Deep Quantile Aggregation

Taesup Kim [1] **Rasool Fakoor** [*1] **Jonas Mueller** [*1] **Ryan J. Tibshirani** [1 2] **Alexander J. Smola** [1]

Abstract

Conditional quantile estimation is a key statistical learning challenge motivated by the need to quantify uncertainty in predictions or to model a diverse population without being overly reductive. As such, many models have been developed for this problem. Adopting a *meta*-viewpoint, we propose a general framework for aggregating any number of conditional quantile models in order to produce superior estimates. We propose weighted ensembling strategies of greater flexibility than previously considered, where aggregation weights may vary over not only individual models, but also feature values and (pairs of) quantile levels. We produce these weights using deep neural networks that adaptively combine arbitrary quantile estimates in a non-crossing manner, enforced through a simple monotonization layer in the network. Our approach is widely applicable and can leverage the full modern deep learning toolkit for building quantile ensembles.

1. Introduction

Estimation of conditional quantiles has a rich history in econometrics/statistics (Koenker, 2005) and is widely-used as a tool for uncertainty quantification (Schaumburg, 2012) or characterizing heterogeneous outcomes across a diverse population (Austin et al., 2005). This paper considers the task of combining any number of conditional quantile models into a unified estimator. Done properly, such *model aggregation* can substantially improve on the accuracy of the individual base models (Dietterich, 2000).

To fix ideas, let $Y \in \mathbb{R}$ be a response variable of interest, and let $X \in \mathcal{X}$ denote input feature variables we use to predict Y. While uncertainty quantification can be achieved as a byproduct of estimating the conditional distribution of $Y|X = x$, this may be nontrivial, especially in high dimensions (when $\mathcal{X} = \mathbb{R}^d$, and d is large). A simpler goal is to estimate *conditional quantiles* of $Y|X = x$ over a discrete set of quantile levels $\mathcal{T} \subseteq [0, 1]$, that is, to estimate $g(x; \tau) = F^{-1}_{Y|X=x}(\tau)$ for $\tau \in \mathcal{T}$. Here for a random variable Z with cumulative distribution function (CDF) F_Z, we denote its level-τ quantile by $F^{-1}_Z(\tau) = \inf\{z : F_Z(z) \geq \tau\}$. For example, one might choose $\mathcal{T} = \{0.01, 0.02, \ldots, 0.99\}$ to finely characterize the conditional distribution $Y|X = x$.

Problem Definition. Assume that we are given a collection of p conditional quantile models $\{\hat{g}_j\}_{j=1}^p$ with respect to a discrete set of quantile levels in $\mathcal{T}$. Each $\hat{g}_j$, which we call a *base model*, is an estimate of the true conditional quantile function g. It is convenient to view g and each $\hat{g}_j$ as functions from $\mathcal{X}$ to $\mathbb{R}^m$, where $m = |\mathcal{T}|$ is the number of quantile levels, with the components of $g(x)$ denoted $g(x; \tau)$ for $\tau \in \mathcal{T}$ and similarly for $\hat{g}_j(x)$. Throughout, we carefully distinguish multiple estimates of the same quantile τ from different base models $\{\hat{g}_j\}_{j=1}^p$ vs. estimates of multiple quantiles $\tau_1, \ldots, \tau_m$ from one base model $\hat{g}_j$. Given $p \times m$ estimates of m quantile levels by p base models, our main goal is to form an optimal ensemble estimate $\hat{g} = H(\hat{g}_1, \ldots, \hat{g}_p) : \mathcal{X} \to \mathbb{R}^m$.

In this paper, we mainly focus on linear aggregation procedures H, though we allow the weights in these linear combinations to be themselves functions of input feature values x, as in

$$\hat{g}(x) = \sum_{j=1}^{p} w_j(x) \cdot \hat{g}_j(x), \quad x \in \mathcal{X}. \tag{1}$$

We propose different aggregation strategies in which each weight $w_j(x)$ is either a scalar, vector, or matrix (and the product "$\cdot$" between $w_j(x)$ and $\hat{g}_j(x)$ is to be interpreted accordingly). A standard approach would be to use global weights (constant over x) and to fit these by optimizing (say) the pinball loss (3) of the ensemble over the training data (Tibshirani, 2020; Nowotarski & Weron, 2015). This paper demonstrates that allowing the weights to vary over x (and particular quantile-pairs) produces more adaptive ensembles with clear practical benefits, and such weights can be conveniently optimized using modern deep learning toolkits.

[*]Equal contribution [1]Amazon Web Services [2]Carnegie Mellon Univeristy. Correspondence to: Taesup Kim <taesup@amazon.com>.

Presented at the ICML 2021 Workshop on Uncertainty and Robustness in Deep Learning.,

2. Methods

In what follows, we introduce various weighted aggregation strategies for combining multiple conditional quantile model into a single unified model. More flexible aggregation strategies can account for finer-grained strengths/weaknesses in constituent models, provided there is enough data to statistically identify these.

Cross-Validated Ensembling. To avoid overfitting in our ensemble, we utilize a standard cross-validation scheme that only uses out-of-fold predictions from the base models when assigning ensemble weights (Van der Laan et al., 2007; Erickson et al., 2020). Here we randomly partition the training data $\{(x_i, y_i)\}_{i=1}^n$ into K disjoint folds (all our experiments use $K = 5$). For each data point (x_i, y_i) in a given fold k, we retrain each base model on the other folds $\{1, \ldots, K\} \setminus \{k\}$ in order to obtain an out-of-sample prediction at x_i, denoted $\hat{g}_j^{-i}(x_i)$ (the superscript indicates that we have trained on folds that exclude the ith data point). When learning the ensemble weight functions $\{w_j(x)\}_{j=1}^p$, we similarly consider out-of-sample ensemble predictions:

$$\hat{g}(x_i) = \sum_{j=1}^{p} w_j(x_i) \cdot \hat{g}_j^{-i}(x_i), \quad i = 1, \ldots, n. \tag{2}$$

Once the weight functions have been learned, to make predictions at new test points $x \in \mathcal{X}$, we use (1), where the base models $\{\hat{g}_j\}_{j=1}^p$ have been fit to the full training set $\{(x_i, y_i)\}_{i=1}^n$.

Weighted Ensembling Strategies. For simplicity, we temporarily assume $w_j(x) = w_j$ (the case where the weights depend on x is analogous). Often aggregators are restricted to a convex combination of base models ($\sum w_j = 1$), both for simplicity/robustness and to limit extra variance in predictions introduced by aggregator-training (Ratcliff, 1979; Lichtendahl et al., 2013; Dietterich, 2000; Caruana et al., 2004; Raftery et al., 2005). Within this framework, we describe various weighted ensembling strategies of increasing flexibility (named akin to coffee grind sizes).

