Multivariate Quantile Function Forecaster

Kelvin Kan[1,†] **François-Xavier Aubet**[2] **Tim Januschowski**[3,†]
Youngsuk Park[2] **Konstantinos Benidis**[2] **Lars Ruthotto**[1]
Jan Gasthaus[2]

[1]Emory University [2]Amazon Research [3]Zalando SE

Abstract

We propose Multivariate Quantile Function Forecaster (MQF^2), a global probabilistic forecasting method constructed using a multivariate quantile function and investigate its application to multi-horizon forecasting. Prior approaches are either autoregressive, implicitly capturing the dependency structure across time but exhibiting error accumulation with increasing forecast horizons, or multi-horizon sequence-to-sequence models, which do not exhibit error accumulation, but also do typically not model the dependency structure across time steps. MQF^2 combines the benefits of both approaches, by directly making predictions in the form of a multivariate quantile function, defined as the gradient of a convex function which we parametrize using input-convex neural networks. By design, the quantile function is monotone with respect to the input quantile levels and hence avoids quantile crossing. We provide two options to train MQF^2: with energy score or with maximum likelihood. Experimental results on real-world and synthetic datasets show that our model has comparable performance with state-of-the-art methods in terms of single time step metrics while capturing the time dependency structure.

1 INTRODUCTION

Among the many applications of time series forecasting (see e.g., Petropoulos et al. (2021) for an overview), inventory management in supply chain contexts has a prominent place. For this use-case in particular, *probabilistic* forecasts provide the input to downstream decision making problems such as replenishment decisions. For example, variations of the classic newsvendor problem show a direct correspondence between different quantile levels of a probabilistic forecast distribution with safety stocks in inventory management. It is therefore no surprise that recently proposed probabilistic forecasting methods have considered quantiles or the quantile function to represent probabilistic forecasts in the univariate case (Gasthaus et al., 2019; Wen et al., 2017; Park et al., 2021; Gouttes et al., 2021).

In this present work, we extend and generalize existing work by considering multivariate quantile functions. Existing quantile-based methods make predictions in the form of univariate quantiles (or quantile functions), i.e., independently for multiple time points in a multi-horizon setting, or independently across items in a multivariate forecasting setting (or both), and thereby ignore existing dependency structures in their forecasts. The extension to multivariate quantile functions allows us to capture these dependencies, which can have a significant impact on the accuracy of downstream systems (e.g., automatic inventory management) by capturing effects such as cannibalization, cross-selling or substitutability of products (Zhang et al., 2014; Rajaram and Tang, 2001; Hanasusanto et al., 2015). Despite their benefits, we note that multivariate quantile functions have not been studied or applied extensively in the literature on forecasting or machine learning (although there is a growing body of work from statistics and econometrics). We speculate that this may be due to the fact that the generalization of quantile functions from univariate to multivariate is not unique and exploring the properties of different notions of multivariate quantiles is still an active area of research. In contrast, multivariate probabilistic forecasts represented as multivariate probability densities have recently received more attention (Rasul et al., 2020; Salinas et al., 2019; de Bézenac et al., 2020; Rasul et al., 2021).

Proceedings of the 25th International Conference on Artificial Intelligence and Statistics (AISTATS) 2022, Valencia, Spain. PMLR: Volume 151.

[†]Work done while at Amazon Research.

In the univariate case, the quantile function of a random variable is (loosely speaking) the inverse of the cumulative distribution function (CDF). In the multivariate case, however, the corresponding notion of a *multivariate quantile function* is not uniquely defined, and in fact several proposals have been presented (see Carlier et al. (2016, Sec 2.4)), emphasizing different attributes of univariate quantile functions. Here we adopt the definition of Carlier et al. (2016) (studied further in Chernozhukov et al. (2017); Hallin et al. (2021)) as maps that (i) map a reference distribution (e.g., uniform on the unit cube) to the target distribution, and (ii) are monotonic, where the particular notion of multivariate monotonicity used is that of being the gradient of a convex function. Through Brenier's theorem (Brenier, 1991) and Knott-Smith optimality criterion (Knott and Smith, 1984), this definition characterizes multivariate quantile functions as the unique solutions to optimal transport problems with quadratic costs. Further, property (i) immediately connects this notion to normalizing flows (where typically the inverse direction is parametrized), and indeed a flow obeying this particular form of monotonicity has recently been proposed (Huang et al., 2021).

In more details, our contributions are as follows:

- Building on the notion of multivariate quantile functions as gradients of convex function put forth in Carlier et al. (2016), we propose to parametrize multivariate quantile functions via the gradients of *input convex neural networks* (Amos et al., 2017).
- We propose a training procedure based on the *energy score* (Gneiting and Raftery, 2007), a generalization of the continuous ranked probability score (Matheson and Winkler, 1976) to the multivariate case, and empirically demonstrate that this is effective and robust. Our model can alternatively be trained using a more standard maximum likelihood estimation approach, by relating it to normalizing flows (in particular the convex potential flows proposed in Huang et al. (2021)).
- We combine the multivariate quantile function model with an RNN-based feature extractor, resulting in a forecasting method that yields accurate joint multi-step forecasts. To the best of our knowledge, we are the first to represent multivariate forecasts using multivariate quantile functions.

In our empirical evaluations we show the practical viability of our approach in a series of experiments on both real-world and synthetic data where we employ the multivariate quantile function to model the multi-step forecast distribution. Our approach avoids pitfalls like error accumulation (Salinas et al., 2020) and quantile crossing (Wen et al., 2017) while allowing for realistic samples from the probabilistic forecast. The latter is particularly important in applications that require human interaction, e.g., in a supply chain context, where business analysts want to consider extreme scenarios to sharpen their intuition about the future.

The rest of the paper is organized as follows. In Section 3 we review the building blocks of our methodology: multivariate quantile functions and various training objectives. In Section 4 we present our model and describe the training and inference procedures. In Section 5 we provide an empirical evaluation on several real-world datasets and conclude the paper in Section 6. We start by reviewing the state of the art.

2 RELATED WORK

Deep learning-based approaches to probabilistic time series forecasting have been widely studied (see Benidis et al. (2020) and references therein). In addition to models utilizing parametric distributions (e.g., DeepAR (Salinas et al., 2019)), approaches based on quantile regression (Koenker and Bassett, 1978; Koenker, 2005) combined with RNN/CNN (Wen et al., 2017) or Transformer-based (Li et al., 2019; Lim et al., 2021) feature extractors have been shown to be flexible and effective. However, these approaches are limited to univariate predictions at pre-specified quantile levels and suffer from the quantile crossing problem. Recent work on modeling univariate quantile functions (Gasthaus et al., 2019; Park et al., 2021) has addressed these limitations while still focusing on the univariate case. Our approach extends this work to multivariate quantile functions, and similarly does not suffer from quantile crossing or require quantile level pre-specification.

The idea of using (univariate) quantile levels as input to a neural network in order to define a flexible quantile function model has previously been explored. Dabney et al. (2018) proposed implicit quantile networks which are trained by minimizing quantile loss using random uniform samples as input in the context of distributional reinforcement learning to model the state-action return distribution, and Gouttes et al. (2021) employed the same approach in the context of time series forecasting. A similar approach using uniform samples as input and minimizing the corresponding quantile loss has been proposed in Tagasovska and Lopez-Paz (2019) as a generic mechanism for modeling aleatoric uncertainty. However, none of these approaches explicitly enforce the monotonicity constraint on the quantile function, nor consider multivariate quantiles.

Other multivariate notions of quantile functions than the one we use here have been proposed, e.g., through univariate conditional quantile functions (requiring the

choice of an ordering) (Wei, 2008), or as gradients of convex potentials but without requiring them to transport from a reference distribution to the target distribution (Koltchinskii, 1997). The notion we make use of here allows us to easily obtain samples (by being an optimal transport) while not requiring the choice of a particular ordering of the dimensions.

Conceptually closest to our approach, although not proposed in the setting of time series forecasting, is the work on convex potential flows (Huang et al., 2021). As in our work, and also inspired by the connection to optimal transport through Brenier's theorem, the authors propose to define an invertible model as the gradient of a convex function and demonstrate how the inverse as well as the Jacobian determinant required for likelihood-based learning can be computed efficiently. They also propose to use input-convex neural networks to model the underlying convex function. Their work does not, however, consider the invertible mapping as a multivariate quantile function and—like other work on normalizing flows trained using maximum likelihood—uses a parametrization in the "reverse" direction (from the target to the Gaussian reference distribution). The same idea of using the gradient of an input-convex neural network model to define functions that are solutions to optimal transport problems under quadratic cost has also been proposed in Bunne et al. (2021), albeit embedded in a larger architecture for modeling population dynamics.

More broadly, density models defined through invertible maps ("normalizing flows") (Kobyzev et al., 2021) have been used in the context of time series forecasting as flexible uni- and multivariate density models: Rasul et al. (2020) proposed to directly parametrize a multivariate forecast distribution using a normalizing flow, while de Bézenac et al. (2020) combined a linear-Gaussian dynamical system with an invertible output model. These approaches, however, treat the flows as generic density estimators and do not draw the connection to multivariate quantile functions.

3 BACKGROUND

In this section we introduce the necessary background and building blocks of our approach: multivariate quantile functions, the energy score (which forms the basis of our training procedure), normalizing flows (which underlie the alternative maximum likelihood training procedure), and (partially) input convex neural networks (which we use to parametrize the convex function).

3.1 Multivariate Quantile Functions

In the univariate case, for a real-valued random variable Z, denote by $F_Z(z)$ its CDF. The corresponding quantile function is defined as

$$q_Z(\alpha) := F_Z^{-1}(\alpha) = \inf\{z \in \mathbb{R} : \alpha \leq F_Z(z)\}. \quad (1)$$

Here $\alpha \in (0, 1)$ is the quantile level, which is the probability that Z is less than $q_Z(\alpha)$.

While the CDF naturally extends to the multivariate case ($F_{\mathbf{Z}} : \mathbb{R}^d \to (0, 1)$), it is not invertible in general, precluding us from defining the corresponding quantile functions as its inverse. One natural way to define a quantile function of an n-variate random variable $\mathbf{Z}$ is as a mapping from $(0, 1)^n$ to $\mathbb{R}^n$. The input to this mapping is a quantile vector $\boldsymbol{\alpha} \in (0, 1)^n$ (instead of a single quantile level α), where the i-th entry α_i represents the quantile level of $[q_Z(\boldsymbol{\alpha})]_i$. However, such a mapping is not uniquely defined, as the entries of the quantile vector can interact with each other, so that the quantile levels do not have the same probabilistic meaning (Carlier et al., 2016). The definition proposed in Carlier et al. (2016) that we adopt here resolves this ambiguity by enforcing a particular notion of monotonicity.

In the univariate case, by construction, the quantile function has two essential properties. The first property is satisfying the representation property[1]

$$\mathbf{Z} = q_{\mathbf{Z}}(\boldsymbol{\alpha}), \quad \boldsymbol{\alpha} \sim U(0, 1)^n, \quad (2)$$

for $n = 1$. Here, we slightly abused notation by denoting $\boldsymbol{\alpha}$ as a random variable. The second property is monotonicity, i.e.,

$$\text{if } \alpha_1 < \alpha_2, \text{ then } q_Z(\alpha_1) \leq q_Z(\alpha_2). \quad (3)$$

The multivariate (vector) quantile function proposed in Carlier et al. (2016) is defined as a gradient of a convex function (a multivariate notion of monotonicity, thus extending (3)) that satisfies the representation property (2). Moreover, the convexity implies

$$(q_{\mathbf{Z}}(\boldsymbol{\alpha}_1) - q_{\mathbf{Z}}(\boldsymbol{\alpha}_2))^\top (\boldsymbol{\alpha}_1 - \boldsymbol{\alpha}_2) \geq 0, \quad (4)$$

which reduces to the monotonicity in the univariate case. Thus, this definition reduces to the classical quantile function in the univariate case. By defining the quantile function to be monotonic in the sense of being the gradient of a convex function, a connection to work on optimal transport (Villani, 2009; Peyré et al., 2019) is drawn, where Brenier's polar factorization theorem (Brenier, 1991) and Knott-Smith optimality criterion (Knott and Smith, 1984) establish that

[1] In general, quantile vectors can follow distributions other than $U(0, 1)^n$ (Carlier et al., 2016). For instance, one can use the isotropic Gaussian distribution.

such functions are the unique optimal transports with quadratic cost. In particular, the representation property and monotonicity are necessary and sufficient for multivariate quantile functions to be the unique optimal transports from a reference distribution (typically chosen to be uniform on the unit cube) to the distribution of interest under quadratic cost, i.e. they minimize the Wasserstein distance $\min_{q:q(\boldsymbol{\alpha})=\mathbf{Z}} \mathbb{E}_{\boldsymbol{\alpha}\sim U(0,1)^n} \|\boldsymbol{\alpha} - q(\boldsymbol{\alpha})\|_2^2$.

