UNCECOMP 2021
4th ECCOMAS Thematic Conference on
Uncertainty Quantification in Computational Sciences and Engineering
M. Papadrakakis, V. Papadopoulos, G. Stefanou (eds.)
Streamed from Athens, Greece, 27—30 June 2021

UNCERTAINTY QUANTIFICATION FOR DEEP LEARNING REGRESSION MODELS IN THE LOW DATA LIMIT

Cristina Garcia-Cardona[1], **Yen Ting Lin**[1], **and Tanmoy Bhattacharya**[1]

[1]Los Alamos National Laboratory
Los Alamos, NM, USA
e-mail: {cgarciac, yentingl, tanmoy}@lanl.gov

Keywords: Uncertainty Quantification, Heteroscedastic Model, Quantile Model.

Abstract. *Deep learning models have contributed to a broad range of applications, but require large amounts of data to learn the desired input-output mapping. Despite the success in developing prediction engines that have high accuracy, much less attention has been given to assessing the error associated with individual predictions. In this work, we study machine-learning models of uncertainty quantification for regression, i.e., methods that are almost purely data driven and use deep learning itself to quantify the confidence in its predictions. We use two approaches, namely the heteroscedastic and quantile formulations, and their extensions to problems with multidimensional output. We focus on the low data limit, where the data sets available are on the order of hundred, not thousands, samples. Through numerical experiments we demonstrate that both heteroscedastic and quantile formulations are robust and good at uncertainty estimation even in this low data limit. We note that the quantile formulation seems to have better performance and is more stable than the heteroscedastic case. Overall, our studies pave the way towards practical design of deep learning models that provide actionable predictions with quantified uncertainty using accessible volumes of data.*

1 INTRODUCTION

Deep learning models have contributed to a broad range of applications including image processing, speech recognition, drug discovery and computational materials science. These models can often vastly accelerate inference, but, being purely data driven with little input about the subject matter, require large amounts of data to learn the desired input-output mapping. Most of the work in the field has been on developing prediction engines that have high accuracy; much less attention has been given to automatically assessing the error associated with individual predictions, but this is no less an essential task when the results are applied in fields such as medicine or engineering [1].

In this work we study machine-learning based models of uncertainty quantification for regression. In contrast with ensemble methods [2], probabilistic machine learning methods [3] or dropout-based approaches [4], that use the mean and variance generated by the dispersion among realizations of models, the strategies that we apply are based on treating the simultaneous prediction of the target and its confidence as a multi-task problem and training the regression models using loss functions that specifically take into account a measure of predictive uncertainty. We specifically evaluate heteroscedastic [5] and quantile [6] formulations and consider their extension to problems with multidimensional output.

The need for uncertainty quantification is especially true in the low data limit, where the density of input points is too low to have replicate measurements in small neighborhoods in feature space, and yet predictions that do not separate the certain from the uncertain are often not actionable! Therefore, in this work, we also study this small data limit. In particular, we study the case where the data sets available are on the order of hundred, not thousands of, samples. Real world data in the biological world are often similarly scarce, due, for example, to the high-cost of experiments, a large number of control parameters, and the high-dimensionality of the poorly-understood feature space, which makes it critical to maximize the predictive value of the trained models.

The document is structured as follows. Sec. 2 introduces the different uncertainty quantification formulations evaluated for regression tasks. Sec. 3 describes the machine learning architectures used. Sec. 4 details the numerical experiments performed, and Sec. 5 finalizes with conclusions drawn from the work.

2 UNCERTAINTY QUANTIFICATION MODELS

We compare two different strategies for deep-learning the uncertainty quantification task, namely the heteroscedastic [5] and quantile [6] formulations. The former models the predictions as normal random variables with parameters that depend on the input. The learning task, then, involves the simultaneous prediction of the mean and the variance of the target output. The quantile formulation, on the other hand, directly learns the various quantile functions for the prediction as a multi-task setting. We also consider extensions of heteroscedastic and quantile formulations for problems with multidimensional output. This section describes both approaches in more detail.