- ***Coarse***: For each base model $\hat{g}_j$, a single weight w_j is allocated, and it is shared over all quantile levels $\mathcal{T}$. With p base models, a *Coarse* aggregator learns or predicts p weights, where each weight w_j is a scalar in $\mathbb{R}$ with $\sum_{j=1}^p w_j = 1$. The product "$\cdot$" in the summands of (1) is interpreted as a scalar-vector product. This is the standard way to generally build weighted ensembles (Dietterich, 2000), but has clear limitations when done in quantile-space. For instance: if all base model predictions belong to the same location-scale family, then Coarse aggregator predictions will be restricted to the same family (Genest, 1992; Thomas & Ross, 1980), and also predictions of the Coarse aggregator will generally have sharper tails than those of the base models (Lichtendahl et al., 2013).
- ***Medium***: For each base model $\hat{g}_j$ and each quantile level τ, a separate weight w_j^τ is allocated. With p base models and m quantile levels of interest, a *Medium* aggregator learns or predicts $p \times m$ weights, where each w_j is a vector in $\mathbb{R}^m$ with $\sum_{j=1}^p w_j^\tau = 1$ for each τ. The product "$\cdot$" in (1) is interpreted as the Hadamard (elementwise) product between vectors. This strategy enables the ensemble to account for the fact that different base models may be better/worse at predicting certain quantile levels.
- ***Fine***: For each quantile level τ, a separate weight $w_j^{\tau,\nu}$ is allocated to the estimate of a quantile level $\nu \in \mathcal{T}$ output by each base model. With p base models and m quantile levels of interest, a *Fine* aggregator learns or predicts by $p \times m^2$ weights. Thus each w_j is actually a matrix in $\mathbb{R}^{m \times m}$, with $\sum_{j=1}^p \sum_{\nu \in \mathcal{T}} w_j^{\tau,\nu} = 1$ for each τ. The product "$\cdot$" in (1) is interpreted as a matrix-vector product. This strategy presumes that for a given quantile level τ, base model estimates for other levels $\nu (\neq \tau)$ could also be useful for aggregation purposes.

Global versus Local Aggregation. *Global* aggregation weights do not depend on feature values x of the input data so that the weights are globally shared across all inputs. This is how weighted ensembles are traditionally defined in the quantile aggregation literature (Nowotarski & Weron, 2018; Brooks et al., 2020). On the other hand, *Local* aggregation weights refer to a heterogeneous aggregation strategy in which the weights are allowed to vary based on the feature values x. This is akin to a *mixture of experts* ensemble (Jacobs et al., 1991), which has not been previously considered for conditional quantile estimation.

Deep Quantile Aggregation. Note that any combination of *Coarse/Medium/Fine* and *Global/Local* can be used for aggregation of quantile models (denoted *Global-Medium, Local-Fine*, etc.). In particular, neither *Fine* nor *Local* settings have been previously considered for quantile ensembling. Our experiments show that the flexible Local-Fine aggregator, when trained using standard deep learning optimization techniques, generally performs substantially better than other aggregators. We subsequently refer to our proposed neural network (NN) implementation of a Local-Fine aggregator as *Deep Quantile Aggregation* (DQA).

2.1. Fitting Global Aggregation Weights

Given a convex loss function $\mathcal{L}$ acting on quantiles, optimizing the weights with respect to $\mathcal{L}$ for any Global aggregator can be cast as a convex program. We will restrict our attention henceforth to a loss $\mathcal{L}$ defined by summing the *pinball loss* (Koenker & Hallock, 2001) across quantile levels $\tau \in \mathcal{T}$. The pinball loss is defined at a given quantile level τ as

$$l_\tau(y, \hat{y}) = \max\{\tau(y - \hat{y}), (\tau - 1)(y - \hat{y})\}. \tag{3}$$

For continuous random variable Y, the expected pinball loss $\mathbb{E}[l_\tau(Y,a)]$ is minimized at the level-τ quantile $a = F_Y^{-1}(\tau)$, which allows us to estimate quantiles even just observing a single sample from $Y|X=x$ for each x in the data. Given training data $\{(x_i,y_i)\}_{i=1}^n$, we fit ensemble weights by solving the optimization problem:

$$\min_{w\geq 0} \sum_{\tau\in\mathcal{T}} \sum_{i=1}^{n} l_\tau\big(y_i, \hat{g}(x_i;\tau)\big) \text{ subject to } Aw = 1. \quad (4)$$

Here linear operator A encodes the unit-sum constraint, with say $Aw = \sum_{j=1}^p \sum_{\nu\in\mathcal{T}} w_j^{\tau,\nu}$ for Fine aggregation.

A key challenge when estimating multiple quantile levels is to ensure that these estimates are in appropriate monotonic order. In the quantile regression literature, this has been addressed in a model-specific manner (Takeuchi et al., 2006; Dette & Volgushev, 2008), or via separate post-processing (Chernozhukov et al., 2010; Kuleshov et al., 2018; Song et al., 2019). To ensure non-crossing in our aggregated quantile estimates, we can add the following constraints to the optimization problem (4):

$$\hat{g}(x;\tau) \leq \hat{g}(x;\tau'), \quad \tau < \tau',\ x \in \mathcal{D}_x, \quad (5)$$

where $\mathcal{D}_x$ is a set of feature values over which we enforce non-crossing. By default we can just use the training feature values $\mathcal{D}_x = \{x_i\}_{i=1}^n$.

One way to fit Global aggregation weights is via linear programming (LP) (Tibshirani, 2020). However the LP is computationally intensive and is empirically less accurate than the technique we propose here. Rather than utilizing LP solvers, we propose to learn aggregation weights via stochastic gradient descent (SGD) methods.

2.2. Local Aggregation with Neural Networks

Here we introduce a *Local* aggregator, which utilizes a neural network to account for the fact that quantile estimates from the base models may be better/worse at different feature values of $x \in \mathcal{X}$. For concreteness, we describe this approach for the case of Fine weights, but the same idea applies to Medium aggregation as well.