3.2 Energy Score

In the univariate case, given realizations z of the random variable Z, we seek to estimate the quantile function q_Z for all quantile levels $\alpha \in (0,1)$. We can achieve this by minimizing the continuous ranked probability score (CRPS) (Gneiting and Raftery, 2007), defined as

$$L_{\text{CRPS}}(q,z) = \mathbb{E}_{w,w'\sim q(U(0,1))}\left[-\frac{1}{2}|w-w'| + |w-z|\right], \tag{5}$$

where w and w' are independent. CRPS is strictly proper (Gneiting and Raftery, 2007), i.e.,

$$\mathbb{E}_{z\sim Z} L_{\text{CRPS}}(q_Z, z) < \mathbb{E}_{z\sim Z} L_{\text{CRPS}}(q, z), \tag{6}$$

for any Z and $q \neq q_Z$, both with finite first moment. In other words, for realizations z of the random variable Z, the unique minimizer of CRPS is the quantile function q_Z.

The energy score (Gneiting and Raftery, 2007) is an extension of the CRPS to the multivariate setting, which takes a statistical energy perspective from Székely (2003). It is defined as

$$L_{\text{ES}}(q,\mathbf{z}) = \mathbb{E}_{\mathbf{w},\mathbf{w}'\sim q(U(0,1)^n)}\Bigg[-\frac{1}{2}\|\mathbf{w}-\mathbf{w}'\|_2^\beta + \|\mathbf{w}-\mathbf{z}\|_2^\beta\Bigg], \tag{7}$$

where $\mathbf{w}$ and $\mathbf{w}'$ are independent. If $\beta \in (0,2)$, the energy score is strictly proper (Székely, 2003) for any $\mathbf{Z}$ and $q \neq q_{\mathbf{Z}}$ satisfying $\mathbb{E}_{\mathbf{z}\sim\mathbf{Z}}\|\mathbf{z}\|_2^\beta < \infty$ and $\mathbb{E}_{\mathbf{w}\sim q(U(0,1)^n)}\|\mathbf{w}\|_2^\beta < \infty$, respectively.

In practice, we approximate the energy score by

$$\tilde{L}_{\text{ES}}(q,\mathbf{z}) = -\frac{1}{2|\mathcal{C}||\mathcal{C}'|}\sum_{\mathbf{w}\in\mathcal{C},\mathbf{w}'\in\mathcal{C}'}\|\mathbf{w}-\mathbf{w}'\|_2^\beta + \frac{1}{|\mathcal{C}''|}\sum_{\mathbf{w}''\in\mathcal{C}''}\|\mathbf{w}''-\mathbf{z}\|_2^\beta \tag{8}$$

where $\mathcal{C}$s are sets of finite samples drawn from $q(U(0,1)^n)$.

3.3 Normalizing Flows

Normalizing flows (Tabak and Turner, 2013; Ruthotto and Haber, 2021) are C^1-diffeomorphic and orientation-preserving functions which map from $\mathbb{R}^n$ to $\mathbb{R}^n$. In particular, they transform a random variable of interest $\mathbf{Z}$ with density $p_{\mathbf{Z}}$ into $\mathbf{Y}$ with a simple density $p_{\mathbf{Y}}$ which can be easily evaluated, typically an isotropic Gaussian. Note that the mappings of normalizing flows go in the opposite direction to quantile functions, which map samples of a uniform (simple) distribution to $\mathbf{Z}$. To distinguish between the two opposite directions, in this paper we denote normalizing flows as $\bar{g}$.

Maximum Likelihood Estimation Using the change of variables formula, we estimate $p_{\mathbf{Z}}(\mathbf{z})$ as

$$p_{\mathbf{Z}}(\mathbf{z}) \approx p_{\bar{g}}(\mathbf{z}) = p_{\mathbf{Y}}(\bar{g}(\mathbf{z}))\det\left(\frac{\partial\bar{g}(\mathbf{z})}{\partial\mathbf{z}}\right), \tag{9}$$

where $p_{\bar{g}}$ is the density induced by $\bar{g}$. We target to find a normalizing flow that approximates the true density $p_{\mathbf{Z}}$ well. One method to evaluate the discrepancy between the two densities in (9) is the Kullback-Leibler (KL) divergence defined by

$$\begin{aligned}\text{KL}(p_{\mathbf{Z}}\|p_{\bar{g}}) &= \mathbb{E}_{\mathbf{z}\sim\mathbf{Z}}\left[\log\left(\frac{p_{\mathbf{Z}}(\mathbf{z})}{p_{\bar{g}}(\mathbf{z})}\right)\right] \\ &= \mathbb{E}_{\mathbf{z}\sim\mathbf{Z}}\log(p_{\mathbf{Z}}(\mathbf{z})) - \mathbb{E}_{\mathbf{z}\sim\mathbf{Z}}\log(p_{\bar{g}}(\mathbf{z})).\end{aligned}$$

The first term is a constant and can be dropped in minimization. Replacing the expectation of the second term by samples $\mathbf{z}_i \sim \mathbf{Z}$, for $i = 1, ..., m$, we obtain the negative log-likelihood given by

$$\begin{aligned}&\frac{1}{m}\sum_{i=1}^m \tilde{L}_{\text{ML}}(\bar{g}(\mathbf{z}_i)) := \frac{1}{m}\sum_{i=1}^m[-\log(p_{\bar{g}}(\mathbf{z}_i))] \\ &= \frac{1}{m}\sum_{i=1}^m\left[-p_{\mathbf{Y}}(\bar{g}(\mathbf{z}_i)) - \log\left[\det\left(\frac{\partial\bar{g}(\mathbf{z}_i)}{\partial\mathbf{z}}\right)\right]\right].\end{aligned} \tag{10}$$

Sample Generation After training the normalizing flow, we can generate predicted samples of $\mathbf{Z}$ by going backward through the flow. That is computing the inverse $\bar{g}^{-1}(\mathbf{y})$, where $\mathbf{y}$ is a sample drawn from the reference distribution defined by $p_{\mathbf{Y}}$.

3.4 Partially Input Convex Neural Network

Input convex neural network (ICNN) (Amos et al., 2017) is a neural network with special constraints on its architecture such that it is convex with respect to (a part of) its input. ICNN has demonstrated successful applications in various optimal transport and optimal control problems (Bunne et al., 2021; Chen et al., 2019; Huang et al., 2021; Makkuva et al., 2020). Moreover, it has been proved that, under mild assumptions, ICNN

and its gradient can universally approximate convex functions (Chen et al., 2019) and their gradients (Huang et al., 2021), respectively. This means that the gradient of ICNN can universally approximate multivariate quantile functions in the marginal case. Moreover, we will demonstrate in our experiments that it can also effectively approximate conditional quantile functions.

We consider in this work a type of the ICNN called the partially input convex neural network (PICNN) (Amos et al., 2017), we follow the PICNN architecture used in (Huang et al., 2021). For an input pair $(\boldsymbol{\alpha}, \mathbf{h}) \in (\mathbb{R}^n \times \mathbb{R}^d)$, the key feature of PICNN is that it is only convex with respect to $\boldsymbol{\alpha}$. The i-th layer of a k-layer PICNN, with $i = 1, \dots, k$, is represented as:

$$\mathbf{v}_{i+1} = a_i\Big(\mathbf{W}_i^{(\mathbf{v})}\Big(\mathbf{v}_i \circ [\mathbf{W}_i^{(\mathbf{vu})}\mathbf{u} + \mathbf{b}_i^{(\mathbf{v})}]_+\Big) + \mathbf{W}_i^{(\boldsymbol{\alpha})}\Big(\boldsymbol{\alpha} \circ (\mathbf{W}_i^{(\boldsymbol{\alpha}\mathbf{u})}\mathbf{u} + \mathbf{b}_i^{(\boldsymbol{\alpha})})\Big) + \mathbf{W}_i^{(\mathbf{u})}\mathbf{u} + \mathbf{b}_i\Big), \tag{11}$$

$$\text{with: } G_{\boldsymbol{\theta}}(\boldsymbol{\alpha}, \mathbf{h}) = \mathbf{v}_k, \mathbf{u} = a(\tilde{\mathbf{W}}\mathbf{h} + \tilde{\mathbf{b}}). \tag{12}$$

Here, $\mathbf{W}$'s and $\mathbf{b}$'s are weights and bias of the network, respectively, they are collectively denoted as $\boldsymbol{\theta}$, $\circ$ denotes the Hadamard product, and $[\cdot]_+$ denotes the ReLU activation function. Moreover, to render the convexity of the network, a_i's are convex and non-decreasing activation functions, and $\mathbf{W}_i^{(\mathbf{v})}$'s have non-negative entries.

4 MULTIVARIATE QUANTILE FUNCTION FORECASTER

In this section, we introduce the Multivariate Quantile Function Forecaster (MQF^2) which uses a multivariate quantile function conditioned on the past time points to make probabilistic forecasts. Contemporary deep learning based probabilistic forecasting methods like DeepAR (Salinas et al., 2020), MQRNN (Wen et al., 2017), or TFT (Lim et al., 2021) consist of two components: an encoder that extracts features from past observations and compresses them into a finite-dimensional hidden state (Salinas et al. (2019) used an RNN-based encoder, Wen et al. (2017) used RNNs and CNNs, and Lim et al. (2021); Eisenach et al. (2020) used a Transformer-based architecture), and probabilistic output model which transforms the hidden state into a representation of the probability distribution over future observations (e.g. parametric density-based for Salinas et al. (2019), univariate quantile predictions for Wen et al. (2017); Eisenach et al. (2020); Lim et al. (2021), normalizing flow-based for Rasul et al. (2020)). We follow the same paradigm here and condition the multivariate quantile function (which constitutes the output model) on the hidden state produced by an encoder network. As our focus lies on the output model, we restrict our attention to that component (see e.g. Salinas et al. (2019); Wen et al. (2017); Benidis et al. (2020) for details on the general setup) and only consider the combination with a DeepAR-based encoder (and follow the same window-based training procedure detailed in Salinas et al. (2020)), while in principle our approach can be combined with any encoder architecture. In the remainder of this section, we first present how the PICNN can be used to model a multivariate quantile function. Then, we propose two alternatives to train the model, using the energy score and maximum likelihood, respectively.

4.1 PICNN Quantile Function

We propose to use the gradient of a PICNN $g_{\boldsymbol{\theta}}(\boldsymbol{\alpha}, \mathbf{h}) := \nabla_{\boldsymbol{\alpha}} G_{\boldsymbol{\theta}}(\boldsymbol{\alpha}, \mathbf{h})$ to model a conditional multivariate quantile function $q_{\boldsymbol{\theta}}(\boldsymbol{\alpha}|\mathbf{h})$ with a quantile vector $\boldsymbol{\alpha} \in (0,1)^n$. We illustrate this in the right half of Figure 1. The PICNN $G_{\boldsymbol{\theta}}(\boldsymbol{\alpha}, \mathbf{h})$ is convex with respect to only $\boldsymbol{\alpha}$. Through this setup our multivariate quantile function satisfies the monotonicity property and hence (3) by design. In addition, the fact that the network is not convex with respect to the second input vector $\mathbf{h}$ allows us to flexibly condition the multivariate quantile function on input features or a representation of them produced by a time series encoder model.