2.1 Heteroscedastic Model

For simplicity, we first consider a heteroscedastic model which regards the one-dimensional observed value as a sample from a univariate normal distribution, with the mean and the variance given by a smooth function of the input features. In the next subsection, we will generalize the method to higher-dimensional observations, for which multivariate Gaussian distributions shall

be adopted. To learn to predict both mean and variance, the machine learning model is trained to minimize the negative log-likelihood (NLL) of the feature-dependent normal distribution [5]. Hence, the heteroscedastic loss over the entire training dataset which consists of N pairs of input $\mathbf{x}_i$ and output y_i, $i = 1 \dots N$, can be expressed as,

$$\mathcal{L}(f, \sigma; \left\{y^i, \mathbf{x}^i\right\}_{i=1}^N) = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{2\sigma(\mathbf{x}^i)^2} \left\| y^i - f(\mathbf{x}^i) \right\|^2 + \frac{1}{2} \log \sigma(\mathbf{x}^i)^2 \,, \tag{1}$$

with $f(\mathbf{x})$ and $\sigma(\mathbf{x})^2$ the mean and variance predictions of the model, respectively, and N the samples in the training set. This is similar to the work of Lakshminarayanan et al. [7], but differs in the fact that we do not use ensembles, and also, we guarantee the positivity of the variance by predicting $\log \sigma(\mathbf{x}^i)^2$ as in [5], instead of the softplus function $\log(1 + \exp(\cdot))$ that they use.

Multivariate Approach

For prediction of multiple outputs, a product of one-dimensional normal distributions is not able to capture the correlation in the uncertainty prediction of the different outputs. Therefore, we use the probability density function (PDF) of a multivariate normal distribution to construct the heteroscedastic loss for the multivariate case. The PDF of a multivariate normal distribution can be written as

$$f(\mathbf{z}; \boldsymbol{\mu}, \Sigma) = \frac{\det(\Sigma)^{-1/2}}{2\pi^{k/2}} \, \exp\left(-\frac{1}{2} (\mathbf{z} - \boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{z} - \boldsymbol{\mu}) \right) \,, \tag{2}$$

where $\boldsymbol{\mu} \in \mathbb{R}^k$ stands for the mean, $\Sigma \in \mathbb{R}^{k \times k}$ represents the positive definite covariance matrix, and k is the output space dimension. To learn to predict both the mean and covariance, the machine learning model is trained to minimize the NLL of the multivariate normal PDF, again with parameters chosen as smooth functions on the input domain. The multivariate heteroscedastic loss corresponds to

$$\mathcal{L}(\mathbf{f}, \Sigma; \left\{\mathbf{y^i}, \mathbf{x^i}\right\}_{i=1}^N) = \frac{1}{N} \sum_{i=1}^{N} \left((\mathbf{y}^i - \mathbf{f}(\mathbf{x}^i))^T \Sigma(\mathbf{x}^i)^{-1} (\mathbf{y}^i - \mathbf{f}(\mathbf{x}^i)) + \log \det(\Sigma(\mathbf{x}^i)) \right) \,, \tag{3}$$

where $\mathbf{f}(\mathbf{x}) \in \mathbb{R}^k$ stands for the vector of the mean prediction of the model with k outputs, $\Sigma(\mathbf{x}) \in \mathbb{R}^{k \times k}$ represents the predicted covariance matrix, $\mathbf{y}$ the true mean value, and the constants $(2\pi)^{-k/2}$ and $1/2$ have been omitted. To guarantee the that the covariance matrix Σ is positive definite, the learning task is specified such that one learns A such that $\Sigma = A^T A$, which is positive by definition.