To fit weight functions of the form $w_j^{\tau,\nu} : \mathcal{X} \to \mathbb{R}$, we model these functions jointly over all base models $j \in [1,\dots,p]$ and pairs of quantile levels $(\tau,\nu) \in \mathcal{T}\times\mathcal{T}$ using a neural network G. All but the last layer of G forms a feature extractor $f_\theta : \mathcal{X} \to \mathbb{R}^d$ that maps x into a hidden vector representation $h \in \mathbb{R}^d$. In our experiments, we take f_θ to be a standard feed-forward network parameterized by θ. The last layer of G is defined, for each $\tau \in \mathcal{T}$, by a projection matrix $W^\tau \in \mathbb{R}^{(p\times m)\times d}$ that maps the hidden representation h, followed by a softmax operation, to local probability weights $w^\tau(x) = [w^{\tau\nu_1}(x),\dots,w^{\tau\nu_m}(x)] \in \mathbb{R}^{p\times m}$, as in

$$w^\tau(x) = G(x;\theta,W^\tau) = \text{softmax}\big(W^\tau f_\theta(x)\big)$$

Parameters θ and $\{W^\tau\}_{\tau\in\mathcal{T}}$ can be easily optimized via standard SGD methods applied to objective (8) as in the Global setting. This reflects a *mixture of experts* ensemble (Jacobs et al., 1991), in which gating network G emits predictions from the experts it believes will be most accurate for a particular x. In our most flexible Local-Fine setting (also called DQA), the experts correspond to estimates of individual quantiles from individual base models, and we fit a separate gating scheme for each target quantile level τ.

2.3. Enforcing Non-crossing Quantile Estimates

Monotonizer. To avoid the non-crossing constraints (5), we propose a *monotonizer* operator that fixes crossing quantile estimates in one sweep. For a given feature value x and quantile level τ_k (indexed in increasing order: $\tau_1 < \cdots < \tau_m$), the monotonizer $\mathcal{M}$ applied to $\hat{g}$ returns $\tilde{g}$ defined by

$$\tilde{g}(x;\tau_k) = \begin{cases} \max\{\hat{g}(x;\tau_k), \tilde{g}(x;\tau_{k-1})\} & \tau_k > 0.5 \\ \hat{g}(x;\tau_k) & \tau_k = 0.5 \\ \min\{\hat{g}(x;\tau_k), \tilde{g}(x;\tau_{k+1})\} & \tau_k < 0.5. \end{cases} \quad (6)$$

By construction, this ensures $\tilde{g}(x;\tau) \leq \tilde{g}(x;\tau')$ for $\tau \leq \tau'$. Our monotonizer may be used a as static post-processing layer, or as an dynamic layer used within optimization whose effects are accounted for during learning. For each group of quantiles above/below the median, the monotonization transform defined above is sequentially applied in an outwards sweep over the quantile levels in succession. Just like the piecewise linear pinball loss, our monotonizer is differentiable almost everywhere, thus when we use it as a layer during optimization, we can we can backpropogate through it to obtain the necessary gradients for SGD. This is hugely beneficial in practice, as non-crossing can be enforced at basically no extra computational cost.

Crossing Penalty. While our monotonizer suffices to ensure non-crossing, it can crudely perturb the original (crossing) aggregated quantile estimates. We additionally introduce a *crossing penalty* during training that smoothly guides the original estimates to be less nonmonotonic, such that the perturbation from subsequent monotonization is less severe. This leads to stabler optimization and superior estimates. For a fixed margin $\delta > 0$, our crossing penalty at $x \in \mathcal{X}$ is

$$\rho(\hat{g}(x)) = \sum_{\tau<\tau'} \max\{\hat{g}(x;\tau) - \hat{g}(x;\tau') + \delta, 0\}. \quad (7)$$

Putting these together, our ultimate objective to fit aggregation weights via SGD is given by

$$\sum_{\tau\in\mathcal{T}} \sum_{i=1}^{n} l_\tau\big(y_i, \tilde{g}(x_i;\tau)\big) + \gamma \sum_{x\in\mathcal{D}_x} \rho(\hat{g}(x)). \quad (8)$$

Here we compute the pinball loss (3) of monotonized estimates $\tilde{g}$ and the crossing penalty for original estimates

	Method \ Dataset	yacht (308)	boston (506)	energy (768)	concrete (1,030)	kin8nm (8,192)	power (9,568)	naval (11,934)	protein (45,730)
Base Models	CGN (Lakshminarayanan et al., 2017)	0.00549	0.08087	0.01714	0.07256	0.07190	0.06235	0.04128	0.18046
	SQR (Tagasovska & Lopez-Paz, 2019)	0.00716	0.08240	0.01522	0.08072	0.07329	0.06319	0.02522	0.16828
	MQR(Ours)	0.00616	0.07531	0.01385	0.07238	0.06979	0.06119	0.01376	0.15701
	RandomForest (Meinshausen, 2006)	0.01353	0.08797	0.01538	0.08218	0.14889	0.04988	0.01675	0.13222
	ExtraTrees (Geurts et al., 2006)	0.01515	0.08728	0.01538	0.09249	0.15731	0.05461	0.02130	0.14601
	LightGBM (Ke et al., 2017)	0.00891	0.09679	0.00999	0.06645	0.13943	0.05051	0.02071	0.16083
	LinearRegression (Koenker et al., 2017)	0.13026	0.12437	N/A	0.17752	0.21694	0.07425	N/A	0.23387
Quantile Aggregators	Uniform Average (Genre et al., 2013)	0.01989	0.07578	0.01264	0.07511	0.10566	0.05426	0.01744	0.15620
	QRA (Nowotarski & Weron, 2015)	0.00551	N/A	0.00921	0.07775	0.06903	0.04742	N/A	0.13073
	FQRA (Maciejowska et al., 2016)	0.02185	0.16828	0.01264	0.10388	0.07431	0.04652	0.01269	0.12877
	Global-Coarse (Ratcliff, 1979)	0.00545	**0.07452**	0.00968	0.05959	**0.06813**	0.04733	0.01314	0.13213
	Global-Medium (LP) (Tibshirani, 2020)	0.00543	0.07555	0.01322	0.06954	0.06814	0.06008	0.01400	N/A
	Global-Medium (SGD, ie. Ours)	0.00526	0.07488	0.00972	0.05956	0.06821	0.04716	0.01341	0.13211
	DQA (Ours, ie. Local-Fine)	**0.00508**	0.07537	**0.00825**	**0.05413**	0.06839	**0.04411**	**0.01132**	**0.12533**

Table 1. **Average pinball loss achieved by various methods on UCI datasets (lower is better)**. The sample size of each dataset is indicated in parentheses. Entries with N/A indicate the model either not converged or prohibitively slow.

pre-monotonization. The crossing penalty is merely an auxiliary objective that helps guide SGD training, which may otherwise be hampered by the monotonizer's perturbations of our iterates.

3. Experiments

We provide an empirical comparison of our proposed quantile aggregation procedures as well as many different established regression/aggregation algorithms. Experiments reported in the main text are mainly over a set of 8 regression datasets from UCI (Dua & Graff, 2017). Throughout we estimate $\mathcal{T} = \{0.01, 0.02, ..., 0.99\}$ as our target quantile levels of interest $(m = 99)$.