As presented in Section 3.1, there are two essential properties for a quantile function, the representation property (2) and the monotonicity property. Our parametrization through the gradient of the PICNN constrains the multivariate quantile function to fulfill the monotonicity property, which means that we can train our model with standard gradient descent optimizers so as to come as close as possible to the representation property. To do so, we propose two alternatives: training with the energy score or with maximum likelihood.

While this representation of a multivariate quantile function is general and can be used in any regression context, we propose to use it in the probabilistic forecasting context. We use a forecasting encoder network $H_{\boldsymbol{\Phi}}(\mathbf{x}) = \mathbf{h}$ to obtain a representation of the past time series on which we condition the quantile function. To the best of our knowledge, this is the first application of the definition of monotonicity to construct a multivariate quantile function for all quantile levels $\boldsymbol{\alpha} \in (0,1)^n$ and the use of ICNN for this application.

4.2 Training Procedure

We propose two alternative procedures to train the multivariate quantile forecasting functions described above. First using the energy score to bring the distribution of samples from the model to as close to the true

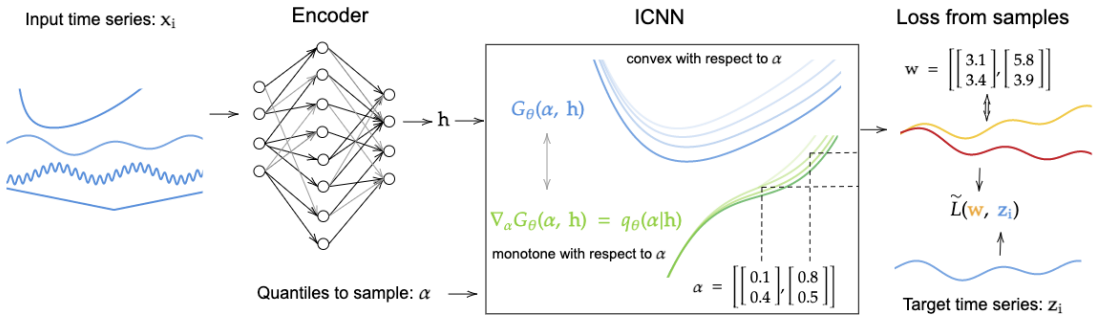


Figure 1: Schematic representation of MQF^2. The crux of the method is the full multivariate quantile function which is monotone with respect to the multivariate quantile vector $\boldsymbol{\alpha}$. It is achieved by modeling the quantile function with the gradient of a function $G_{\boldsymbol{\theta}}$ which is convex with respect to $\boldsymbol{\alpha}$. The quantile function is conditioned on the past of the time series through a representation obtained from an encoder network. The network is trained by minimizing the loss between forecast samples and the true target.

distribution as possible and so fulfill the representation property. The second option is to use normalizing flows as the inverse of the quantile function to map the observed samples to a Gaussian distribution. The network is then trained to maximize the likelihood of the mapped samples under the Gaussian distribution.

4.2.1 Training via Energy Score

We propose to train MQF^2 using the energy score (Gneiting and Raftery, 2007), the generalization of CRPS to multivariate distributions.

Training Consider m training example pairs $\{(\mathbf{x}_i, \mathbf{z}_i)\}_{i=1}^m$, where $\mathbf{x}_i$ denotes the input features and $\mathbf{z}_i$ the target output. Each $\mathbf{z}_i$ can span multiple time steps and/or across multiple time series. We minimize the approximated energy score $\tilde{L}_{\text{ES}}$ in (8) as

$$\min_{\boldsymbol{\theta},\boldsymbol{\Phi}} \frac{1}{m} \sum_{i=1}^{m} \tilde{L}_{\text{ES}}(q_{\boldsymbol{\theta}}(\cdot|H_{\boldsymbol{\Phi}}(\mathbf{x}_i)), \mathbf{z}_i),$$

where $q_{\boldsymbol{\theta}}(\cdot|H_{\boldsymbol{\Phi}}(\mathbf{x}_i)) = g_{\boldsymbol{\theta}}(\cdot, H_{\boldsymbol{\Phi}}(\mathbf{x}_i))$ is the multivariate quantile function.

Inference Our multivariate quantile function is trained to provide estimate on all quantile levels, for inference we can compute $\tilde{\mathbf{z}} = q_{\boldsymbol{\theta}}(\boldsymbol{\alpha}|\mathbf{h})$, where $\boldsymbol{\alpha} \in (0,1)^n$ is drawn from a uniform distribution.[2]

4.2.2 Training via Maximum Likelihood

The second option is to train the gradient of PICNN through (conditional) normalizing flows. This approach follows Huang et al. (2021), which proposed to use ICNN as normalizing flows.

We note that normalizing flows take target samples $\mathbf{z}$ as input and return the corresponding Gaussian samples, as opposed to quantile functions which take uniform samples and output $\mathbf{z}$. To distinguish this reversed direction of mapping, we denote the PICNN used for normalizing flows as $\breve{G}_{\boldsymbol{\theta}}$.

Invertible Gradient Since the gradient of PICNN is used as normalizing flows, it needs to be invertible. To this end, an l_2 term is added to $\mathbf{v}_k$, the final layer of the PICNN (12), i.e.,

$$\breve{G}_{\boldsymbol{\theta}}(\mathbf{z}, \mathbf{h}) = \mathbf{v}_k(\mathbf{z}, \mathbf{h}) + \frac{\gamma}{2}\|\mathbf{z}\|_2^2, \tag{13}$$

where $\gamma > 0$ is a trainable parameter. The additional term renders $\breve{G}_{\boldsymbol{\theta}}$ strongly convex, and hence its gradient is invertible. In Huang et al. (2021), they term the mapping of the gradient $\breve{g}_{\boldsymbol{\theta}} := \nabla_{\mathbf{z}} \breve{G}_{\boldsymbol{\theta}}$ as convex potential flows.

Training Given the training example pairs $\{(\mathbf{x}_i, \mathbf{z}_i)\}_{i=1}^m$, we minimize the negative log-likelihood, with $\tilde{L}_{\text{ML}}$ defined in (10), as

$$\min_{\boldsymbol{\theta},\boldsymbol{\Phi}} \frac{1}{m} \sum_{i=1}^{m} \tilde{L}_{\text{ML}}(\breve{g}_{\boldsymbol{\theta}}(\mathbf{z}_i, H_{\boldsymbol{\Phi}}(\mathbf{x}_i))).$$

Inference The normalizing flow can serve as a (generalized) quantile function (Carlier et al., 2016), which takes inputs drawn from an isotropic Gaussian distribution and output the predicted target samples.

In particular, we first sample $\mathbf{y} \in \mathbb{R}^n$ from the isotropic Gaussian distribution. Then we go backward through the flow to obtain the prediction $\tilde{\mathbf{z}}$. To this end, we

[2] In practice, we use a generalized quantile vector (Carlier et al., 2016), which follows the isotropic Gaussian distribution. Because this empirically allows for a better training.

solve the convex minimization problem

$$\min_{\mathbf{z}} \overleftarrow{G}_{\boldsymbol{\theta}}(\mathbf{z}, \mathbf{h}) - \mathbf{z}^\top \mathbf{y}, \tag{14}$$

whose minimum $\tilde{\mathbf{z}}$ satisfies $\overleftarrow{g}_{\boldsymbol{\theta}}(\tilde{\mathbf{z}}, \mathbf{h}) = \mathbf{y}$. That is $\tilde{\mathbf{z}} = \overleftarrow{g}_{\boldsymbol{\theta}}^{-1}(\mathbf{y}, \mathbf{h})$. The minimization problem (14) is solved using the L-BFGS algorithm (Liu and Nocedal, 1989).

Monotonicity Since we are using the inverse of the normalizing flow as the quantile function, it is important to show that the inverse $\overleftarrow{g}_{\boldsymbol{\theta}}^{-1}$ is also monotone, i.e., it is the gradient of a convex function. The following proposition guarantees the monotonicity of $\overleftarrow{g}_{\boldsymbol{\theta}}^{-1}(\cdot, \mathbf{h})$.

Proposition 1. *Let $\mathcal{D} \subseteq \mathbb{R}^n$ be open, $G : \mathcal{D} \to \mathbb{R}$ be a strongly convex and smooth function and g be its gradient. Then g^{-1} exists and is the gradient of a convex function.*

For the proof of Proposition 1, we refer the readers to the Appendix. Note that the assumption of smoothness is satisfied when the PICNN architecture uses smooth activation functions such as the softplus function to render the whole network smooth.

5 EXPERIMENTS

Our MQF^2 can be multivariate in prediction horizon (multi-horizon) and/or across multiple time series. In our experimental evaluations, we focus on the former case, where at a given time point the model outputs a distribution of multiple points into the future. This evaluation setup allows us to compare to standard univariate forecasting models, here MQCNN (Wen et al., 2017) and DeepAR (Salinas et al., 2020). These two models represent different approaches for multi-horizon predictions. On the one hand, MQCNN factorizes the multivariate distribution over the time steps, considering them independently of each other and therefore the time dependency structure among them is ignored. On the other hand, DeepAR only predicts a single time step at a time and the model is unrolled to predict the full forecast horizon. By doing so it implicitly models the forward dependency among time steps, but at the cost of error accumulating.

We recall that MQF^2 is generic because it can be used in many sequence-to-sequence architectures as an alternative to the decoder. For our experiments, we choose to implement MQF^2 on top of a DeepAR encoder. We use the default hyperparameters for the comparison methods as found in GluonTS (Alexandrov et al., 2020). For MQF^2 we use the default parameters for the DeepAR encoder and PICNN with 40 hidden units and 5 hidden layers for the real experiments, and with 10 hidden units and 2 hidden layers for the synthetic experiments. We train the model to convergence (For real data experiments, we use 100 epochs for MQCNN and DeepAR and 300 for MQF^2 as it is more complex. For synthetic experiments, we use 50 epochs for all models). Otherwise all the hyperparameters are kept constant across models. MQF^2 is implemented in PyTorch[3] (Paszke et al., 2019). We refer the readers to the Appendix for more details on the experimental details, model hyperparameters and their robustness.

We evaluate our model on both real and synthetic data. For the real experiments, we evaluate the methods on several real-world datasets and report the performance in terms of various univariate and multivariate metrics. For the synthetic experiment, we test the ability of different models to learn and predict artificial data which follow a Gaussian process.

5.1 Experiments on Real Data

We perform experiments on `Elec` and `Traf` from the UCI data repository (Dheeru and Karra Taniskidou, 2017), and different `M4` competition datasets (Makridakis et al., 2018). The results are shown in Table 1. Experimental results in terms of more metrics and hyperparameter robustness tests are available in the Appendix. In the following we analyze them along different angles.

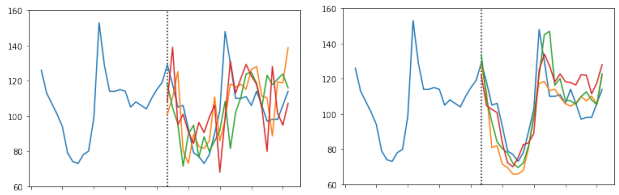
MQF^2 is competitive with the state of the art. Table 1 shows the mean scaled interval score (MSIS) (Gneiting and Raftery, 2007) and mean weighted quantile loss, averaged over the $\{0.1, 0.2, ..., 0.9\}$ quantiles and over the full forecast horizon. These are univariate probabilistic forecasting metrics which are computed at each point in the forecast horizon and are averaged over the points. We observe that MQF^2 is very competitive with the state of the art. In particular, under these two metrics, MQF^2 performs the best in all but 1 dataset (in which MQF^2's performance is close to the best one). While a main advantage of our method is to model the time dependencies across the time dimension of the forecast horizon, its performance on modeling the marginal distributions is comparable to that of MQCNN, which by design only learns such distributions.