In general, we need to be careful to avoid the flat-directions of A, i.e., the changes in A that do not change $A^T A$ making the learning task difficult. In the two-output case, with vector $\mathbf{f}(\mathbf{x}) \in \mathbb{R}^2$ and matrix $A(\mathbf{x}) \in \mathbb{R}^{2 \times 2}$, which will be the focus of our study, this problem is easily solved. For simplicity, the explicit $\mathbf{x}$ dependence of A is (mostly) omitted in the following description. Since the 2×2 matrix A can be represented in terms of scalar components a, b, c, d, as

$$A = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \,,$$

the corresponding covariance matrix can be written as

$$\Sigma = A^T A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}^T \begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} a^2+c^2 & ab+cd \\ ab+cd & b^2+d^2 \end{pmatrix} \triangleq \begin{pmatrix} \sigma_{11}^2 & \sigma_{12} \\ \sigma_{12} & \sigma_{22}^2 \end{pmatrix} . \quad (4)$$

Note that any simultaneous rotation by the same angle of the vectors given by the columns of matrix A leaves the covariance matrix unchanged. We choose rotation angle $\theta = \arctan(c - b)/(a+d)$, to force $b = c$. With this convention, we can learn the three unconstained parameters a, $b \equiv c$, and d instead of the three parameters σ_{ij} that need to be constrained to obtain a positive matrix Σ. The remaining freedom in the choice of the parameters turns out to be discrete sign freedoms of the two rows that do not pose difficulties in learning. With Σ in hand, the likelihood function can be explicitly calculated by computing the inverse of the 2×2 covariance matrix analytically,

$$\Sigma^{-1} = \frac{1}{\det(\Sigma)} \begin{pmatrix} \sigma_{22}^2 & -\sigma_{12} \\ -\sigma_{12} & \sigma_{11}^2 \end{pmatrix} , \quad (5)$$

with $\det(\Sigma) = \sigma_{11}^2\sigma_{22}^2 - \sigma_{12}^2 \neq 0$. Defining: $\mathbf{e} = (e_1, e_2)^T = \mathbf{y} - \mathbf{f}(\mathbf{x})$, allows to write

$$\begin{aligned} \text{NLL} &= \frac{1}{\det(\Sigma)} \begin{pmatrix} e_1 \\ e_2 \end{pmatrix}^T \begin{pmatrix} \sigma_{22}^2 & -\sigma_{12} \\ -\sigma_{12} & \sigma_{11}^2 \end{pmatrix} \begin{pmatrix} e_1 \\ e_2 \end{pmatrix} + \log\det(\Sigma) \\ &= \frac{1}{\det(\Sigma)} (\sigma_{22}^2\, e_1^2 - 2\,\sigma_{12}\, e_1\, e_2 + \sigma_{11}^2\, e_2^2) + \log\det(\Sigma) . \end{aligned} \quad (6)$$

Thus, the loss function for a heteroscedastic approach with two outputs can be written in a simplified form as

$$\begin{aligned} \mathcal{L}(\mathbf{f}, \Sigma; \{\mathbf{y^i}, \mathbf{x^i}\}_{i=1}^N) = \frac{1}{N} \sum_{i=1}^{N} \Bigg(& \frac{1}{\det(\Sigma(\mathbf{x}^i))} (\sigma_{22}(\mathbf{x}^i)^2 (y_1^i - f_1(\mathbf{x}_i))^2 \\ & - 2\,\sigma_{12}(\mathbf{x}^i)(y_1^i - f_1(\mathbf{x}^i))(y_2^i - f_2(\mathbf{x}^i)) \\ & + \sigma_{11}(\mathbf{x}^i)^2\, (y_2^i - f_2(\mathbf{x}^i))^2) + \log\det(\Sigma(\mathbf{x}^i)) \Bigg) . \end{aligned} \quad (7)$$

2.2 Quantile Model

The quantile model does not make any assumption about the underlying distribution of the samples. Thus, instead of predicting mean and variance of a normal distribution, the predictions are directly of the various quantiles. A quantile is a set of values that divides a frequency distribution of a variable into equal groups, each containing the same fraction of the whole variable range. To learn a quantile map with a machine learning model, the model is trained to minimize the quantile loss for any given quantile $\alpha \in (0, 1)$. The quantile loss for an individual sample $\mathbf{x}_i$ is defined as [8]

$$\mathcal{L}_\alpha(\xi^i) = \begin{cases} \alpha\, \xi^i & \text{if } \xi^i \geq 0 , \\ (\alpha - 1)\, \xi^i & \text{if } \xi^i < 0 . \end{cases} , \quad (8)$$

where $\xi^i = y^i - f^\alpha(\mathbf{x}^i)$, and, y^i and $f^\alpha(\mathbf{x}^i)$, correspond to the observed value and the predicted quantile α, respectively.