We obtain multiple quantile estimates $\{\hat{g}_j\}_{j=1}^{p}$ using $p = 7$ different types of base models. Each base model predicts all quantile levels in $\mathcal{T}$. We consider three neural network models with different architectures/objectives: 1) conditional Gaussian network (CGN) (Lakshminarayanan et al., 2017), 2) simultaneous quantile regression (SQR) (Tagasovska & Lopez-Paz, 2019), 3) our multi-quantile regression (MQR) model from Appendix D.2, which is simply a fully-connected neural network that maps x to vector $\{\hat{g}(x;\tau)\}_{\tau\in\mathcal{T}}$ and is trained via SGD applied to our (monotized) objective in (8). We also consider three competitive decision tree variants adapted for quantile regression: 4) random forest (Meinshausen, 2006), 5) extremely randomized trees (ExtraTrees) (Geurts et al., 2006), 6) LightGBM gradient boosting (Ke et al., 2017). Our final base model is standard: 7) linear quantile regression (Koenker et al., 2017).

Established baseline quantile aggregators we consider include a Global-Coarse aggregator, as well as QRA (Nowotarski & Weron, 2015) and FQRA (Maciejowska et al., 2016). We employ QRA as it was applied by Lima & Meng (2017), which is similar to our Global-Medium aggregator (but without restricting ensemble weights to be convex). FQRA further employs PCA prior to weighted aggregation (Maciejowska et al., 2016), which enables the extension of QRA to Global-Fine aggregation that would otherwise suffer from severe collinearity.

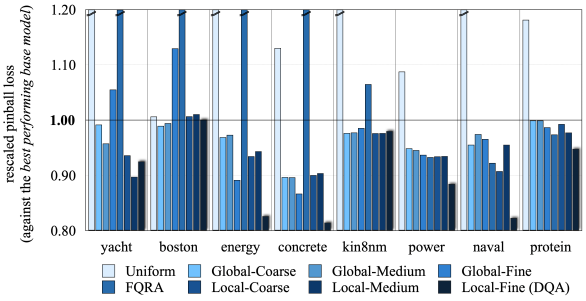

Figure 1. **Quantile aggregators ablation analysis.** Pinball loss results are scaled against the best performing base model (shown at 1.0 on the y-axis).

Results. Following Gasthaus et al. (2019); Chung et al. (2020), we adopt the average pinball loss as a metric to evaluate our quantile estimates. Table 1 shows that DQA typically outperforms the individual base quantile regression models and other published aggregation methods. Our Global-Medium aggregators outperform (unconstrained linear aggregators) QRA/FQRA, highlighting the utility of our unit sum constraint on ensemble weights. We also perform comprehensive ablation analyses to study our proposed weighted aggregation strategies of increasing flexibility. While the simplest form of quantile aggregation, a *uniform* (unweighted) average, is not more accurate than best base model on each dataset, Figure 1 illustrates that Local aggregators outperform not only the base models but also the Global aggregators on the vast majority of datasets. Among Local aggregators, our Local-Fine (a.k.a DQA) variant tends to produce the best estimates.

4. Conclusion

This paper proposes a general adaptive framework for aggregating any number of conditional quantile models to produce superior estimates. Moreover, simple backpropagation-compatible techniques are presented to ensure non-crossing estimates, which can both be utilized in both our aggregators and standalone neural quantile regression models.

References

Austin, P. C., Tu, J. V., Daly, P. A., and Alter, D. A. The use of quantile regression in health care research: a case study examining gender differences in the timeliness of thrombolytic therapy. *Statistics in medicine*, 24(5):791–816, 2005.

Bowman, V., Silk, D., Dalrymple, U., and Woods, D. Uncertainty quantification for epidemiological forecasts of covid-19 through combinations of model predictions. *arXiv preprint arXiv:2006.10714*, 2020.

Brooks, L. C., Ray, E. L., Bien, J., Bracher, J., Rumack, A., Tibshirani, R. J., and Reich, N. G. Comparing ensemble approaches for short-term probabilistic covid-19 forecasts in the us. *International Institute of Forecasters*, 2020.

Browell, J., Gilbert, C., Tawn, R., and May, L. Quantile combination for the eem20 wind power forecasting competition. In *2020 17th International Conference on the European Energy Market (EEM)*, pp. 1–6, 2020.

Busetti, F. Quantile aggregation of density forecasts. *Oxford Bulletin of Economics and Statistics*, 79(4):495–512, 2017.

Caruana, R., Niculescu-Mizil, A., Crew, G., and Ksikes, A. Ensemble selection from libraries of models. In *Proceedings of the twenty-first international conference on Machine learning*, pp. 18, 2004.

Chernozhukov, V., Fernández-Val, I., and Galichon, A. Quantile and probability curves without crossing. *Econometrica*, 78(3):1093–1125, 2010.

Chung, Y., Neiswanger, W., Char, I., and Schneider, J. Beyond pinball loss: Quantile methods for calibrated uncertainty quantification. *arXiv preprint arXiv: 2011.09588*, 2020.

Cumings-Menon, R. and Shin, M. Probability forecast combination via entropy regularized wasserstein distance. *Entropy*, 22(9), 2020. ISSN 1099-4300.

Dette, H. and Volgushev, S. Non-crossing non-parametric estimates of quantile curves. *Journal of the Royal Statistical Society: Series B*, 70(3):609–627, 2008.

Dietterich, T. G. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, 2000.

Dua, D. and Graff, C. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

Erickson, N., Mueller, J., Shirkov, A., Zhang, H., Larroy, P., Li, M., and Smola, A. J. AutoGluon-Tabular: Robust and accurate AutoML for structured data. *arXiv preprint arXiv:2003.06505*, 2020.

Gaillard, P., Goude, Y., and Nedellec, R. Additive models and robust aggregation for GEFCom2014 probabilistic electric load and electricity price forecasting. *International Journal of forecasting*, 32(3):1038–1050, 2016.

Gasthaus, J., Benidis, K., Wang, Y., Rangapuram, S. S., Salinas, D., Flunkert, V., and Januschowski, T. Probabilistic forecasting with spline quantile function RNNs. In *International Conference on Artificial Intelligence and Statistics*, 2019.

Genest, C. Vincentization revisited. *Annals of Statistics*, 20 (2):1137–1142, 1992.

Genre, V., Kenny, G., Meyler, A., and Timmermann, A. Combining expert forecasts: Can anything beat the simple average? *International Journal of Forecasting*, 29(1):108–121, 2013.

Geurts, P., Ernst, D., and Wehenkel, L. Extremely randomized trees. *Machine Learning*, pp. 3–42, 2006.