MQF^2 captures the time dependency between outputs. We use two multivariate metrics to evaluate the multivariate distributions produced by different models. First, we measure the energy score between samples from the forecasting models and the observed target time series. In addition, we compute CRPS between the sum of these samples and the sum of the observed target time series. The distribution of a sum depends on the dependency among its elements. Hence,

[3] available at `https://github.com/awslabs/gluon-ts/tree/master/src/gluonts/torch/model/mqf2`.

Dataset	Model	Metrics over full horizon				Mean Quantile Loss over differing forecast horizon				
		sum CRPS	Energy score	MSIS	mean_wQL	1 step	5 steps	10 steps	15 steps	20 steps
Elec	MQCNN	2323.5 ± 54.2	1282.1 ± 1.0	11.7 ± 0.0	0.086 ± 0.0	0.042 ± 0.0	0.125 ± 0.01	0.117 ± 0.02	0.094 ± 0.01	**0.062 ± 0.0**
	DeepAR	3059.6 ± 180.8	971.7 ± 59.0	7.3 ± 0.1	0.07 ± 0.0	**0.027 ± 0.0**	**0.055 ± 0.0**	**0.059 ± 0.0**	0.074 ± 0.0	0.089 ± 0.01
	MQF^2+ ES	**1723.7 ± 122.8**	**891.1 ± 32.1**	**6.9 ± 0.1**	**0.066 ± 0.0**	0.031 ± 0.0	0.08 ± 0.01	0.102 ± 0.0	0.056 ± 0.0	0.068 ± 0.01
	MQF^2+ ML	2332.523 ± 146.88	893.6 ± 53.8	7.2 ± 0.6	**0.066 ± 0.0**	0.038 ± 0.01	0.088 ± 0.02	0.073 ± 0.01	**0.053 ± 0.0**	0.067 ± 0.01
Traf	MQCNN	0.419 ± 0.33	0.161 ± 0.06	46.1 ± 1.6	0.993 ± 0.29	0.905 ± 0.4	5.909 ± 0.37	0.878 ± 0.42	0.871 ± 0.23	0.644 ± 0.18
	DeepAR	0.108 ± 0.01	0.061 ± 0.0	7.2 ± 0.1	0.131 ± 0.0	**0.074 ± 0.0**	**0.163 ± 0.0**	**0.117 ± 0.0**	0.123 ± 0.0	0.144 ± 0.0
	MQF^2+ ES	**0.095 ± 0.0**	**0.06 ± 0.0**	7.2 ± 0.0	0.142 ± 0.0	0.104 ± 0.0	0.298 ± 0.01	0.139 ± 0.01	0.127 ± 0.0	**0.139 ± 0.01**
	MQF^2+ ML	0.097 ± 0.0	0.062 ± 0.0	**6.6 ± 0.1**	**0.13 ± 0.0**	0.078 ± 0.0	0.165 ± 0.01	0.13 ± 0.0	**0.12 ± 0.0**	0.14 ± 0.0
M4-daily	MQCNN	3089.8 ± 10.4	923.1 ± 3.6	41.9 ± 0.9	0.027 ± 0.0	**0.009 ± 0.0**	0.019 ± 0.0	**0.024 ± 0.0**	-	-
	DeepAR	3186.2 ± 966.5	989.3 ± 244.3	50.5 ± 8.0	0.039 ± 0.01	0.015 ± 0.0	0.028 ± 0.01	0.049 ± 0.01	-	-
	MQF^2+ ES	**1752.0 ± 47.3**	**619.2 ± 8.7**	31.1 ± 0.3	**0.024 ± 0.0**	0.013 ± 0.0	**0.019 ± 0.0**	0.027 ± 0.0	-	-
	MQF^2+ ML	1786.028 ± 60.94	622.0 ± 14.7	**30.5 ± 0.3**	**0.024 ± 0.0**	0.01 ± 0.0	0.019 ± 0.0	0.029 ± 0.0	-	-
M4-monthly	MQCNN	9196.6 ± 175.4	3269.9 ± 34.6	18.7 ± 0.4	0.12 ± 0.0	0.072 ± 0.0	0.096 ± 0.0	0.115 ± 0.0	**0.134 ± 0.0**	-
	DeepAR	**7337.0 ± 345.5**	2572.1 ± 95.0	14.0 ± 1.5	0.113 ± 0.0	0.063 ± 0.0	0.092 ± 0.0	0.115 ± 0.0	0.143 ± 0.01	-
	MQF^2+ ES	7365.7 ± 218.1	**2554.6 ± 79.3**	**12.8 ± 1.4**	**0.112 ± 0.0**	**0.059 ± 0.0**	**0.087 ± 0.0**	**0.113 ± 0.0**	0.145 ± 0.0	-
	MQF^2+ ML	8235.445 ± 0.0	2839.7 ± 0.0	14.4 ± 0.0	0.124 ± 0.0	0.066 ± 0.0	0.1 ± 0.0	0.124 ± 0.0	0.159 ± 0.0	-
M4-yearly	MQCNN	3753.7 ± 28.1	1976.2 ± 12.8	**34.2 ± 0.3**	**0.115 ± 0.0**	**0.064 ± 0.0**	0.141 ± 0.0	-	-	-
	DeepAR	3749.1 ± 42.6	1917.1 ± 7.9	34.9 ± 0.6	0.118 ± 0.0	0.065 ± 0.0	0.145 ± 0.0	-	-	-
	MQF^2+ ES	**3649.3 ± 60.9**	**1859.4 ± 28.1**	36.7 ± 1.6	0.116 ± 0.0	0.075 ± 0.0	**0.135 ± 0.0**	-	-	-
	MQF^2+ ML	3784.486 ± 105.76	1913.2 ± 35.2	38.8 ± 2.1	0.119 ± 0.0	0.07 ± 0.0	0.143 ± 0.0	-	-	-

Table 1: Results of MQF^2 compared with other state of the art methods (for all columns lower is better.) We show the mean and standard deviation over 3 training runs. A "-" indicates that the corresponding time step is beyond the prediction length of the dataset.



(a) MQCNN (b) MQF^2

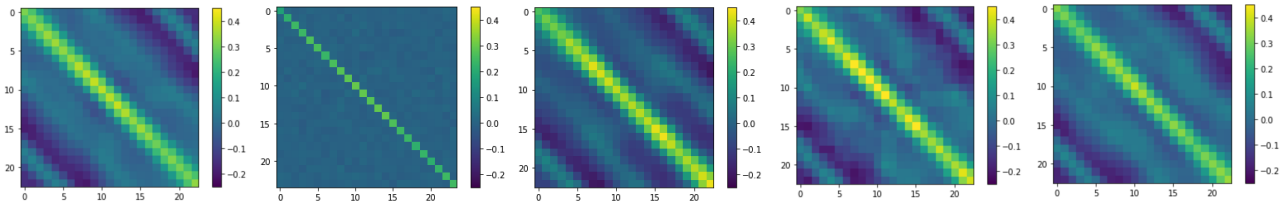
Figure 2: Three sample paths generated by MQCNN and MQF^2. The dotted vertical lines represent the start of the prediction horizon.

accurately measuring the dependency between the time points will result in a better estimate of the distribution of their sum. These two metrics are shown in Table 1 and are computed over the full forecast horizon of each dataset. We see that MQF^2 outperforms the comparing methods by some margin, especially when it is trained with energy score. In particular, MQF^2 performs the best in all but 1 result, in which it is very close to the best method and reports a much lower standard deviation over training runs.

We observe that MQCNN is underperforming because it assumes that the time points over the prediction horizon are independent and hence cannot capture time dependency. On the other hand, on some datasets like `Traf`, `M4-monthly`, and `M4-yearly`, DeepAR's implicit modeling of the forward time dependencies allows it to obtain results very close to MQF^2.

MQF^2 avoids error accumulation. DeepAR is able to model the forward dependency across time points implicitly through the unrolling on samples, however this can result in error accumulation through the unrolling (Rangapuram et al., 2018). To compare DeepAR with our model in this respect, we compute the mean weighted quantile losses on different forecast horizons. Table 1 shows the loss for 1, 5, 10, 15, and 20 steps ahead. Note that on some datasets the selected steps are longer than the prediction length, and the loss cannot be computed beyond the prediction length. We see that MQF^2 has competitive performance across all time steps and datasets. For all the datasets either MQCNN or MQF^2 perform the best on the furthest quantile horizon, even on datasets where DeepAR performs the best at shorter horizons. However, in the results of `Traf` dataset, we observe that MQF^2 has a more stable performance than MQCNN, which reports very high losses at all the time steps.

MQF^2 produces consistent sample paths. Beyond the quantitative evaluation of the multivariate distribution, we evaluate it qualitatively by visually inspecting predicted sample paths. In a model where the distribution over each of the time steps is modeled independently, sample paths would fail to represent the dependency between time points which can lead to unrealistic sampled forecasts. Figure 2 shows sample paths from MQCNN and MQF^2 on the same time series. We observe that the distributions of the samples at each time step are similar for both models. However, the sample paths from MQCNN fail to mimic the smoothness of the real time series, as each time point is modeled and sampled independently. On the contrary, note that the samples from MQF^2 indeed display realistic behavior because of its modeling of the time dependencies. We provide additional visualizations in the Appendix.



(a) Correlation matrix of the training samples. (b) MQCNN MAE: 0.075 (c) DeepAR MAE: 0.033 (d) MQF^2 + ES MAE: 0.023 (e) MQF^2 + ML MAE: 0.019

Figure 3: Experiments with generated samples from a Gaussian process over 24 time steps which have the correlation matrix visualized in (a). Figures (b)-(e) show the correlation matrices obtained from 200 samples of different models and the mean absolute error (MAE) between the model correlation matrix and the ground truth.

5.2 Experiments on Synthetic Data

In the real experiments, we observe that our MQF^2 best captures the time dependency structure. Here we further illustrate this advantage using a synthetic dataset of 500 time series of 24 points drawn from a Gaussian process (GP) with a correlation matrix shown in Figure 3(a). The kernel of the GP governing the covariance between time points is composed of a radial basis function kernel and a periodic kernel, resulting in a complex correlation structure.

We evaluate how well the different methods can model the marginal GP distribution. We train each of the methods on the GP samples and then generate 200 sample paths for each of them and compute the correlation matrix of the generated sample paths. The correlation matrices are shown in Figure 3(b)-(e). In addition to the visualization, we compute the mean absolute error (MAE) between the correlation matrix from model samples and the true correlation matrix. If a method captures the GP well, it will generate sample paths which closely follow the distribution and hence report a correlation matrix similar to the true one.

We see that MQCNN generates a correlation matrix which is essentially diagonal and reports the highest error. This shows that it fails to capture the correlation, as it assumes each time point to be independent and therefore ignores the time dependency structure. For the DeepAR method, its unrolling mechanism allows it to capture the correlation matrix reasonably well and report a much lower error than MQCNN. Finally, as our MQF^2 explicitly considers the whole sample path at once, it best approximates the true correlation matrix and has the lowest errors.

6 DISCUSSION

In this paper, we presented MQF^2, a novel method for probabilistic forecasting via a multivariate quantile function that we model as the gradient of an input convex neural network. Our experiments show that we maintain favorable properties of prior work on (univariate) quantile functions for probabilistic forecasts while addressing some of their shortcomings. In particular, sample paths (which are a commonly-used way of passing probabilistic forecasts to downstream components) can easily be generated from our model and correctly reflect the dependency structure across time (which also makes them visually coherent). Further, there is no accumulation of forecast error over the length of the forecast horizon and our method is overall very competitive with the state of the art.

