Supplementary Material for High-Quality Prediction Intervals for Deep Learning: A Distribution-Free, Ensembled Approach

Tim Pearce [1 2] **Mohamed Zaki** [1] **Alexandra Brintrup** [1] **Andy Neely** [1]

A. Experimental details

In this section we give full experimental details of the work described in the main paper. Code is made available online[1]. Note that whilst single-layer NNs were used to be consistent with previous works, the developed methods may be applied without modification to deeper architectures.

A.1. Qualitative Experiments

A.1.1. Training method: PSO vs. GD

For the qualitative training method comparison, PSO vs. GD (section 5.1), NNs used ReLU activations and 50 nodes in one hidden layer. GD was trained using $Loss_{QD-soft}$ and run for 2,000 epochs, PSO was trained using $Loss_{QD}$ and run for 50 particles over 2,000 iterations, parameters as given in SPSO 2011 were followed (Thomas et al., 2012). Data consisted of 200 points sampled uniformly from the interval $[-2, 2]$.

A.1.2. Loss function: QD vs. MVE

For the loss function comparison, QD vs. MVE (section 5.2), NNs used Tanh activations and 50 nodes in one hidden layer. Both methods were trained with GD and results are for an individual NN (not ensembled). Data consisted of 200 points sampled uniformly from the interval $[-2, 2]$.

A.1.3. Model Uncertainty Estimation: Ensembles

For evaluation of ensembling (section 5.3), we sampled 50 points uniformly in the interval $[-4, -1]$, and another 50 from $[1, 4]$. An ensemble of ten QD NNs using ReLU activations and 50 nodes in one hidden layer was trained with GD, using parameter resampling.

A.1.4. Training Method / Loss Function Permutation

We provide quantitative results in table 1 for the two synthetic datasets described in sections 5.1 & 5.2. These results cover permutations of loss function and training method [LUBE, QD, MVE]x[GD, PSO], using individual NNs (not ensembled).

Again, 200 training data points were sampled uniformly from the interval $[-2, 2]$. Validation was on 2,000 data points sampled from the same interval. Experiments were repeated ten times. PIs targeted 95% coverage.

LUBE relates to the 'softened' version of eq. (16), and QD to $Loss_{QD-soft}$. Softened versions were used for both GD and PSO (note 'soft' and 'hard' versions were contrasted in section 5.1).

For GD, 2,000 epochs were used, for PSO, 10 particles were run over 2,000 epochs. This meant the computational effort for PSO was five times that required by GD - the computational equivalent of 2,000 forward and backward passes for GD is 2 particles at 2,000 epochs for PSO.

NLL & RMSE metrics were computed, however should be viewed with caution for LUBE and QD - see section A.2.2.

All training methods and loss functions slightly overfitted the training data, producing PICP's lower than 95%. Generally GD-trained NNs outperformed their PSO counterparts in terms of the primary metric of their loss function ($MPIW$ for LUBE and QD, NLL for MVE), although quality of other metrics was similar. QD produced comparable results to LUBE - we note that our contributions to the loss function were from a theoretical and usability perspective and did not expect a large impact on performance. MVE produced PIs of comparable width to LUBE and QD for the case of normal noise, but PIs were significantly wider in the exponential noise case.

[1]Department of Engineering, University of Cambridge, UK [2]Alan Turing Institute, UK. Correspondence to: Tim Pearce <tp424@cam.ac.uk>.

Proceedings of the 35th International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018. Copyright 2018 by the author(s).

[1]https://github.com/TeaPearce

Table 1. Full permutation results of training method and loss function on synthetic data with differing noise distributions. Mean ± one standard error.

	LUBE		MVE		QD	
	GD	PSO	GD	PSO	GD	PSO
Normal noise						
PICP	0.90 ± 0.01	0.91 ± 0.01	0.91 ± 0.01	0.93 ± 0.01	0.91 ± 0.01	0.91 ± 0.01
MPIW	1.16 ± 0.05	1.13 ± 0.04	1.11 ± 0.04	1.00 ± 0.02	0.97 ± 0.04	1.07 ± 0.04
RMSE	0.42 ± 0.01	0.43 ± 0.01	0.39 ± 0.00	0.38 ± 0.01	0.40 ± 0.01	0.41 ± 0.01
NLL	0.60 ± 0.07	0.47 ± 0.06	-0.44 ± 0.02	-0.23 ± 0.04	0.13 ± 0.08	0.37 ± 0.08
Exponential noise						
PICP	0.91 ± 0.01	0.92 ± 0.01	0.93 ± 0.01	0.95 ± 0.00	0.91 ± 0.01	0.93 ± 0.01
MPIW	0.80 ± 0.02	0.93 ± 0.04	1.07 ± 0.04	0.99 ± 0.03	0.84 ± 0.03	0.98 ± 0.04
RMSE	0.39 ± 0.00	0.40 ± 0.01	0.38 ± 0.01	0.37 ± 0.01	0.39 ± 0.01	0.41 ± 0.01
NLL	0.36 ± 0.09	0.64 ± 0.13	-0.48 ± 0.02	-0.18 ± 0.02	0.40 ± 0.05	0.54 ± 0.13