Gneiting, T. and Katzfuss, M. Probabilistic forecasting. *Annual Review of Statistics and Its Application*, 1:125–151, 2014.

Gneiting, T. and Ranjan, R. Combining predictive distributions. *Electronic Journal of Statistics*, 7:1747–1782, 2013.

Jacobs, R. A., Jordan, M. I., Nowlan, S. J., and Hinton, G. E. Adaptive mixtures of local experts. *Neural Computation*, 3:79–87, 1991.

Kapetanios, G., Mitchell, J., Price, S., and Fawcett, N. Generalised density forecast combinations. *Journal of Econometrics*, 188(1):150–165, 2015.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., and Liu, T.-Y. LightGBM: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems*, 2017.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. 2015. International Conference for Learning Representations.

Koenker, R. *Quantile Regression*. Cambridge University Press, 2005.

Koenker, R. and Hallock, K. F. Quantile regression. *Journal of economic perspectives*, 15(4):143–156, 2001.

Koenker, R., Chernozhukov, V., He, X., and Peng, L. *Handbook of Quantile Regression*. CRC press, 2017.

Kuleshov, V., Fenner, N., and Ermon, S. Accurate uncertainties for deep learning using calibrated regression. In *International Conference on Machine Learning*, 2018.

Lakshminarayanan, B., Pritzel, A., and Blundell, C. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, 2017.

Lichtendahl, K. C., Grushka-Cockayne, Y., and Winkler, R. L. Is it better to average probabilities or quantiles? *Management Science*, 59(7):1594–1611, 2013.

Lima, L. R. and Meng, F. Out-of-sample return predictability: A quantile combination approach. *Journal of Applied Econometrics*, 32(4):877–895, 2017.

Lopez, J. A. and Diebold, F. Forecast evaluation and combination. *Federal Reserve Bank of New York, Research Paper*, (9525), 1995.

Maciejowska, K. and Nowotarski, J. A hybrid model for gefcom2014 probabilistic electricity price forecasting. *International Journal of Forecasting*, 32(3):1051–1056, 2016.

Maciejowska, K., Nowotarski, J., and Weron, R. Probabilistic forecasting of electricity spot prices using factor quantile regression averaging. *International Journal of Forecasting*, 32(3):957–965, 2016.

Meinshausen, N. Quantile regression forests. *Journal of Machine Learning Research*, 7(35):983–999, 2006.

Nowotarski, J. and Weron, R. Computing electricity spot price prediction intervals using quantile regression and forecast averaging. *Computational Statistics*, 30(3):791–803, 2015.

Nowotarski, J. and Weron, R. Recent advances in electricity price forecasting: A review of probabilistic forecasting. *Renewable and Sustainable Energy Reviews*, 81:1548–1568, 2018. ISSN 1364-0321.

Raftery, A. E., Gneiting, T., Balabdaoui, F., and Polakowski, M. Using bayesian model averaging to calibrate forecast ensembles. *Monthly Weather Review*, 133(5):1155–1174, 2005.

Ranjan, R. and Gneiting, T. Combining probability forecasts. *Journal of the Royal Statistical Society: Series B*, 72(1):71–91, 2010.

Rasp, S. and Lerch, S. Neural networks for postprocessing ensemble weather forecasts. *Monthly Weather Review*, 146(11):3885–3900, 2018.

Ratcliff, R. Group reaction time distributions and an analysis of distribution statistics. *Psychological Bulletin*, 86(3):446–461, 1979.

Schaumburg, J. Predicting extreme value at risk: Nonparametric quantile regression with refinements from extreme value theory. *Computational Statistics & Data Analysis*, 56(12):4081–4096, 2012.

Schepen, A. and Wang, Q. Model averaging methods to merge operational statistical and dynamic seasonal streamflow forecasts in a ustralia. *Water Resources Research*, 51(3):1797–1812, 2015.

Smyl, S. and Hua, N. G. Machine learning methods for gefcom2017 probabilistic load forecasting. *International Journal of Forecasting*, 35(4):1424–1431, 2019.

Song, H., Diethe, T., Kull, M., and Flach, P. Distribution calibration for regression. In *International Conference on Machine Learning*, 2019.

Stone, M. The opinion pool. *Annals of Mathematical Statistics*, 32(4):1339–1342, 1961.

Tagasovska, N. and Lopez-Paz, D. Single-model uncertainties for deep learning. In *Advances in Neural Information Processing Systems*, 2019.

Takeuchi, I., Le, Q. V., Sears, T. D., and Smola, A. J. Nonparametric quantile estimation. *Journal of Machine Learning Research*, 7:1231–1264, 2006.

Thomas, E. A. C. and Ross, B. H. On appropriate procedures for combining probability within the same family. *Journal of Mathematical Psychology*, 21:136–152, 1980.

Tibshirani, R. J. quantgen: Tools for generalized quantile modeling, 2020. URL https://github.com/ryantibs/quantgen.

Timmermann, A. Forecast combinations. *Handbook of economic forecasting*, 1:135–196, 2006.

Uniejewski, B. and Weron, R. Regularized quantile regression averaging for probabilistic electricity price forecasting. *Energy Economics*, 95:105121, 2021.

Van der Laan, M. J., Polley, E. C., and Hubbard, A. E. Super learner. *Statistical applications in genetics and molecular biology*, 6(1), 2007.

Zhang, S., Wang, Y., Zhang, Y., Wang, D., and Zhang, N. Load probability density forecasting by transforming and combining quantile forecasts. *Applied Energy*, 277:115600, 2020. ISSN 0306-2619.

Appendix

A. Related Work

As a reliable way to improve predictions, model ensembling has been widely studied for aggregation of point estimates (Dietterich, 2000; Van der Laan et al., 2007; Caruana et al., 2004) or distributional estimates (Stone, 1961; Raftery et al., 2005; Ranjan & Gneiting, 2010; Gneiting & Ranjan, 2013; Gneiting & Katzfuss, 2014; Rasp & Lerch, 2018; Cumings-Menon & Shin, 2020; Kapetanios et al., 2015). The latter is of particular interest in *forecast combination* (Timmermann, 2006), where small performance gains can have immense value in fields like finance (Lopez & Diebold, 1995). Here previous empirical comparisons have found that quantile aggregation often produces superior distributional predictions than density aggregation (Busetti, 2017; Lichtendahl et al., 2013; Gaillard et al., 2016; Nowotarski & Weron, 2018; Browell et al., 2020; Bowman et al., 2020; Smyl & Hua, 2019; Schepen & Wang, 2015).