Despite these benefits, there are situations and applications where alternative approaches might be better suited. In particular, autoregressive constructions that decompose the joint distribution into its telescoping univariate marginals (Wei, 2008; Uria et al., 2013; Papamakarios et al., 2017; Wang et al., 2019; Jaini et al., 2019), allow the quantile levels to retain their classical probabilistic interpretation (e.g. for the construction of univariate prediction intervals) and provide direct access to certain conditional distributions of interest (future conditioned on past). Similarly, multi-horizon approaches provide direct access to the univariate marginal distributions, which in our approach can only be obtained through sampling. In fact, an interesting avenue for future work is to explore whether a multivariate quantile function model can be constructed that retains the ability to access marginal and conditional distributions without resorting to sampling. Future work could further extend our approach to the practically important case of count distributions and assess the quality of our approach for quantile functions jointly over the time and item dimensions. Finally, more suitable forecasters in domain adaptation (Jin et al., 2022) with faster training schemes (Lu et al., 2021) can be developed, ultimately being able to be incorporated for downstream decision makings, e.g., planning cloud computing and vehicle controllers (Park et al., 2019, 2020; Kim et al., 2020).

Acknowledgements

The authors would like to thank the five anonymous referees for their thorough review and constructive suggestions. They would also like to thank Michael Bohlke-Schneider, Syama Sundar Rangapuram, Lorenzo Stella, and Jasper Zschiegner for reviewing the code and giving helpful advice.

References

Alexandrov, A., Benidis, K., Bohlke-Schneider, M., Flunkert, V., Gasthaus, J., Januschowski, T., Maddix, D. C., Rangapuram, S., Salinas, D., Schulz, J., Stella, L., Türkmen, A. C., and Wang, Y. (2020). GluonTS: Probabilistic and neural time series modeling in python. *Journal of Machine Learning Research*, 21(116):1–6.

Amos, B., Xu, L., and Kolter, J. Z. (2017). Input convex neural networks. In *International Conference on Machine Learning*, pages 146–155. PMLR.

Benidis, K., Rangapuram, S. S., Flunkert, V., Wang, B., Maddix, D. C., Türkmen, A. C., Gasthaus, J., Bohlke-Schneider, M., Salinas, D., Stella, L., Callot, L., and Januschowski, T. (2020). Neural forecasting: Introduction and literature overview. *CoRR*, abs/2004.10240.

Brenier, Y. (1991). Polar factorization and monotone rearrangement of vector-valued functions. *Communications on pure and applied mathematics*, 44(4):375–417.

Bunne, C., Meng-Papaxanthos, L., Krause, A., and Cuturi, M. (2021). Jkonet: Proximal optimal transport modeling of population dynamics. *arXiv preprint arXiv:2106.06345*.

Carlier, G., Chernozhukov, V., Galichon, A., et al. (2016). Vector quantile regression: an optimal transport approach. *Annals of Statistics*, 44(3):1165–1192.

Chen, Y., Shi, Y., and Zhang, B. (2019). Optimal control via neural networks: A convex approach. In *International Conference on Learning Representations*.

Chernozhukov, V., Galichon, A., Hallin, M., and Henry, M. (2017). Monge–Kantorovich depth, quantiles, ranks and signs. *The Annals of Statistics*, 45(1):223 – 256.

Dabney, W., Ostrovski, G., Silver, D., and Munos, R. (2018). Implicit quantile networks for distributional reinforcement learning. In Dy, J. and Krause, A., editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1096–1105. PMLR.

de Bézenac, E., Rangapuram, S. S., Benidis, K., Bohlke-Schneider, M., Kurle, R., Stella, L., Hasson, H., Gallinari, P., and Januschowski, T. (2020). Normalizing Kalman filters for multivariate time series analysis. *Advances in Neural Information Processing Systems*, 33.

Dheeru, D. and Karra Taniskidou, E. (2017). UCI machine learning repository.

Eisenach, C., Patel, Y., and Madeka, D. (2020). Mqtransformer: Multi-horizon forecasts with context dependent and feedback-aware attention. *arXiv preprint arXiv:2009.14799*.

Gasthaus, J., Benidis, K., Wang, Y., Rangapuram, S. S., Salinas, D., Flunkert, V., and Januschowski, T. (2019). Probabilistic forecasting with spline quantile function rnns. In *The 22nd international conference on artificial intelligence and statistics*, pages 1901–1910. PMLR.

Gneiting, T. and Raftery, A. E. (2007). Strictly proper scoring rules, prediction, and estimation. *Journal of the American statistical Association*, 102(477):359–378.

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*.

Hallin, M., del Barrio, E., Cuesta-Albertos, J., and Matrán, C. (2021). Distribution and quantile functions, ranks and signs in dimension d: A measure transportation approach. *The Annals of Statistics*, 49(2):1139 – 1165.

Hanasusanto, G. A., Kuhn, D., Wallace, S. W., and Zymler, S. (2015). Distributionally robust multi-item newsvendor problems with multimodal demand distributions. *Math. Program.*, 152(1-2):1–32.

Huang, C.-W., Chen, R. T. Q., Tsirigotis, C., and Courville, A. (2021). Convex potential flows: Universal probability distributions with optimal transport and convex optimization. In *International Conference on Learning Representations*.

Jaini, P., Selby, K. A., and Yu, Y. (2019). Sum-of-squares polynomial flow. In Chaudhuri, K. and Salakhutdinov, R., editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 3009–3018. PMLR.

Jin, X., Park, Y., Maddix, D. C., Wang, H., and Wang, Y. (2022). Domain adaptation for time series forecasting via attention sharing.

Kass, R. E. and Vos, P. W. (2011). *Geometrical foundations of asymptotic inference*, volume 908. John Wiley & Sons.

Kim, J., Park, Y., Fox, J. D., Boyd, S. P., and Dally, W. (2020). Optimal operation of a plug-in hybrid vehicle with battery thermal and degradation model. In *2020 American Control Conference (ACC)*, pages 3083–3090. IEEE.

Knott, M. and Smith, C. S. (1984). On the optimal mapping of distributions. *Journal of Optimization Theory and Applications*, 43(1):39–49.

Kobyzev, I., Prince, S. J., and Brubaker, M. A. (2021). Normalizing flows: An introduction and review of current methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(11):3964–3979.

Koenker, R. (2005). *Quantile Regression*. Econometric Society Monographs. Cambridge University Press.

Koenker, R. and Bassett, G. (1978). Regression quantiles. *Econometrica*, 46(1):33–50.

Koltchinskii, V. I. (1997). M-estimation, convexity and quantiles. *The Annals of Statistics*, 25(2):435 – 477.

Li, S., Jin, X., Xuan, Y., Zhou, X., Chen, W., Wang, Y.-X., and Yan, X. (2019). Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Lim, B., Arif, S., Loeff, N., and Pfister, T. (2021). Temporal fusion transformers for interpretable multi-horizon time series forecasting. *International Journal of Forecasting*, 37(4):1748–1764.

Liu, D. C. and Nocedal, J. (1989). On the limited memory BFGS method for large scale optimization. *Mathematical programming*, 45(1):503–528.

Lu, Y., Park, Y., Chen, L., Wang, Y., De Sa, C., and Foster, D. (2021). Variance reduced training with stratified sampling for forecasting models. 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 7145–7155. PMLR.

Makkuva, A., Taghvaei, A., Oh, S., and Lee, J. (2020). Optimal transport mapping via input convex neural networks. In *International Conference on Machine Learning*, pages 6672–6681. PMLR.

Makridakis, S. et al. (2018). The M4 competition: Results, findings, conclusion and way forward. *International Journal of Forecasting*, 34(4):802–808.

Matheson, J. E. and Winkler, R. L. (1976). Scoring rules for continuous probability distributions. *Management Science*, 22(10):1087–1096.

Papamakarios, G., Pavlakou, T., and Murray, I. (2017). Masked autoregressive flow for density estimation. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Park, Y., Maddix, D., Aubet, F.-X., Kan, K., Gasthaus, J., and Wang, Y. (2021). Learning quantile functions without quantile crossing for distribution-free time series forecasting. *arXiv preprint arXiv:2111.06581*.

Park, Y., Mahadik, K., Rossi, R. A., Wu, G., and Zhao, H. (2019). Linear quadratic regulator for resource-efficient cloud services. In *Proceedings of the ACM Symposium on Cloud Computing*, pages 488–489.

Park, Y., Rossi, R., Wen, Z., Wu, G., and Zhao, H. (2020). Structured policy iteration for linear quadratic regulator. In *International Conference on Machine Learning*, pages 7521–7531. PMLR.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, 32:8024–8035.

Petropoulos, F., Apiletti, D., Assimakopoulos, V., Babai, M. Z., Barrow, D. K., Taieb, S. B., Bergmeir, C., Bessa, R. J., Bijak, J., Boylan, J. E., Browell, J., Carnevale, C., Castle, J. L., Cirillo, P., Clements, M. P., Cordeiro, C., Oliveira, F. L. C., Baets, S. D., Dokumentov, A., Ellison, J., Fiszeder, P., Franses, P. H., Frazier, D. T., Gilliland, M., Gönül, M. S., Goodwin, P., Grossi, L., Grushka-Cockayne, Y., Guidolin, M., Guidolin, M., Gunter, U., Guo, X., Guseo, R., Harvey, N., Hendry, D. F., Hollyman, R., Januschowski, T., Jeon, J., Jose, V. R. R., Kang, Y., Koehler, A. B., Kolassa, S., Kourentzes, N., Leva, S., Li, F., Litsiou, K., Makridakis, S., Martin, G. M., Martinez, A. B., Meeran, S., Modis, T., Nikolopoulos, K., Önkal, D., Paccagnini, A., Panagiotelis, A., Panapakidis, I., Pavía, J. M., Pedio, M., Pedregal, D. J., Pinson, P., Ramos, P., Rapach, D. E., Reade, J. J., Rostami-Tabar, B., Rubaszek, M., Sermpinis, G., Shang, H. L., Spiliotis, E., Syntetos, A. A., Talagala, P. D., Talagala, T. S., Tashman, L., Thomakos, D., Thorarinsdottir, T., Todini, E., Arenas, J. R. T., Wang, X., Winkler, R. L., Yusupova, A., and Ziel, F. (2021). Forecasting: theory and practice.

Peyré, G., Cuturi, M., et al. (2019). Computational optimal transport: With applications to data science. *Foundations and Trends in Machine Learning*, 11(5-6):355–607.

Rajaram, K. and Tang, C. S. (2001). The impact of product substitution on retail merchandising. *European Journal of Operational Research*, 135(3):582–601.

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

Rasul, K., Seward, C., Schuster, I., and Vollgraf, R. (2021). Autoregressive denoising diffusion models for multivariate probabilistic time series forecasting.

Rasul, K., Sheikh, A.-S., Schuster, I., Bergmann, U., and Vollgraf, R. (2020). Multivariate probabilistic time series forecasting via conditioned normalizing flows. *arXiv preprint arXiv:2002.06103*.

Ruthotto, L. and Haber, E. (2021). An introduction to deep generative modeling. *GAMM-Mitteilungen*, 44(2):e202100008.

Salinas, D., Bohlke-Schneider, M., Callot, L., Medico, R., and Gasthaus, J. (2019). High-dimensional multivariate forecasting with low-rank gaussian copula processes. *Advances in neural information processing systems*, 32.

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.

Székely, G. J. (2003). E-statistics: The energy of statistical samples. *Bowling Green State University, Department of Mathematics and Statistics Technical Report*, 3(05):1–18.

Tabak, E. G. and Turner, C. V. (2013). A family of nonparametric density estimation algorithms. *Communications on Pure and Applied Mathematics*, 66(2):145–164.