In order to learn to predict several quantiles simultaneously, a loss function composed by the sum over the individual quantile losses of the entire data set can be formulated as follows

$$\mathcal{L}(\{f\}^{\alpha}; \{y^i, \mathbf{x}^i\}_{i=1}^N) = \sum_{\alpha} \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}_{\alpha} \left(y^i - f^{\alpha}(\mathbf{x}^i) \right) . \tag{9}$$

Specifically, we devise our model to predict three quantiles: the 1st, 5th, and 9th deciles, corresponding to α = 0.1, 0.5 and 0.9, respectively. Note that the quantile for $\alpha = 0.5$ is the median of the distribution.

Multivariate Approach

Extending the quantile formulation to prediction with multiple outputs is not as direct as in the heteroscedastic case, since there is no underlying assumption of the form of the sample distribution, nor a basis for imposing an ordering in multivariate observations as is in the scalar one-output case. Among possible multivariate quantile extensions, we use the generalization given by the geometric quantile formulation proposed in [9]. In the geometric quantile formulation, the d-dimensional multivariate quantiles are indexed by elements of the open unit ball $B^{(d)} = \{\mathbf{u} | \mathbf{u} \in \mathbb{R}^d, |\mathbf{u}| < 1\}$. A function $\Phi(\mathbf{u}, \mathbf{t}) = |\mathbf{t}| + \langle \mathbf{u}, \mathbf{t} \rangle$ is defined for any element $\mathbf{u} \in B^{(d)}$ and $t \in \mathbb{R}^d$, with $\langle \cdot, \cdot \rangle$ denoting the usual Euclidean inner product. The geometric quantile $\hat{\mathbf{Q}}_n(\mathbf{u})$ corresponding to ('index') $\mathbf{u}$ and based on d-dimensional data points $\mathbf{X}^1, \mathbf{X}^2, \ldots, \mathbf{X}^n$ is defined as

$$\hat{\mathbf{Q}}_n(\mathbf{u}) = \arg \min_{\mathbf{Q} \in \mathbb{R}^d} \sum_{i=1}^{n} \Phi(\mathbf{u}, \mathbf{X}^i - \mathbf{Q}) . \tag{10}$$

Note that for $\mathbf{u} = 0$, the quantile $\hat{\mathbf{Q}}_n(\mathbf{0})$ corresponds to the spatial median. Also, note that a $\mathbf{u}$ with $|\mathbf{u}|$ close to 1 corresponds to an extreme quantile, while close to 0 corresponds to a central quantile. In general, the magnitude of $|\mathbf{u}|$ measures the extent of deviation of $\hat{\mathbf{Q}}_n(\mathbf{u})$ with respect to the center of the cloud formed by the $\{\mathbf{X}^i\}_{i=1}^n$ points, while the direction of $\mathbf{u}$ can be interpreted as providing a notion of how 'out' a point is in a given direction with respect to the center of the cloud, considering the geometry of the cloud itself. Further details can be found in [9].

In our case, we restrict ourselves to directions of $\mathbf{u} = \mathbf{1}$, i.e., the vector with unit components, while adopting the proper normalization to make it an element of $B^{(d)}$. Overloading the alpha notation, to keep a knob $\alpha \in (0, 1)$, we define the following geometric multivariate quantile loss

$$\mathcal{L}_{\alpha}(\boldsymbol{\xi}^i) = \Phi \left(\frac{\mathbf{1}}{\sqrt{d}} (2\alpha - 1) , \boldsymbol{\xi}^i \right) , \tag{11}$$