For quantile aggregation, Nowotarski & Weron (2015) proposed *quantile regression averaging* (QRA), in which a linear quantile regression model (Koenker & Hallock, 2001) is fit on top the collective base model predictions as its features. Variants of QRA have since been developed with regularization (Uniejewski & Weron, 2021), additional input features (Maciejowska & Nowotarski, 2016), and dimensionality reduction via Factor QRA (FQRA) (Maciejowska et al., 2016) which applies PCA to reduce collinearity in the base model outputs before fitting the QRA regressor. Other existing quantile aggregation strategies have mostly been basic weighted ensembles (Ratcliff, 1979; Lichtendahl et al., 2013; Nowotarski & Weron, 2018; Bowman et al., 2020; Browell et al., 2020; Zhang et al., 2020; Brooks et al., 2020; Tibshirani, 2020), none of which are nearly as rich as the Fine/Local aggregation strategies proposed in this paper. As perhaps evidence for a dearth of performant quantile aggregation methods, embarrassingly simple mean/median aggregation has been shown to outperform existing strategies (Genre et al., 2013; Brooks et al., 2020), and leading solutions in recent distributional forecasting competitions have simply used basic variations of our so-called Coarse quantile aggregation (Gaillard et al., 2016; Smyl & Hua, 2019; Browell et al., 2020).

B. Global Aggregation via Linear Programming (LP)

Here we formally state the full LP formulation of the optimization problem that could be solved to obtain aggregation weights (in this case for our Global-Medium aggregator):

$$
\begin{aligned}
\underset{w,u}{\text{minimize}} \quad & \sum_{\tau\in\mathcal{T}}\sum_{i=1}^{n} u_i^\tau \\
\text{subject to} \quad & u_i^\tau \geq \tau\left(y_i - \sum_{j=1}^{p} w_j^\tau \hat{g}_j\left(x_i;\tau\right)\right) \quad \forall (x_i,y_i)\in\mathcal{D}_n \text{ and } \forall\tau\in\mathcal{T}, \\
& u_i^\tau \geq (\tau-1)\left(y_i - \sum_{j=1}^{p} w_j^\tau \hat{g}_j\left(x_i;\tau\right)\right) \quad \forall (x_i,y_i)\in\mathcal{D}_n \text{ and } \forall\tau\in\mathcal{T}, \\
& \sum_{j=1}^{m} w_j^\tau = 1, w_j^\tau \geq 0 \quad \forall\tau\in\mathcal{T}, \\
& \sum_{j=1}^{p} w_j^{\tau_k}\hat{g}_j\left(x_{i'};\tau_k\right) \leq \sum_{j=1}^{p} w_j^{\tau_{k+1}}\hat{g}_j\left(x_{i'};\tau_{k+1}\right) \quad \forall x_{i'}\in\mathcal{D}_x \text{ and } \forall\tau_k,\tau_{k+1}\in\mathcal{T}
\end{aligned}
\tag{9}
$$

Functionality for setting up and solving this LP is provided in the `quantgen` R package (Tibshirani, 2020). The LP formulation, while conceptually tidy, has limitations: 1) it would be difficult to use it for Local aggregation without adding significant complexity to the optimization formulation, 2) even for Global aggregation, solving the LP can be computationally inefficient for large datasets, 3) it is nontrivial to mitigate overfitting in the LP optimization of aggregation weights, unlike our SGD procedure which can simply be early stopped.

C. Visualizing Local Weights

In this section, we study how our quantile aggregation gathers information from the given different base models' different quantile estimates. We focus here on local weights, which are designed to adaptively vary for different $x \in \mathcal{X}$, as described in Section 2.2. It is of interest to see how the aggregator's preferences vary between different x. Visually inspecting the weights of our quantile aggregators can thus help us interpret for particular quantiles (and particular x): which particular base estimates (and which of their estimated quantiles in Fine settings) are most useful for accurate estimation.

Local-Medium Aggregator. Here, we inspect the local weights from our Local-Medium aggregator. Figure 2 depicts them as a heatmap visualization. Different heatmaps are generated from different test x, each of size $p \times m$. The columns and rows index different quantile levels τ and base model estimates $\hat{g}_j$, respectively. It is evident that different x, the aggregator differently weights the base models $\hat{g}_j$ and the weights also vary significantly over different quantile levels τ. Some base models appear to be better than others at estimating certain quantile levels for certain x, while these same models produce less useful estimates for other x, τ. Our aggregator is not hampered by the presence of poor base models (e.g. LinearRegression in Table 1) as it can simply allocate more weight to other other models instead (e.g. LinearRegression is consistently the worst base model across all datasets, and our aggregator has learned to discount its negative effect during aggregation with small weights assigned to the bottom row, which corresponds to our LinearRegression base model.

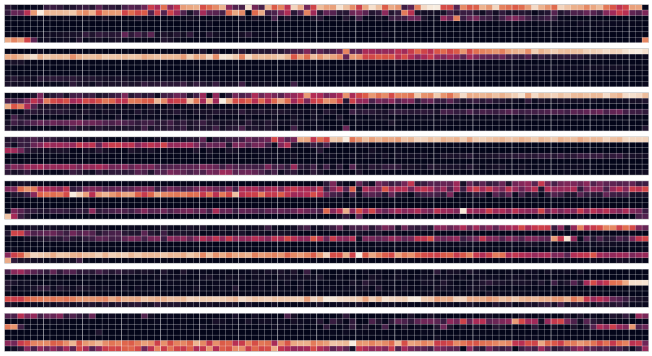

Figure 2. Heatmap visualization of the local weights from the Local-Medium aggregator over the concrete dataset. There are 8 different heatmaps arranged vertically, with each heatmap showing a distinct x (for 8 different datapoints) and how base models (rows) are locally weighed for each quantile level τ_k (columns). Entirely black rows correspond to base models whose predictions are essentially ignored by the aggregator.

Local-Fine Aggregator. Our Local-Fine aggregator has the versatility to consider the interaction between different quantiles. Specifically, it learns to adaptively attend to each quantile from each base model and uses a different weight for each target quantile level τ and each x considering their interactions. To get a better understanding of its mechanism, we visualize Local-Fine aggregator weights, where for each x, we now have a large number of weights ($p \times m \times m$). In particular, we are interested in studying how different target/base quantile levels interact with each other during aggregation. We thus average weights across different inputs x in a dataset and across the different base models $\hat{g}_j$ to obtain a single heatmap of size $m \times m$, which only depicts quantile-quantile interactions. Figure 3 shows such a heatmap for each of two

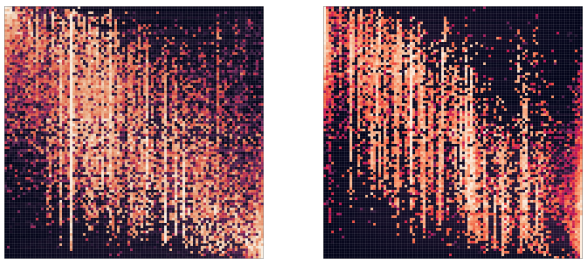

Figure 3. **Heatmap of DQA aggregation weights**, averaged across samples in the *concrete* dataset (on left) and the *power* dataset (on right). For each target quantile level τ (rows), we show weights assigned to base estimates of all quantile levels ν (columns).

different datasets. Here, each row depicts how each target quantile level τ aggregates information across base estimates at different quantile levels ν (columns). Both heatmaps show a common diagonal pattern indicating that the interaction typically occurs between neighboring quantile levels, although the behavior is slightly different in tails (which are presumably harder to estimate).