Tagasovska, N. and Lopez-Paz, D. (2019). Single-model uncertainties for deep learning. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Uria, B., Murray, I., and Larochelle, H. (2013). Rnade: The real-valued neural autoregressive density-estimator. In Burges, C. J. C., Bottou, L., Welling, M., Ghahramani, Z., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc.

Villani, C. (2009). *Optimal transport: old and new*. Springer.

Wang, J., Sun, S., and Yu, Y. (2019). Multivariate triangular quantile maps for novelty detection. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Wei, Y. (2008). An approach to multivariate covariate-dependent quantile contours with application to bivariate conditional growth charts. *Journal of the American Statistical Association*, 103(481):397–409.

Wen, R., Torkkola, K., Narayanaswamy, B., and Madeka, D. (2017). A multi-horizon quantile recurrent forecaster. In *NIPS 2017 Time Series Workshop*.

Zhang, R.-Q., Zhang, L.-K., Zhou, W.-H., Saigal, R., and Wang, H.-W. (2014). The multi-item newsvendor model with cross-selling and the solution when demand is jointly normally distributed. *European Journal of Operational Research*, 236(1):147–159.

Supplementary Material: Multivariate Quantile Function Forecaster

A EXPERIMENTAL DETAILS

A.1 Datasets

Table 2 lists the information of the datasets used in the experiments. The datasets are available in the GluonTS dataset repository.[4]

DOMAIN	NAME	SUPPORT	FREQ	NO. TS	AVG. LEN	PRED. LEN	NO. COVARIATES
electrical load	`Elec`	$\mathbb{R}^+$	H	321	21044	24	4
road traffic	`Traf`	$[0,1]$	H	862	14036	24	4
M4 forecasting competition	`M4-daily`	$\mathbb{R}^+$	D	4227	2357	14	3
	`M4-weekly`	$\mathbb{R}^+$	W	359	1022	13	2
	`M4-monthly`	$\mathbb{R}^+$	M	48000	216	18	1
	`M4-quarterly`	$\mathbb{R}^+$	Q	24000	92	8	1
	`M4-yearly`	$\mathbb{R}^+$	Y	23000	31	6	0

Table 2: Summary of dataset statistics, where `Elec` and `Traf` are dervied from the UCI data repository (Dheeru and Karra Taniskidou, 2017), and `M4` are competition datasets (Makridakis et al., 2018).

A.2 Hyperparameters

The hyperparameters used in the experiments are listed in Table 3. For the RNN parameters we use the default setting of the `DeepAREstimator` in the GluonTS package (Alexandrov et al., 2020). The other hyperparameters were selected by performing a grid search only on the `Elec` dataset, and were used as default values on all the other datasets.

TYPE	HYPERPARAMETER	VALUE
RNN	layers	2
	nodes	40
PICNN	layers	5
	nodes	40
Energy Score	num. of samples	50
Training	epochs	100 / 300
	batch size	32

Table 3: Summary of hyperparameters. For the number of training epochs, 100 is used for DeepAR and MQCNN, and 300 is used for MQF^2. This is because DeepAR and MQCNN have already converged after 100 epochs and MQF^2 takes more epochs to converge.

B DEFINITION OF EVALUATION METRICS

Consider the target value $z_{i,t}$ for the i-th time series at time t, where $i = 1, ..., m$ and $t = T+1, ..., T+\tau$, and the corresponding predictions $\{\hat{z}_{i,t,s}\}_{s=1}^{S}$ from S sample paths. We denote the α-quantile of the predictions as $\hat{z}_{i,t}^{\alpha}$.

[4] `https://github.com/awslabs/gluon-ts/blob/master/src/gluonts/dataset/repository/datasets.py`.

B.1 Mean Weighted Quantile Loss

The α-quantile loss is defined as

$$\rho_\alpha(z_{i,t}, \hat{z}^\alpha_{i,t}) = (z_{i,t} - \hat{z}^\alpha_{i,t})(\alpha - \mathbf{1}\{z_{i,t} - \hat{z}^\alpha_{i,t} < 0\}).$$

The mean weighted quantile loss is defined as

$$\text{mean\_wQL} = \frac{1}{|\mathcal{A}|} \sum_{\alpha \in \mathcal{A}} \frac{\sum_{i=1}^m \sum_{t=T}^{T+\tau} 2\rho_\alpha(z_{i,t}, \hat{z}^\alpha_{i,t})}{\sum_{i=1}^m \sum_{t=T}^{T+\tau} |z_{i,t}|},$$

where $\mathcal{A}$ is a set of prespecified quantile levels. In our experiments, we used $\mathcal{A} = \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$.

B.2 n-th Step Mean Weighted Quantile Loss

The mean weighted quantile loss at the n-th step is defined as

$$\text{mean\_wQL}(n) = \frac{1}{|\mathcal{A}|} \sum_{\alpha \in \mathcal{A}} \frac{\sum_{i=1}^m 2\rho_\alpha(z_{i,n}, \hat{z}^\alpha_{i,n})}{\sum_{i=1}^m |z_{i,n}|}.$$

B.3 Sum CRPS

The sum CRPS is the (approximated) CRPS for the sum of the predictions over the prediction horizon and defined as

$$\text{sum\_CRPS} = \frac{1}{m} \sum_{i=1}^m \left(-\frac{1}{2|S|^2} \sum_{j=1}^S \sum_{k=1}^S |\hat{u}_{i,j} - \hat{u}_{i,k}| + \frac{1}{|S|} \sum_{j=1}^S |\hat{u}_{i,j} - u_i| \right),$$

where $\hat{u}_{i,j} = \sum_t \hat{z}_{i,t,j}$ and $u_i = \sum_t u_{i,t}$.

B.4 Mean Scaled Interval Score

The mean scaled interval score (MSIS) is defined as

$$\text{MSIS}(\zeta) = \frac{1}{\text{SE}(z)} \Big(\frac{1}{m\tau} \sum_{i=1}^m \sum_{t=T+1}^{T+\tau} (\hat{z}^{\alpha_U}_{i,t} - \hat{z}^{\alpha_L}_{i,t} + \frac{2}{\zeta}[(\hat{z}^{\alpha_L}_{i,t} - z_{i,t})\mathbf{1}\{z_{i,t} < \hat{z}^{\alpha_L}_{i,t}\} + (z_{i,t} - \hat{z}^{\alpha_U}_{i,t})\mathbf{1}\{z_{i,t} > \hat{z}^{\alpha_U}_{i,t}\}] \Big),$$

where the upper quantile $\alpha_U = 1 - \zeta/2$, and the lower quantile $\alpha_L = \zeta/2$. The seasonal error SE for time series frequency f is given by

$$\text{SE}(z) = \frac{1}{m(T-f)} \sum_{i=1}^m \sum_{t'=1}^{T-f} |z_{i,t'} - z_{i,t'+f}|.$$

C PROOF OF PROPOSITION 1

Here, we state Proposition 1 again and provide the proof.

Proposition 1. *Let $\mathcal{D} \subseteq \mathbb{R}^n$ be open, $G : \mathcal{D} \to \mathbb{R}$ be a strongly convex and smooth function and g be its gradient. Then g^{-1} exists and is the gradient of a convex function.*

Proof. The strong convexity and smoothness of G implies the existence of g^{-1} and that $\nabla g(\mathbf{x})$ is symmetric positive definite (SPD) for all $\mathbf{x} \in \mathcal{D}$. Since g is one-to-one, smooth, and $\nabla g(\mathbf{x})$ is SPD for all $\mathbf{x} \in \mathcal{D}$, by Kass and Vos (2011, Corollary A.2), g^{-1} is also smooth and therefore $\nabla g^{-1}(g(\mathbf{x}))\nabla g(\mathbf{x}) = \mathbf{I}_n$ for all $\mathbf{x} \in \mathcal{D}$. This implies $\nabla g^{-1}(\mathbf{y})$ is SPD for all $\mathbf{y} \in g(\mathcal{D})$ and hence

$$\frac{\partial [g^{-1}]_i}{\partial y_j} = \frac{\partial [g^{-1}]_j}{\partial y_i} \quad \text{for all } i, j = 1, 2, ..., n. \tag{15}$$

Let $\psi(\mathbf{y}) = \sum_{j=1}^{n} y_j \int_0^1 [g^{-1}]_j(t\mathbf{y})dt$. Consider its partial derivative

$$\frac{\partial \psi}{\partial y_i}(\mathbf{y}) = \int_0^1 [g^{-1}]_i(t\mathbf{y})dt + \int_0^1 \sum_{j=1}^{n} y_j t \frac{\partial [g^{-1}]_j}{\partial y_i}(t\mathbf{y})dt$$

using (15), we get

$$= \int_0^1 [g^{-1}]_i(t\mathbf{y})dt + \int_0^1 \sum_{j=1}^{n} y_j t \frac{\partial [g^{-1}]_i}{\partial y_j}(t\mathbf{y})dt$$

applying the chain rule $\frac{\partial}{\partial t}\Big([g^{-1}]_i(t\mathbf{y})\Big) = t\frac{\partial [g^{-1}]_i}{\partial y_j}(t\mathbf{y})$, we obtain

$$= \int_0^1 [g^{-1}]_i(t\mathbf{y})dt + \int_0^1 t\frac{\partial}{\partial t}\Big([g^{-1}]_i(t\mathbf{y})\Big)dt$$

performing integration by parts, we have

$$\begin{aligned} &= \int_0^1 [g^{-1}]_i(t\mathbf{y})dt + t[g^{-1}]_i(t\mathbf{y})\big|_{t=0}^{t=1} - \int_0^1 [g^{-1}]_i(t\mathbf{y})dt \\ &= [g^{-1}]_i(\mathbf{y}), \quad \text{for} \quad i = 1, 2, ..., n. \end{aligned}$$

Therefore, $\nabla\psi = g^{-1}$. Moreover ψ is convex because its Hessian $\nabla g^{-1}(\mathbf{y})$ is SPD for all $\mathbf{y} \in g(\mathcal{D})$. Therefore, g^{-1} is the gradient of the convex function ψ. □