where $\boldsymbol{\xi}^i = \mathbf{y}^i - \mathbf{f}^{\alpha}(\mathbf{x}^i)$, $\mathbf{f}^{\alpha}(\mathbf{x}^i)$ is the predicted multivariate quantile model, $\mathbf{y}^i$ is the observed value and d is the output dimension. Note that $\alpha = 0.5$ corresponds to the spatial median, while for $\alpha < 0.5$ this quantile formulation really uses the $\mathbf{u} = -\mathbf{1}$ direction. Analogously to the 1D case, we learn simultaneously several geometric multivariate quantiles (in the $\mathbf{u} = \pm\mathbf{1}$ direction), by minimizing

$$\mathcal{L}(\{f\}^{\alpha}; \{\mathbf{y}^i, \mathbf{x}^i\}_{i=1}^N) = \sum_{\alpha} \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}_{\alpha} \left(\mathbf{y}^i - \mathbf{f}^{\alpha}(\mathbf{x}^i) \right) . \tag{12}$$

With this setting, we again use α = 0.1, 0.5 and 0.9.

3 MACHINE LEARNING MODELS

To instantiate the uncertainty quantification formulations discussed and estimate $\mathbf{f}$, $\mathbf{f}^{\alpha}$ and Σ, we construct different machine learning architectures and train them for regression tasks. Specifically, we build multi-layer feed-forward neural networks with different number of layers and train them using the loss functions described for heteroscedastic and quantile formulations.

The multi-layer feed-forward neural network that we train is composed of neurons with dense connections. The output o_{ν}^{λ} of each artificial neuron ν in layer λ is computed as

$$o_{\nu}^{\lambda} = h\left(\mathbf{w}_{\nu}^{\lambda} \cdot \boldsymbol{\xi}^{\lambda} + b_{\nu}^{\lambda}\right) , \tag{13}$$

where $\boldsymbol{\xi}^{\lambda}$ represents the input vector at layer λ; $\mathbf{w}_{\nu}^{\lambda}$ and b_{ν}^{λ} represent neuron parameters: weight vector and bias, respectively; the operator $\cdot$ denotes a dot product; and h, the activation function. We use a rectified linear unit (ReLU): $\mathrm{ReLU}(\beta) = \max(0, \beta)$ as the activation function, except in the output layer where we use a linear activation.

We keep the same architecture in all cases to asses the impact of the loss function. We adapt the number of model outputs according to the uncertainty quantification to evaluate:

- 1D Heteroscedastic: prediction of mean response $f(\mathbf{x})$ and variance $\sigma(\mathbf{x})^2$.
- 2D Heteroscedastic: prediction of mean response $\mathbf{f}(\mathbf{x})$ and covariance matrix $\Sigma(\mathbf{x})$.
- 1D Quantile: prediction of $\alpha = 0.1, 0.5, 0.9$ quantiles, $f^{0.1}(\mathbf{x})$, $f^{0.5}(\mathbf{x})$, $f^{0.9}(\mathbf{x})$, respectively.
- 2D Quantile: prediction of $\mathbf{u} = \pm\mathbf{1}$ $\alpha = 0.1, 0.5, 0.9$ multivariate quantiles, $\mathbf{f}^{0.1}(\mathbf{x})$, $\mathbf{f}^{0.5}(\mathbf{x})$, $\mathbf{f}^{0.9}(\mathbf{x})$, respectively.

Specifically, a heteroscedastic model for one-variable prediction has two outputs: $f(\mathbf{x}^i)$ and $\sigma(\mathbf{x}^i)$, while a heteroscedastic model for two-variable prediction has five outputs: $f_1(\mathbf{x}^i)$, $f_2(\mathbf{x}^i)$, $\sigma_{11}(\mathbf{x}^i)$, $\sigma_{12}(\mathbf{x}^i)$ and $\sigma_{22}(\mathbf{x}^i)$. Analogously, a quantile model for one-variable prediction has three outputs, since we chose to predict three quantiles: $f^{0.5}(\mathbf{x}^i)$, $f^{0.1}(\mathbf{x}^i)$ and $f^{0.9}(\mathbf{x}^i)$. A quantile model for two-variable prediction has six outputs, since we chose to predict three quantiles: $f_1^{0.5}(\mathbf{x}^i)$, $f_2^{0.5}(\mathbf{x}^i)$, $f_1^{0.1}(\mathbf{x}^i)$, $f_2^{0.1}(\mathbf{x}^i)$, $f_1^{0.9}(\mathbf{x}^i)$ and $f_2^{0.9}(\mathbf{x}^i)$.