D. Experiment and Model Details

D.1. Experiment Details

All experiments are conducted 5 times by varying the random seed (e.g. seed=$\{1, 2, 3, 4, 5\}$) which affects the data splits for train/validation/test ($0.72 : 0.18 : 0.10$), hyper-parameter selection, and model initialization. Reported values throughout are an average over these 5 runs. For base models, we search for the best hyper-parameters via cross-validation over 20 randomly selected hyper-parameter configurations. After the best hyper-parameters are decided, out-of-fold estimates from base models are produced for every training data point via 5-fold ($K = 5$) cross-validation (bagging) by properly re-training base models for each fold. The hyper-parameters for aggregation models are also tuned via cross-validation over 20 randomly-selected hyper-parameter configurations (which with our SGD procedure was still much faster than a single run of the LP optimization). All experiments are conducted on Amazon Web Services (AWS) EC2 instances with GPUs.

D.2. Base Models

We obtain multiple quantile estimates $\{\hat{g}_j\}_{j=1}^{p}$ using $p = 7$ different types of base models. Each base model predicts all quantile levels in $\mathcal{T}$. We first consider three neural network models that compute quantile estimates differently. All models are based on a feed-forward network (multilayer perceptron) and are able to predict all quantile levels from a single network.

- **Conditional Gaussian Network** (CGN (Lakshminarayanan et al., 2017)): This model assumes that the conditional distribution $Y|X = x$ is Gaussian. The model therefore outputs estimates of sufficient statistics, namely the conditional mean $\mu(x)$ and variance $\sigma^2(x)$. The model is optimized by minimizing the negative log-likelihood criterion. Subsequent quantile estimation is straightforward that can be computed from $\mathcal{N}\left(\mu(x), \sigma(x)\right)$ in a closed form (e.g. $\hat{g}(x;\tau) = \mu(x) + \sigma(x)\sqrt{2}\text{erf}^{-1}(2\tau - 1)$).
- **Simultaneous Quantile Regression** (SQR (Tagasovska & Lopez-Paz, 2019)): This model takes the target quantile level τ as input along with x and then outputs an estimate of the corresponding level-τ quantile. This enables the SQR model to estimate the entire conditional distribution at any quantile (e.g. $\hat{g}(x;\tau) = f(x||\tau)$, where x and τ are concatenated). The model is trained by minimizing a pinball loss for many different randomly sampled quantile levels τ. To estimate multiple quantile-levels with SQR, we must feed in the same input x numerous times, each paired with a different $\tau \in \mathcal{T}$.
- **Multi-Quantile Regression** (MQR): This is our neural network model that is able to only estimate the pre-defined quantile levels $\mathcal{T}$, but can simultaneously output all estimates of them (e.g. $\hat{g}(x;\tau_k) = f(x)_k$). Our MQR model is simply a standard fully-connected network that takes in x and outputs a vector in $\mathbb{R}^m$ to jointly predict all m quantile levels in $\mathcal{T}$, where non-crossing is easily enforced by incorporating our monotonizer (6) and crossing penalty (7). Like our DQA aggregator, this MQR network is trained via SGD applied to the same objective in (8).

All of these models are optimized using Adam (Kingma & Ba, 2015) with mini-batches of size 64. They also all share the same architecture/optimization hyper-parameter search space: # of layers: $\{2, 3\}$, # of hidden units: $\{64, 128\}$, dropout ratio: $\{0.0, 0.05, 0.1, 0.2\}$, learning rate: $\{$1e-3, 5e-4$\}$, weight decay: $\{$1e-5, 5e-7$\}$. In all settings, we use early-stopping where the validation loss is evaluated every epoch and if it has not decreased for the last 50 epochs, the optimization is stopped by returning the epoch with the lowest validation loss.

We also consider other base models, which are not neural networks:

- **RandomForest** (Meinshausen, 2006) and **ExtraTrees** (Geurts et al., 2006): Both models are based on random forests trained as a regular regression task. After training, quantile estimates are computed via a trick based on the empirical quantile estimates. The leaf node of the trees contains many instances of the target value from which one can obtain empirical quantile estimates. We use `scikit-garden` (`https://scikit-garden.github.io/`) for both models. Both models have the same hyper-parameter search space: # of trees (estimators): $\{100, 1000\}$ ($\{100\}$, for

large datasets $n > 10,000$), minimum # of samples for splitting nodes: {4, 8, 16}, minimum # of sample for leaf nodes: {4, 8, 16}

- **LightGBM** (Geurts et al., 2006): LightGBM (`https://lightgbm.readthedocs.io/`) is a gradient boosting framework that uses tree based base learners. As a boosting framework, it can optimize the pinball loss, and thus be directly used for quantile regression. LightGBM handles multiple quantile levels independently by modeling each with a separate model. The hyper-parameter space we use is: # of boosted trees (estimators): {100, 1000} ({100}, for large datasets $n > 10,000$), # of leaves: {10, 50, 100}, minimum child samples: {3, 9, 15}, minimum child weight: {1e-2, 1e-1, 1}, subsample ratio: {0.4, 0.6, 0.8}, subsample ratio of columns: {0.4, 0.6}, L1 regularization weight: {1e-1, 1, 5}, L2 regularization weight: {1e-1, 1, 5}.

- **LinearRegression** (Koenker et al., 2017): LinearRegression is the standard quantile regression model. This approach handles multiple quantile levels independently by fitting separate linear models for each, and we train the model using the default settings in `statsmodels` (`https://www.statsmodels.org`).

D.3. Aggregation Models

Aggregation models are designed to linearly combine all estimates from our base models. Here, we explain the details on how the models are designed and optimized.