D ADDITIONAL RESULTS TABLE

Dataset	Model	Metrics over full horizon				Mean Quantile Loss over differing forecast horizon				
		sum CRPS	Energy score	MSIS	mean_wQL	1 step	5 steps	10 steps	15 steps	20 steps
Elec	MQCNN	2323.5 ± 54.2	1282.1 ± 1.0	11.7 ± 0.0	0.086 ± 0.0	0.042 ± 0.0	0.125 ± 0.01	0.117 ± 0.02	0.094 ± 0.01	**0.062 ± 0.0**
	DeepAR	3059.6 ± 180.8	971.7 ± 59.0	7.3 ± 0.1	0.07 ± 0.0	**0.027 ± 0.0**	**0.055 ± 0.0**	**0.059 ± 0.0**	0.074 ± 0.0	0.089 ± 0.01
	MQF^2+ ES	**1723.7 ± 122.8**	**891.1 ± 32.1**	**6.9 ± 0.1**	**0.066 ± 0.0**	0.031 ± 0.0	0.08 ± 0.01	0.102 ± 0.0	0.056 ± 0.0	0.068 ± 0.01
	MQF^2+ ML	2332.523 ± 146.88	893.6 ± 53.8	7.2 ± 0.6	**0.066 ± 0.0**	0.038 ± 0.01	0.088 ± 0.02	0.073 ± 0.01	**0.053 ± 0.0**	0.067 ± 0.01
Traf	MQCNN	0.419 ± 0.33	0.161 ± 0.06	46.1 ± 1.6	0.993 ± 0.29	0.905 ± 0.4	5.909 ± 0.37	0.878 ± 0.42	0.871 ± 0.23	0.644 ± 0.18
	DeepAR	0.108 ± 0.01	0.061 ± 0.0	7.2 ± 0.1	0.131 ± 0.0	**0.074 ± 0.0**	**0.163 ± 0.0**	**0.117 ± 0.0**	0.123 ± 0.0	0.144 ± 0.0
	MQF^2+ ES	**0.095 ± 0.0**	**0.06 ± 0.0**	7.2 ± 0.0	0.142 ± 0.0	0.104 ± 0.0	0.298 ± 0.01	0.139 ± 0.01	0.127 ± 0.0	**0.139 ± 0.01**
	MQF^2+ ML	0.097 ± 0.0	0.062 ± 0.0	**6.6 ± 0.1**	**0.13 ± 0.0**	0.078 ± 0.0	0.165 ± 0.01	0.13 ± 0.0	**0.12 ± 0.0**	0.14 ± 0.0
M4-daily	MQCNN	3089.8 ± 10.4	923.1 ± 3.6	41.9 ± 0.9	0.027 ± 0.0	**0.009 ± 0.0**	0.019 ± 0.0	**0.024 ± 0.0**	-	-
	DeepAR	3186.2 ± 966.5	989.3 ± 244.3	50.5 ± 8.0	0.039 ± 0.01	0.015 ± 0.0	0.028 ± 0.01	0.049 ± 0.01	-	-
	MQF^2+ ES	**1752.0 ± 47.3**	**619.2 ± 8.7**	31.1 ± 0.3	**0.024 ± 0.0**	0.013 ± 0.0	**0.019 ± 0.0**	0.027 ± 0.0	-	-
	MQF^2+ ML	1786.028 ± 60.94	622.0 ± 14.7	**30.5 ± 0.3**	**0.024 ± 0.0**	0.01 ± 0.0	0.019 ± 0.0	0.029 ± 0.0	-	-
M4-weekly	MQCNN	3572.45 ± 104.73	1463.4 ± 18.8	62.3 ± 3.2	0.065 ± 0.0	0.045 ± 0.0	0.067 ± 0.0	0.072 ± 0.0	-	-
	DeepAR	2885.141 ± 443.69	1166.9 ± 119.6	26.0 ± 4.5	0.054 ± 0.01	**0.034 ± 0.0**	**0.053 ± 0.0**	0.062 ± 0.01	-	-
	MQF^2 + ES	2831.64 ± 175.06	1122.6 ± 30.6	**21.5 ± 1.5**	**0.052 ± 0.0**	0.043 ± 0.0	**0.053 ± 0.0**	0.056 ± 0.0	-	-
	MQF^2 + ML	**2577.461 ± 107.32**	**1107.9 ± 32.5**	26.1 ± 1.8	**0.052 ± 0.0**	0.039 ± 0.0	0.054 ± 0.0	**0.054 ± 0.0**	-	-
M4-monthly	MQCNN	9196.6 ± 175.4	3269.9 ± 34.6	18.7 ± 0.4	0.12 ± 0.0	0.072 ± 0.0	0.096 ± 0.0	0.115 ± 0.0	**0.134 ± 0.0**	-
	DeepAR	**7337.0 ± 345.5**	2572.1 ± 95.0	14.0 ± 1.5	0.113 ± 0.0	0.063 ± 0.0	0.092 ± 0.0	0.115 ± 0.0	0.143 ± 0.01	-
	MQF^2+ ES	7365.7 ± 218.1	**2554.6 ± 79.3**	**12.8 ± 1.4**	**0.112 ± 0.0**	**0.059 ± 0.0**	**0.087 ± 0.0**	**0.113 ± 0.0**	0.145 ± 0.0	-
	MQF^2+ ML	8235.445 ± 0.0	2839.7 ± 0.0	14.4 ± 0.0	0.124 ± 0.0	0.066 ± 0.0	0.1 ± 0.0	0.124 ± 0.0	0.159 ± 0.0	-
M4-quarterly	MQCNN	3348.365 ± 53.53	1718.6 ± 26.4	15.6 ± 1.6	0.09 ± 0.0	0.059 ± 0.0	0.096 ± 0.0	-	-	-
	DeepAR	3184.673 ± 46.2	1575.3 ± 34.2	15.0 ± 2.7	**0.085 ± 0.0**	**0.051 ± 0.0**	**0.092 ± 0.0**	-	-	-
	MQF^2 + ES	**3134.404 ± 235.3**	**1533.2 ± 81.5**	11.8 ± 0.5	**0.085 ± 0.0**	0.054 ± 0.0	**0.092 ± 0.01**	-	-	-
	MQF^2 + ML	3338.119 ± 135.57	1591.5 ± 47.0	**11.7 ± 0.8**	0.088 ± 0.0	0.053 ± 0.0	0.095 ± 0.0	-	-	-
M4-yearly	MQCNN	3753.7 ± 28.1	1976.2 ± 12.8	**34.2 ± 0.3**	**0.115 ± 0.0**	**0.064 ± 0.0**	0.141 ± 0.0	-	-	-
	DeepAR	3749.1 ± 42.6	1917.1 ± 7.9	34.9 ± 0.6	0.118 ± 0.0	0.065 ± 0.0	0.145 ± 0.0	-	-	-
	MQF^2+ ES	**3649.3 ± 60.9**	**1859.4 ± 28.1**	36.7 ± 1.6	0.116 ± 0.0	0.075 ± 0.0	**0.135 ± 0.0**	-	-	-
	MQF^2+ ML	3784.486 ± 105.76	1913.2 ± 35.2	38.8 ± 2.1	0.119 ± 0.0	0.07 ± 0.0	0.143 ± 0.0	-	-	-

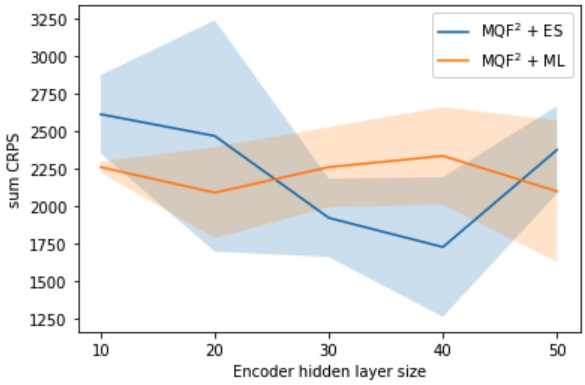
Table 4: Results (with additional datasets) of MQF^2 compared with other state of the art methods (for all columns lower is better.) We show the mean and standard deviation over 3 training runs. A "-" indicates that the corresponding time step is beyond the prediction length of the dataset.

Dataset	Model	Point forecast metrics			Probabilistic metrics			
		MASE	sMAPE	NRMSE	wQL 0.1	wQL 0.5	wQL 0.9	MAE coverage
Elec	MQCNN	1.179 ± 0.0	0.168 ± 0.0	0.843 ± 0.0	0.055 ± 0.0	0.107 ± 0.0	0.055 ± 0.0	**0.034 ± 0.0**
	DeepAR	0.95 ± 0.0	0.129 ± 0.0	0.757 ± 0.0	**0.034 ± 0.0**	**0.082 ± 0.0**	0.049 ± 0.0	0.205 ± 0.0
	MQF^2 + ES	1.568 ± 1.143	0.177 ± 0.093	1.157 ± 0.953	0.077 ± 0.073	0.153 ± 0.127	0.072 ± 0.051	0.074 ± 0.011
	MQF^2 + ML	**0.918 ± 0.051**	**0.121 ± 0.004**	**0.647 ± 0.037**	0.036 ± 0.002	0.083 ± 0.004	**0.045 ± 0.004**	0.105 ± 0.008
Traf	MQCNN	3.155 ± 0.0	0.998 ± 0.0	0.892 ± 0.0	0.864 ± 0.0	0.655 ± 0.0	2.197 ± 0.0	0.454 ± 0.0
	DeepAR	**0.598 ± 0.0**	0.157 ± 0.0	0.417 ± 0.0	0.071 ± 0.0	**0.156 ± 0.0**	**0.107 ± 0.0**	0.046 ± 0.0
	MQF^2 + ES	0.667 ± 0.014	0.2 ± 0.002	**0.407 ± 0.004**	0.074 ± 0.002	0.171 ± 0.003	0.118 ± 0.002	**0.038 ± 0.013**
	MQF^2 + ML	0.604 ± 0.004	**0.156 ± 0.001**	0.415 ± 0.003	**0.064 ± 0.001**	0.156 ± 0.001	0.111 ± 0.002	0.046 ± 0.025
M4-daily	MQCNN	3.892 ± 0.0	0.035 ± 0.0	0.108 ± 0.0	0.021 ± 0.0	0.032 ± 0.0	0.016 ± 0.0	**0.03 ± 0.0**
	DeepAR	4.256 ± 0.0	0.038 ± 0.0	0.107 ± 0.0	0.022 ± 0.0	0.034 ± 0.0	0.017 ± 0.0	0.036 ± 0.0
	MQF^2 + ES	3.76 ± 0.125	0.035 ± 0.001	0.103 ± 0.001	0.018 ± 0.0	0.031 ± 0.001	0.014 ± 0.001	0.079 ± 0.022
	MQF^2 + ML	**3.584 ± 0.134**	**0.034 ± 0.001**	**0.102 ± 0.002**	**0.017 ± 0.0**	**0.029 ± 0.001**	**0.013 ± 0.0**	0.077 ± 0.047
M4-weekly	MQCNN	3.463 ± 0.0	0.098 ± 0.0	0.133 ± 0.0	0.039 ± 0.0	0.069 ± 0.0	0.057 ± 0.0	0.082 ± 0.0
	DeepAR	3.362 ± 0.0	0.093 ± 0.0	0.127 ± 0.0	0.029 ± 0.0	0.069 ± 0.0	0.037 ± 0.0	0.133 ± 0.0
	MQF^2 + ES	**3.005 ± 0.16**	0.09 ± 0.005	**0.12 ± 0.002**	**0.026 ± 0.001**	0.066 ± 0.002	0.039 ± 0.004	0.072 ± 0.034
	MQF^2 + ML	3.135 ± 0.19	**0.087 ± 0.004**	0.124 ± 0.004	0.027 ± 0.001	**0.065 ± 0.002**	**0.035 ± 0.001**	**0.06 ± 0.019**
M4-monthly	MQCNN	1.217 ± 0.0	0.146 ± 0.0	0.306 ± 0.0	0.103 ± 0.0	0.133 ± 0.0	0.092 ± 0.0	**0.091 ± 0.0**
	DeepAR	1.255 ± 0.0	**0.145 ± 0.0**	**0.294 ± 0.0**	**0.069 ± 0.0**	**0.13 ± 0.0**	0.084 ± 0.0	0.1 ± 0.0
	MQF^2 + ES	**1.141 ± 0.022**	0.149 ± 0.002	0.306 ± 0.006	0.073 ± 0.004	0.133 ± 0.002	**0.083 ± 0.005**	0.097 ± 0.018
	MQF^2 + ML	1.29 ± 0.0	0.165 ± 0.0	0.327 ± 0.0	0.08 ± 0.0	0.146 ± 0.0	0.098 ± 0.0	0.092 ± 0.0
M4-quarterly	MQCNN	1.64 ± 0.0	0.122 ± 0.0	0.244 ± 0.0	0.055 ± 0.0	0.116 ± 0.0	0.064 ± 0.0	0.147 ± 0.0
	DeepAR	**1.306 ± 0.0**	**0.108 ± 0.0**	**0.233 ± 0.0**	**0.049 ± 0.0**	**0.101 ± 0.0**	0.06 ± 0.0	**0.026 ± 0.0**
	MQF^2 + ES	1.364 ± 0.092	0.112 ± 0.005	0.235 ± 0.006	0.05 ± 0.003	0.104 ± 0.006	0.059 ± 0.004	0.067 ± 0.029
	MQF^2 + ML	1.444 ± 0.08	0.12 ± 0.004	0.244 ± 0.005	0.053 ± 0.003	0.11 ± 0.003	**0.058 ± 0.002**	0.073 ± 0.035
M4-yearly	MQCNN	3.358 ± 0.0	**0.14 ± 0.0**	**0.286 ± 0.0**	0.086 ± 0.0	**0.134 ± 0.0**	**0.09 ± 0.0**	0.132 ± 0.0
	DeepAR	**3.235 ± 0.0**	0.14 ± 0.0	0.295 ± 0.0	**0.066 ± 0.0**	0.138 ± 0.0	0.103 ± 0.0	**0.059 ± 0.0**
	MQF^2 + ES	3.442 ± 0.137	0.146 ± 0.005	0.292 ± 0.005	0.071 ± 0.008	0.141 ± 0.003	0.1 ± 0.007	0.108 ± 0.024
	MQF^2 + ML	3.507 ± 0.124	0.15 ± 0.004	0.295 ± 0.004	0.071 ± 0.004	0.143 ± 0.003	0.093 ± 0.003	0.083 ± 0.028

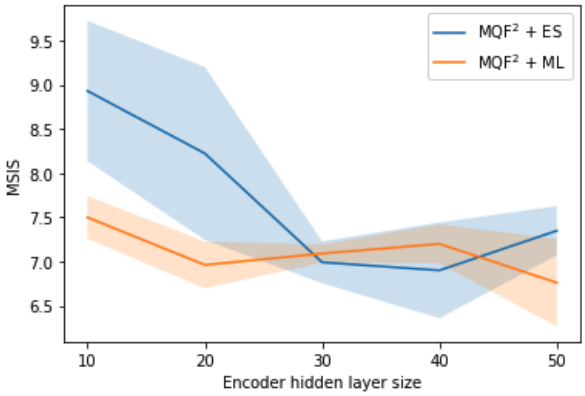
Table 5: Results (with additional metrics) of MQF^2 compared with other state of the art methods. We show the mean and standard deviation over 3 training runs. A "-" indicates that the corresponding time step is beyond the prediction length of the dataset.