Regularization To reduce the possibility of model overfitting we apply two well known techniques. We use dropout [10] after each dense layer to randomly drop a fraction of the layer's neurons and avoid co-adaptation. We also make use of the Tikhonov regularization by minimizing the ℓ_2 norm of the weight vectors $\mathbf{w}_{\nu}^{\lambda}$ associated to the different neurons in the model.

4 NUMERICAL EXPERIMENTS

We compare the heteroscedastic and quantile formulations in terms of predictive uncertainty for synthetic data, where we know the ground truth, using regression prediction tasks with one-output (1D case) and two-outputs (2D case). We assess the effect of model complexity by training machine learning models with different numbers of layers. To measure the performance we use the following metrics: mean squared error (MSE) eq. (14), the coefficient of determination (R^2) eq. (15) and the negative log-likelihood (NLL) eq. (16).

$$\mathrm{MSE}(y, f) = \frac{1}{N} \sum_{i=1}^{N} \left\| y^i - f(\mathbf{x}^i) \right\|^2 , \tag{14}$$

$$R^2(y, f) = 1 - \frac{\sum_{i=1}^{N} \left(y^i - f(\mathbf{x}^i)\right)^2}{\sum_{i=1}^{n} \left(y^i - \bar{y}\right)^2}, \quad \bar{y} = \frac{1}{N} \sum_{i=1}^{N} y^i, \tag{15}$$

$$\text{NLL}(y, f, \sigma) = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{2\sigma(\mathbf{x}^i)^2} \left\| y^i - f(\mathbf{x}^i) \right\|^2 + \frac{1}{2} \log \sigma(\mathbf{x}^i)^2 . \tag{16}$$

Note that these expressions refer to the 1D case but analogous expressions apply to the 2D case for MSE and NLL. For computing R^2 in the 2D case, the outputs are concatenated in a 1D vector. Note also that for the quantile model the prediction $f(\mathbf{x}^i)$ used is the median, i.e., the 5th decile: $f^{0.5}(\mathbf{x}^i)$. Additionally, note that the NLL expression is valid only for the heteroscedastic model. For measuring NLL of a quantile model, in 1D, we use a normal approximation where we suppose that the interdecile range (IDR), equivalent to the difference between 9th and 1st deciles (i.e., 80% of the samples), corresponds to 80% of the samples of a normal distribution. In a normal distribution this percentage is contained in a band of 1.28σ radius from the mean. Hence we use $\sigma(\mathbf{x}^i)_q = \text{IDR}(\mathbf{x}^i)/2.56$. We do not estimate the likelihood for the 2D quantile model.

Moreover, in order to have a metric that is sensitive to predictive uncertainty but that is independent of the hypothesized sample distribution, we use the 80% coverage. This is estimated in the 1D heteroscedastic case by computing the fraction of samples falling inside the band of 1.28σ radius from the mean. In the 2D heteroscedastic case, by computing the fraction of samples falling inside the ellipsoid with isolevel equal to $-2\log(0.2)$. In the 1D quantile case, by computing the fraction of samples falling inside the interval between 1st and 9th deciles. And in the 2D quantile case, by computing the fraction of samples falling inside the square defined by using multivariate quantiles $\alpha = 0.1$ and $\alpha = 0.9$ as corners.

4.1 Synthetic Data Generation

Synthetic data has been constructed using the Hill model,

$$H(x; r_1, r_2, k, h) = \begin{cases} r_1 + (r_2 - r_1)\frac{k^h}{k^h + x^h} & \text{for } h \geq 0\,, \\ \\ r_1 + (r_2 - r_1)\frac{k^{-h}}{k^{-h} + x^{-h}} & \text{for } h < 0\,, \end{cases} \tag{17}$$

with parameters h and $r_1, r_2, k > 0$.