A.2. Benchmarking Experiments

A.2.1. Set up and Hyperparameters

For the benchmarking section, experiments were run across ten open-access datasets, train/test folds were randomly split 90%/10%, with experiments repeated 20 times, input and target variables were normalised to zero mean and unit variance. NNs had 50 neurons in one hidden layer with ReLU activations. The exceptions to this were for experiments with the two largest datasets, *Protein* and *Song Year*, where NNs had 100 neurons in one hidden layer, and were repeated five times and one time respectively.

The softening factor was constant for all datasets, $s = 160.0$. For the majority of the datasets $\lambda = 15.0$, but was set to 4.0 for *naval*, 40.0 for *protein*, 30.0 for *wine*, and 6.0 for *yacht*. The Adam optimiser was used with batch sizes of 100. Five NNs were used in each ensemble, using parameter resampling.

Hyperparameters requiring tuning were learning rate, decay rate, λ, initialising variance, and number of training epochs. Tuning was done on a single 80%/20% train/validation split and using random search.

A.2.2. NLL & RMSE Results

In table 2 we report NLL & RMSE in unnormalised form to be consistent with previous works. Note that in the main results (section 6) we found it more meaningful to leave $MPIW$ in normalised form so that comparisons could be made across datasets.

To compute NLL & RMSE for QD-Ens, we used the midpoint of the PIs as the point estimate to calculate RMSE. We computed the equivalent Gaussian distribution of the PIs by centering around this midpoint and using a standard deviation of $(y_{Ui} - y_{Li})/3.92$ (since the PI represented 95% coverage), which enabled NLL to be computed. We emphasise that by doing this, we break the distribution-free assumption of the PIs, and include these purely for the purpose of consistency with previous work. Unsurprisingly, NLL & RMSE metrics for QD-Ens are poor. MVE-Ens results are in line with previously reported work (Lakshminarayanan et al., 2017).

References

Lakshminarayanan, B., Pritzel, A., and Blundell, C. Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles. In *31st Conference on Neural Information Processing Systems*, 2017.

Thomas, P., Mansot, J., Delbe, K., Sauldubois, A., and Bilas, P. Standard Particle Swarm Optimisation, 2012. URL `https://hal.archives-ouvertes.fr/file/index/docid/926514/filename/Delbe{_}9783.pdf`.

Table 2. RMSE and NLL on ten benchmarking regression datasets; mean ± one standard error, best result in bold.

			RMSE		NLL	
	n	D	MVE-ENS	QD-ENS	MVE-ENS	QD-ENS
BOSTON	506	13	**2.84 ± 0.19**	3.38 ± 0.26	**2.60 ± 0.10**	2.74 ± 0.14
CONCRETE	1,030	8	**5.20 ± 0.10**	5.76 ± 0.10	**2.95 ± 0.04**	3.10 ± 0.02
ENERGY	768	8	**1.67 ± 0.05**	2.30 ± 0.04	**1.12 ± 0.05**	1.62 ± 0.06
KIN8NM	8,192	8	**0.08 ± 0.00**	0.09 ± 0.00	**-1.28 ± 0.01**	-1.14 ± 0.01
NAVAL	11,934	16	**0.00 ± 0.00**	**0.00 ± 0.00**	**-5.67 ± 0.03**	-5.73 ± 0.03
POWER PLANT	9,568	4	**3.94 ± 0.03**	4.10 ± 0.03	**2.77 ± 0.01**	2.83 ± 0.01
PROTEIN	45,730	9	**4.35 ± 0.02**	4.98 ± 0.02	**2.74 ± 0.02**	3.12 ± 0.02
WINE	1,599	11	**0.62 ± 0.01**	0.65 ± 0.01	**1.07 ± 0.06**	1.15 ± 0.03
YACHT	308	6	1.36 ± 0.09	**1.00 ± 0.08**	1.02 ± 0.05	**0.76 ± 0.07**
SONG YEAR	515,345	90	**8.88 ± NA**	9.30 ± NA	**3.37 ± NA**	3.58 ± NA