For $\kappa_{i,t}$, $m_{i,t}$ and $\gamma_{i,t}$, we apply a fully connected layer with softplus activation to ensure positivity. Specifically,

$$\begin{aligned}\kappa_{i,t} &= \log(1+\exp(\mathbf{w}_{\kappa}\mathbf{h}_{i,t} + b_{\kappa})),\\ m_{i,t} &= \log(1+\exp(\mathbf{w}_{m}\mathbf{h}_{i,t} + b_{m})),\\ \gamma_{i,t} &= \log(1+\exp(\mathbf{w}_{\gamma}\mathbf{h}_{i,t} + b_{\gamma})).\end{aligned}$$

3.4 Prediction

At the prediction stage, there is no closed-form solution for computing the quantiles of $\mathbf{y}^i_{T+1:T+H}$. We alternatively draw random samples from the probability density function of $\mathbf{y}^i_{T+1:T+H}$ and estimate each quantile empirically.

Random sampling. We first sample the scale parameter σ according to Equation (8) and next, we draw a sample of $\mathbf{y}^i_{T+1:T+H}$ according to Equation (3). Sampling $\mathbf{y}^i_{T+1:T+H}$ via Equation (3) is equivalent to first sampling a unit vector from a VMF distribution and multiplying it by σ. An H-dimensional random unit vector $\mathbf{y}$ subjecting to a VMF distribution with parameters $\boldsymbol{\mu}$, κ, can be decomposed as

$$\mathbf{y} = \boldsymbol{\mu} + \boldsymbol{v}\sqrt{(1-t^2)}, \tag{16}$$

where $\boldsymbol{v}$ is a uniformly distributed unit tangent at $\boldsymbol{\mu}$, and $t \in [-1, 1]$ subjects to

$$p(t) = \frac{(\frac{\kappa}{2})^{\frac{H}{2}-1}\exp(\kappa t)(1-t^2)^{\frac{H-3}{2}}}{\Gamma\left(\frac{H-1}{2}\right)\gamma\left(\frac{1}{2}\right)I_{\frac{H-1}{2}}(\kappa)} \tag{17}$$

Therefore, we respectively sample t and $\boldsymbol{v}$, and construct a sample $\mathbf{y}$ by Equation (16). More details on the decomposition defined by Equation (16) can be referred to in [24].

4 Experiment

4.1 Bessel Function Approximation Error

We evaluate the approximation error of the logarithm of the Bessel function given by Equation (7). We compute both the upper and

Table 2: Comparison of the sensitivity of the VMFTransformer to the sampling size at the prediction step. The prediction length (horizon) is set to H=24 and H=168. Sample size (S) is set to 100, 1000, 10000, and 100000. The best results are marked in bold (lower is better). The stability of performance is measured by the average (AVG) divided by the standard deviation (STD).

Samplesize		100	1000	10000	100000	AVG	STD	STD/AVG
H=24	MAE	0.2341	**0.2339**	0.2424	0.2368	0.2368	0.0040	0.0168
	MSE	0.1846	**0.1828**	0.1995	0.1924	0.1898	0.0077	0.0406
	50-loss	0.3243	**0.3238**	0.3370	0.3274	0.3281	0.0061	0.0187
	90-loss	**0.2435**	0.2667	0.2490	0.2823	0.2604	0.0177	0.0678
H=168	MAE	**0.2000**	0.2014	0.2005	0.2002	0.2005	0.0006	0.0030
	MSE	0.1303	0.1307	**0.1286**	0.1305	0.1300	0.0010	0.0077
	50-loss	**0.2830**	0.2848	0.2834	0.2833	0.2836	0.0008	0.0028
	90-loss	0.1733	**0.1724**	0.1749	0.1772	0.1744	0.0021	0.0118

Table 3: Comparison of similarity measurements of attention module. The best results are in bold (lower is better).