E ROBUSTNESS EXPERIMENTS

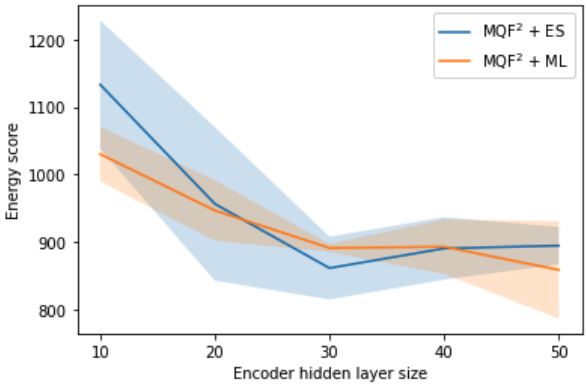
To show that our MQF^2 is robust with respect to its hyperparameters, we perform experiments with varying encoder hidden state size and ICNN hidden layer size. We perform 3 training runs, and the mean and standard deviation over the runs are reported in Figures 4-5. We observe that the performance of our MQF^2 is steady when the hyperparameters are changing. We also see that the standard deviations are small relative to the mean. In particular, the magnitudes of the standard deviations are about 15% of that of the means. When the encoder hidden state and ICNN hidden layer sizes increase, the performance gets slightly better in general.



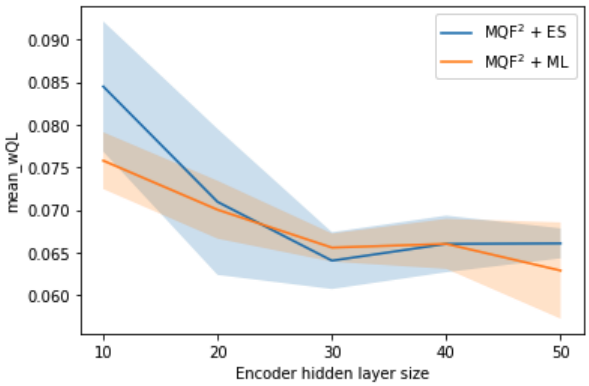
(a) sum CRPS as we vary the hidden state size



(b) MSIS as we vary the hidden state size

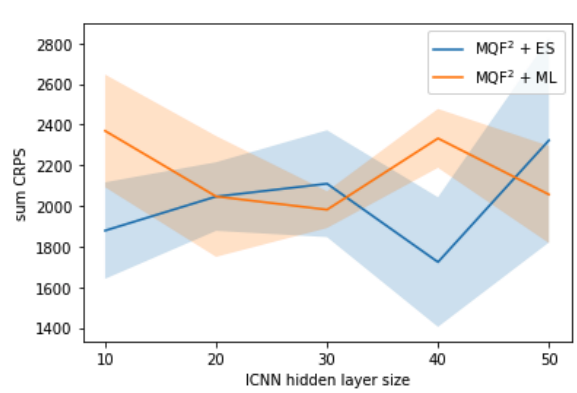


(c) ES as we vary the hidden state size

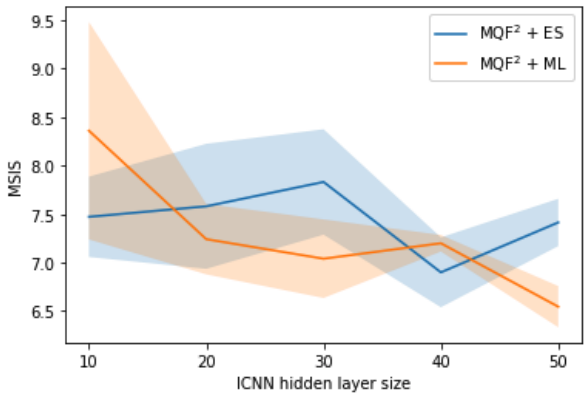


(d) wQL as we vary the hidden state size

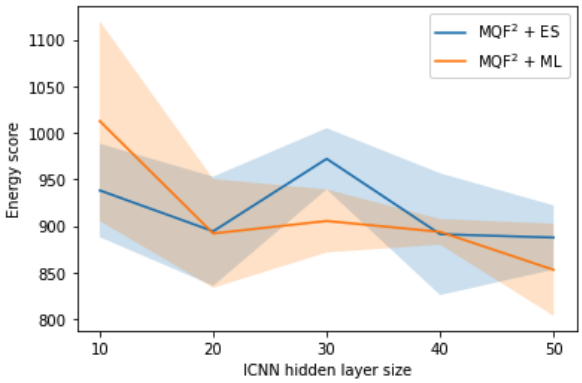
Figure 4: To test the robustness of our MQF^2, we investigate the influence of the size of the encoder's hidden state on its performance. We report the sum CRPS, MSIS, Energy Score and mean weighted quantile loss when different hidden state sizes are used. The experiments are repeated 3 times. The solid lines represent the mean of the results, and the colored regions represent the range of 1 standard deviation.



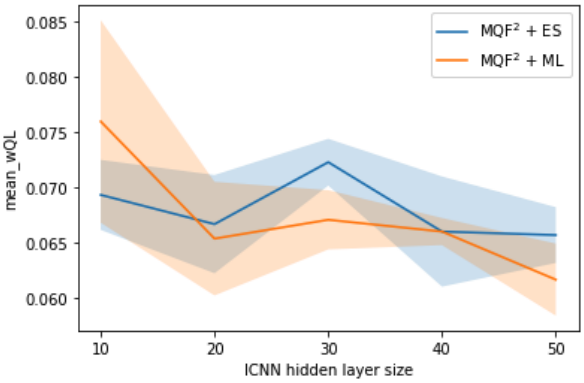
(a) sum CRPS as we vary the hidden layer size



(b) MSIS as we vary the hidden layer size



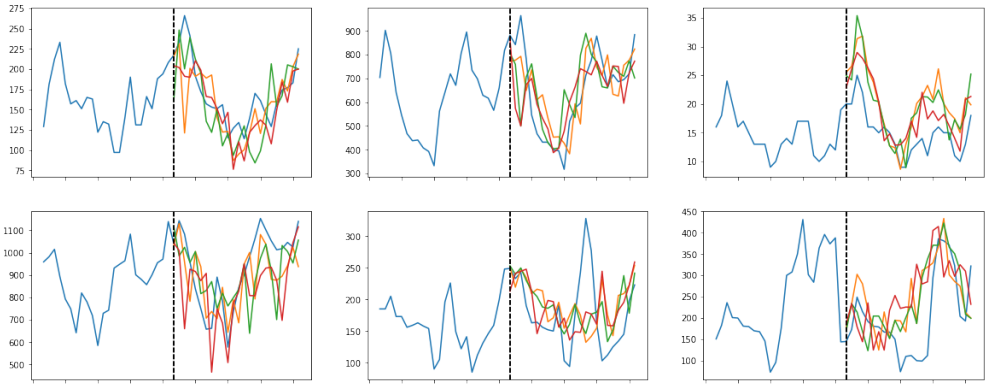
(c) ES as we vary the hidden layer size



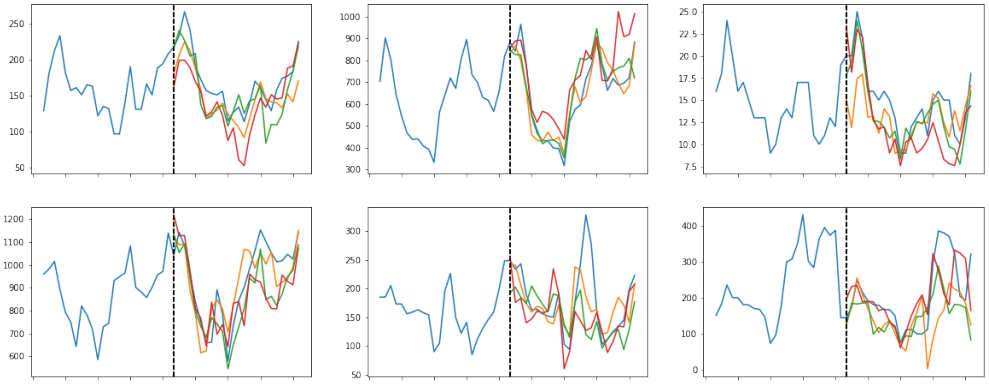
(d) wQL as we vary the hidden layer size

Figure 5: To test the robustness of our MQF^2, we investigate the influence of the width of the ICNN on its performance. We report the sum CRPS, MSIS, Energy Score and mean weighted quantile loss when different hidden layer sizes are used. The experiments are repeated 3 times. The solid lines represent the mean of the results, and the colored regions represent the range of 1 standard deviation.

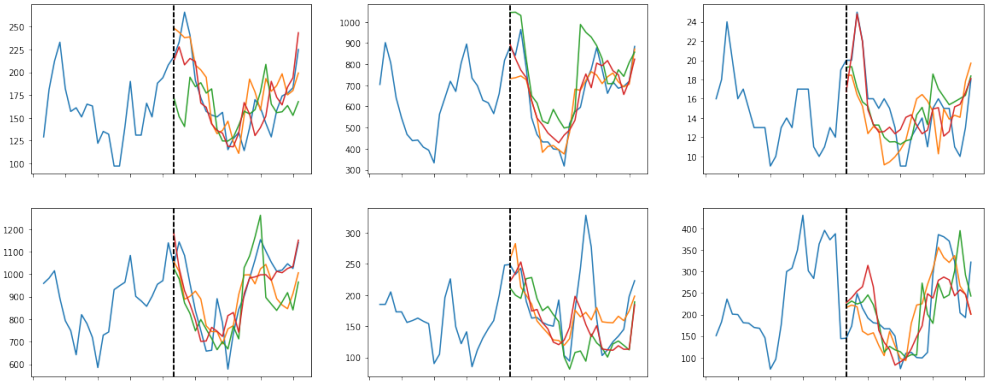
F SAMPLE PATH FIGURES



(a) MQCNN



(b) DeepAR



(c) MQF^2

Figure 6: Sample paths generated by MQCNN, DeepAR and MQF^2. Three sample paths are generated for each of the 6 time series of the `Elec` dataset. The dotted vertical lines represent the start of the prediction horizon.