- **Global Aggregators**: Particularly for the case of Medium weights, each quantile level τ has it own weights and we reparametrize them in terms of the softmax applied to unconstrained parameters $\{\phi_j^\tau\}_{j=1}^p$ as in $w_j^\tau = e^{\phi_j^\tau} / \sum_{k=1}^p e^{\phi_k^\tau}$, $j = 1, \ldots, p$. We initialize all parameters $\{\phi_j^\tau\}_{j=1}^p$ by zeros. As the Global aggregators do not require any explicit model (i.e. feedforward network), the hyper-parameter space is only defined by optimization related parameters: learning rate: {1e-3, 5e-4}, weight decay: {1e-5, 1e-7}, crossing penalty weight: {0.1, 0.5, 1.0, 2.0, 5.0}

- **Local Aggregators**: This model requires a network G that outputs the weights as described in Section 2.2. The feature extractor f_θ is a feedforward network and it has hyper-parameters such as the number of layers L and the hidden layer size d. The last layer of G linearly maps the hidden representation $h = f_\theta(x)$, followed by a softmax operation, to local probability weights $w^\tau(x) = (w_1^\tau(x), \ldots, w_p^\tau(x))$, where each weight for τ is $w_j^\tau(x) \in \mathbb{R}^p$ for Local-Medium, or $w_j^\tau(x) \in \mathbb{R}^{pm}$ for Local-Fine. This last layer is parametrized, for each $\tau \in \mathcal{T}$, by a projection matrix $W^\tau \in \mathbb{R}^{p \times d}$ in the case of Local-Medium, (or $W^\tau \in \mathbb{R}^{pm \times d}$ for Local-Fine). The hyper-parameter space is defined as: learning rate: {1e-3, 5e-4}, weight decay: {1e-5, 1e-7}, crossing penalty weight: {0.1, 0.5, 1.0, 2.0, 5.0}, # of layers: {2}, # of hidden units: {64, 128}, Dropout ratio: {0.0, 0.05, 0.1, 0.2}.

Both aggregators are optimized by using Adam optimizer (Kingma & Ba, 2015) and the mini-batch size is set as 64 (256, for larger datasets $n > 10,000$). We again employ the same early-stopping strategy previously discussed for our neural network base models.

E. Additional Results for UCI Datasets

In this section, we provide overall results on the 8 datasets from UCI. Note that we omitted the *wine* dataset from this UCI benchmark, as its Y is discrete ratings. We report the pinball loss and interval score (lower values are better for both metrics), as well as the empirical coverage of the central 90% and 50% prediction intervals (values closer to 90% and 50% are better for these metrics). All metrics are computed on the test data in these tables.

Table 2. Overall results for the UCI datasets (Part I).

	yacht (308)		boston (506)		energy (768)		concrete (1,030)	
	Pinball loss	Interval score	Pinball loss	Interval score	Pinball loss	Interval score	Pinball loss	Interval score
CGN	0.00549	0.14386	0.08087	1.89117	0.01714	0.42368	0.07256	1.64985
SQR	0.00716	0.12396	0.08240	1.59034	0.01522	0.31192	0.08072	1.57220
MQR	0.00616	0.22399	0.07531	1.67686	0.01385	0.40273	0.07238	1.78708
RandomForest	0.01353	0.28773	0.08797	2.59063	0.01538	0.42696	0.08218	2.50894
ExtraTrees	0.01515	0.32914	0.08728	2.81273	0.01538	0.57861	0.09249	3.15178
LightGBM	0.00891	0.65042	0.09679	2.12739	0.00999	0.33479	0.06645	1.99147
LinearRegression	0.13026	1.99935	0.12437	2.82730	N/A	N/A	0.17752	4.10138
Uniform	0.01989	0.46887	0.07578	2.01090	0.01264	0.38962	0.07511	2.19220
Global-Coarse (LP)	0.00550	0.18411	0.07567	1.70976	0.01328	0.36080	0.06989	1.69448
Global-Coarse	0.00545	0.20033	**0.07452**	1.72714	0.00968	0.34259	0.05959	1.70450
Global-Medium (LP)	0.00543	0.19259	0.07555	2.06229	0.01322	0.37130	0.06954	1.84009
Global-Medium	0.00526	0.15821	0.07488	1.95853	0.00972	0.32620	0.05956	1.76101
Global-Fine	0.00580	0.15175	0.08508	**1.51464**	0.00890	0.31731	0.05761	1.52328
Local-Coarse	0.00514	0.17697	0.07582	1.75343	0.00933	0.32781	0.05982	1.69648
Local-Medium	**0.00493**	0.13302	0.07610	1.74402	0.00942	0.28327	0.06007	1.63722
Local-Fine (DQA)	0.00508	**0.12041**	0.07537	1.60724	**0.00825**	**0.21999**	**0.05413**	**1.39384**

Table 3. Overall results for the UCI datasets (Part II).

	kin8nm (8,192)		power (9,568)		naval (11,934)		protein (45,730)	
	Pinball loss	Interval score	Pinball loss	Interval score	Pinball loss	Interval score	Pinball loss	Interval score
CGN	0.07190	1.81872	0.06235	1.66607	0.04128	1.08126	0.18046	4.84186
SQR	0.07329	1.50440	0.06319	1.32014	0.02522	0.61972	0.16828	3.32438
MQR	0.06979	1.85376	0.06119	1.64430	0.01376	0.46928	0.15701	3.61982
RandomForest	0.14889	4.65525	0.04988	1.51362	0.01675	0.84528	0.13222	3.35804
ExtraTrees	0.15731	5.03812	0.05461	1.70923	0.02130	1.19824	0.14601	3.59518
LightGBM	0.13943	3.65442	0.05051	1.16868	0.02071	1.13409	0.16083	3.61974
LinearRegression	0.21694	5.83818	0.07425	1.89121	N/A	N/A	0.23387	4.34796
Uniform	0.10566	3.23390	0.05426	1.49574	0.01744	0.83361	0.15620	3.69877
Global-Coarse (LP)	0.06813	1.71149	0.06034	1.55758	0.01372	0.48941	N/A	N/A
Global-Coarse	0.06813	1.71629	0.04733	**1.20494**	0.01314	0.57181	0.13213	3.35962
Global-Medium (LP)	0.06814	1.76113	0.06008	1.59381	0.01400	0.48300	N/A	N/A
Global-Medium	0.06821	1.76564	0.04716	1.40143	0.01341	0.45548	0.13211	3.33118
Global-Fine	0.06877	**1.70202**	0.04673	1.42614	0.01329	0.33642	0.13050	3.24122
Local-Coarse	**0.06812**	1.71743	0.04659	1.21179	0.01249	0.51100	0.13127	3.32136
Local-Medium	0.06815	1.74869	0.04663	1.37338	0.01314	0.43309	0.12924	3.17276
Local-Fine (DQA)	0.06839	1.74857	**0.04411**	1.29783	**0.01132**	**0.21337**	**0.12533**	**3.00417**