	Similarity	Angle&Scale	Dot-product
H=24	MAE	**0.2339**	0.2389
	MSE	0.1828	**0.1749**
	50-loss	**0.3238**	0.3317
	90-loss	**0.2667**	0.2707
H=168	MAE	**0.2014**	0.2082
	MSE	**0.1307**	0.1316
	50-loss	**0.2848**	0.2950
	90-loss	**0.1724**	0.2250

lower bounds on a grid where κ ranges from 1×10^{-7} to 100, and d ranges from 2 to 100. We present the results in Figure 3. Besides, we observe that the logarithm of the Bessel function is bounded in a range of (-2093.07, 97.77) on the grid. The minimum and maximum values are achieved at $(\kappa, d) = (1 \times 10^{-10}, 100)$ and $(\kappa, d) = (100, 2)$ respectively. Recall that the underflow problem usually appears when d is large (empirically larger than 5) and κ is small; in our case, we avoid this problem via the approximation. In summary, the difference between the upper and lower bounds ranges from 0 to 0.72, and on about 95.7% of all grid points, the difference is smaller than 0.3. Since the absolute approximation error is smaller than the difference, we subsequently conclude that the upper bound of the absolute approximation error is smaller than 0.3.

4.2 *Real-world Data Experiment*

Dataset. We evaluate the performance of VMFTransformer on three public datasets, which are electricity, solar energy, and traffic. Electricity contains hourly time series of the electricity consumption of 370 customers ranging from 2012-01-01 to 2014-08-31 [29]. Solar energy, ranging from 2006-01-01 to 2006-08-31, consists of 137 5-minute solar power time series obtained from the Monash Time Series Forecasting Repository [13]. All solar power time series are aggregated to 1-hour granularity. Traffic, also used in [29], contains the hourly measured occupancy rate, between 0 and 1, of 963 car lanes of San Francisco Bay Area freeways, ranging from 2008-01-02 to 2008-06-22. We follow the standard protocol and split all datasets into training, validation, and test sets in chronological order by the ratio of 7:1:2.

Implementation details and Baselines. Our method is trained using the ADAM optimizer [18] with an initial learning rate of 10^{-3}. The batch size is set to 64. The training process is early stopped within 10 epochs. Our method contains 2 encoder layers and 2 decoder layers. All experiments are implemented in PyTorch [26]. We include a total number of 8 baseline methods. Specifically, we select 5 state-of-the-art transformer-based models: Informer [43], Autoformer [38], Temporal Fusion Transformer (TFT) [23], the vanilla Transformer [33] [22], and PatchTST [25] one RNN based method: DeepAR [29], one recent state-of-the-art one-layer linear model: Dlinear [41], and one simple feed-forward neural network.

Main Results. We evaluate models with three prediction lengths: 24, 168, and 720, corresponding to one-day, one-week and one-month horizons, respectively. We use three metrics, i.e., mean absolute error (MAE), mean squared error (MSE), and q-risk, to evaluate the performance of different methods. The first two measure the performance for forecasting the mean value, while the last quantifies the accuracy of a quantile q of the predictive distribution [29]. We set $q = 50$ and 90 [29]. The results are presented in Table 1. VMFTransformer mostly outperforms other baseline models. VMFTransformer overall gives 46.9% MAE reduction, 52.3% MSE reduction, 40% q-50-loss reduction, and 29.4% q-90-loss reduction. Compared to the two most recent methods PatchTST and DLinear, VMFTransformer outperforms in general.

4.3 *Ablation Studies.*

We use the solar energy dataset for the ablation study. **Sensitivity to Sampling Size.** Since the prediction is conducted by random sampling, we study if the model performance is sensitive to sampling size. The sampling size (S) is set as 100, $1,000$, $10,000$, and $100,000$. In table 2, we show that the performance of VMFTransformer is relatively stable, especially when the prediction length is 168 (STD/AVG$<$ 1.2%). When the prediction length is short (24), $S = 1000$ tends to show the optimal performance. Therefore, we recommend $S = 1,000$ for practical use. **Attention Module.** We compare Angle&Scale similarity versus the original dot-product similarity for the attention module. We set the prediction length at H=24 and H=168. Table 3 shows that the Angle&Scale similarity outperforms the dot-product similarity in most cases (7 out of 8). **Time Complexity.** Our Angle&Scale similarity requires more computation than the original dot-product similarity. The popular Transformer-based methods such as Reformer, Informer, and Autoformer have a theoretical time complexity of $O(L\log L)$, while the vanilla Transformer is $O(L^2)$, where L is the encoding length. The theoretical time complexity of our VMFTransformer is still $O(L^2)$, which is the same as the vanilla Transformer.

5 Conclusion

We propose a probabilistic forecasting model termed VMFTransformer, which captures the temporal dependence of multi-horizon targets. Extensive experiments demonstrate that our model's performance is state-of-the-art on public datasets. The novel similarity measurement termed the "Angle&Scale" similarity is effective for multi-horizon time series forecasting.

References

[1] A. Banerjee, I. S. Dhillon, J. Ghosh, S. Sra, and G. Ridgeway. Clustering on the unit hypersphere using von mises-fisher distributions. *Journal of Machine Learning Research*, 6(9), 2005.

[2] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems-Volume 1*, pages 1171–1179, 2015.

[3] J. Burkardt. The truncated normal distribution. *Department of Scientific Computing Website, Florida State University*, 1:35, 2014.

[4] N. Chapados. Effective bayesian modeling of groups of related count time series. In *International conference on machine learning*, pages 1395–1403. PMLR, 2014.

[5] Z. Chen, L. Ding, Z. Chu, Y. Qi, J. Huang, and H. Wang. Monotonic neural ordinary differential equation: Time-series forecasting for cumulative data. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, pages 4523–4529, 2023.

[6] Z. Chen, L. Ding, J. Huang, Z. Chu, Q. Dai, and H. Wang. Unsupervised anomaly detection & diagnosis: A stein variational gradient descent approach. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, pages 3783–3787, 2023.

[7] G. Chevillon. Direct multi-step estimation and forecasting. *Journal of Economic Surveys*, 21(4):746–785, 2007.

[8] Z. Chu, R. Li, and S. Li. Causal interventional time series forecasting on multi-horizon and multi-series data. In *Machine Learning for Causal Inference*, pages 265–282. Springer, 2023.

[9] Z. Chu, H. Ding, G. Zeng, S. Wang, and Y. Li. Causal interventional prediction system for robust and explainable effect forecasting. *arXiv preprint arXiv:2407.19688*, 2024.

[10] T. R. Davidson, L. Falorsi, N. De Cao, T. Kipf, and J. M. Tomczak. Hyperspherical variational auto-encoders. *arXiv preprint arXiv:1804.00891*, 2018.

[11] I. S. Dhillon and S. Sra. Modeling data using directional distributions. Technical report, Citeseer, 2003.

[12] C. Fan, Y. Zhang, Y. Pan, X. Li, C. Zhang, R. Yuan, D. Wu, W. Wang, J. Pei, and H. Huang. Multi-horizon time series forecasting with temporal attention learning. In *Proceedings of the 25th ACM SIGKDD International conference on knowledge discovery & data mining*, pages 2527–2535, 2019.

[13] R. Godahewa, C. Bergmeir, G. I. Webb, R. J. Hyndman, and P. Montero-Manso. Monash time series forecasting archive. In *Neural Information Processing Systems Track on Datasets and Benchmarks*, 2021.

[14] S. Gopal and Y. Yang. Von mises-fisher clustering models. In *International Conference on Machine Learning*, pages 154–162. PMLR, 2014.

[15] M. Hasnat, J. Bohné, J. Milgram, S. Gentric, L. Chen, et al. von mises-fisher mixture model-based deep learning: Application to face verification. *arXiv preprint arXiv:1706.04264*, 2017.

[16] R. J. Hyndman, R. A. Ahmed, G. Athanasopoulos, and H. L. Shang. Optimal combination forecasts for hierarchical time series. *Computational statistics & data analysis*, 55(9):2579–2589, 2011.

[17] M. Jin, S. Wang, L. Ma, Z. Chu, J. Y. Zhang, X. Shi, P.-Y. Chen, Y. Liang, Y.-F. Li, S. Pan, et al. Time-llm: Time series forecasting by reprogramming large language models. *arXiv preprint arXiv:2310.01728*, 2023.

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

[19] N. Kitaev, Ł. Kaiser, and A. Levskaya. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*, 2020.

[20] S. Kumar and Y. Tsvetkov. Von mises-fisher loss for training sequence to sequence models with continuous outputs. In *International Conference on Learning Representations*, 2018.

[21] A. Lamb, A. Goyal, Y. Zhang, S. Zhang, A. Courville, and Y. Bengio. Professor forcing: A new algorithm for training recurrent networks, 2016.

[22] S. Li, X. Jin, Y. Xuan, X. Zhou, W. Chen, Y.-X. Wang, and X. Yan. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting, 2020.

[23] B. Lim, S. Ö. Arık, N. Loeff, and T. Pfister. Temporal fusion transformers for interpretable multi-horizon time series forecasting. *International Journal of Forecasting*, 2021.

[24] K. V. Mardia and P. E. Jupp. *Directional statistics*, volume 494. John Wiley & Sons, 2009.

[25] Y. Nie, N. H. Nguyen, P. Sinthong, and J. Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers, 2023.

[26] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[27] S. S. Rangapuram, M. W. Seeger, J. Gasthaus, L. Stella, Y. Wang, and T. Januschowski. Deep state space models for time series forecasting. *Advances in neural information processing systems*, 31:7785–7794, 2018.

[28] D. Ruiz-Antolín and J. Segura. A new type of sharp bounds for ratios of modified bessel functions. *arXiv: Classical Analysis and ODEs*, 2016.

[29] D. Salinas, V. Flunkert, J. Gasthaus, and T. Januschowski. DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, 2019. ISSN 0169-2070.

[30] T. R. Scott, A. C. Gallagher, and M. C. Mozer. von mises-fisher loss: An exploration of embedding geometries for supervised learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10612–10622, 2021.

[31] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[32] S. B. Taieb and A. F. Atiya. A bias and variance analysis for multistep-ahead time series forecasting. *IEEE Transactions on Neural Networks and Learning Systems*, 27(1):62–76, 2016. doi: 10.1109/TNNLS.2015.2411629.

[33] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin. Attention is all you need. In *NIPS*. 2017.

[34] S. Wang, Z. Chu, Y. Sun, Y. Liu, Y. Guo, Y. Chen, H. Jian, L. Ma, X. Lu, and J. Zhou. Multiscale representation enhanced temporal flow fusion model for long-term workload forecasting. *arXiv preprint arXiv:2407.19697*, 2024.

[35] T. Wang and P. Isola. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *International Conference on Machine Learning*, pages 9929–9939. PMLR, 2020.

[36] R. Wen, K. Torkkola, B. Narayanaswamy, and D. Madeka. A multi-horizon quantile recurrent forecaster. *arXiv preprint arXiv:1711.11053*, 2017.

[37] G. Woo, C. Liu, D. Sahoo, A. Kumar, and S. Hoi. Learning deep time-index models for time series forecasting, 2023.

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

[39] S. Xue, Y. Wang, Z. Chu, X. Shi, C. Jiang, H. Hao, G. Jiang, X. Feng, J. Zhang, and J. Zhou. Prompt-augmented temporal point process for streaming event sequence. *Advances in Neural Information Processing Systems*, 36:18885–18905, 2023.

[40] H.-F. Yu, N. Rao, and I. S. Dhillon. Temporal regularized matrix factorization for high-dimensional time series prediction. In *NIPS*. 2016.

[41] A. Zeng, M. Chen, L. Zhang, and Q. Xu. Are transformers effective for time series forecasting?, 2022.

[42] L. Zhang, T. Dunn, J. Marshall, B. Olveczky, and S. Linderman. Animal pose estimation from video data with a hierarchical von mises-fisher-gaussian model. In *International Conference on Artificial Intelligence and Statistics*, pages 2800–2808. PMLR, 2021.

[43] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, and W. Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 11106–11115, 2021.

[44] Y. Zhou, Z. Chu, Y. Ruan, G. Jin, Y. Huang, and S. Li. ptse: a multi-model ensemble method for probabilistic time series forecasting. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*, pages 4684–4692, 2023.