For the 1D case, i.e., the one-output case, the Hill model with parameters $r_1 = 10$, $r_2 = 30$, $k = 10$ and $h = -6$ is used as the base function, $y = H(x)$. A 1D Gaussian noise is added to it. The Gaussian noise has mean zero and standard deviation given by another Hill model with parameters $r_1 = 5.48$, $r_2 = 3.16$, $k = 10$ and $h = -6$. In this way we simulate a heteroscedastic, i.e., feature-dependent, noise model. This data is plotted in Figure 1 (Left). It can be seen that it corresponds to a monotonically decreasing 1D signal with relatively high dispersion and that the dispersion moderately increases for higher x-coordinate values. Note that the MSE of the noisy data with respect to the clean data is MSE=19.03.

For the 2D case, i.e., the two-outputs case, a Hill model with parameters $r_1 = 10$, $r_2 = 30$, $k = 10$ and $h = 5$ is used as base function for $y_1 = H(x)$ and $y_2 = H(x + 1)$. A 2D

Gaussian noise is added to the outputs. The Gaussian noise has mean zero and covariance matrix $\Sigma = A^T A$ with

$$A(x) = \begin{pmatrix} 5 \times 10^{-3}\, x^2 & 2 \times 10^{-3}\, x \\ 2 \times 10^{-3}\, x & 8 \times 10^{-3}\, x^2 \end{pmatrix}.$$

This data is plotted in Figure 1 (Right). It can be seen that it corresponds to a monotonically increasing 2D signal with relatively low dispersion and that the dispersion increases for higher x-coordinate values. Note that the MSE of the noisy data with respect to the clean data is MSE=0.19.

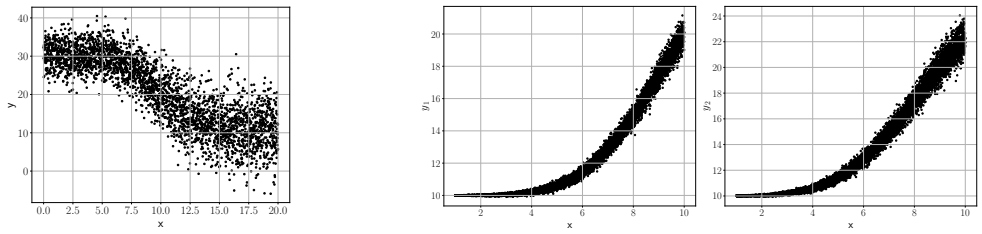

Figure 1: Synthetic data generated for model training. **Left: 1D, N=3000. Right: 2D, N=10000**.

4.2 Results

For the numerical experiments, we build sets of 3000 samples for the 1D case and 10000 samples for the 2D case. For the 1D case, we construct uncertainty estimator models with 50 neurons per layer, and, 2, 3, 5 and 10 layers. For the 2D case, we construct uncertainty estimator models with 100 neurons per layer and, 1, 2, 3, 5 and 10 layers. We use low and high levels of regularization. These correspond to: dropout=0.01 and Reg $\ell_2 = 1 \times 10^{-8}$ (i.e., the factor used for weighting the regularization term given by the ℓ_2 norm of the network weight parameters), for low regularization and dropout=0.2 and Reg $\ell_2 = 1 \times 10^{-5}$ for high regularization. We split the data in 80% training and 20% testing. We train these models with the Adam optimizer [11] during 500 epochs for 1D models and 1000 epochs for 2D models, since those are the approximate number of epochs for stabilized loss function. All the models are implemented in Keras [12].

Box plots for the 1D results obtained for training with N=2400 samples evaluated over the test set (600 samples) for 20 repetitions are shown in Figure 2. Note that both type of models exhibit good performance not only with respect to the predicted value (MSE comparable to training data, median $R^2 > 0.75$) but also in terms of the uncertainty estimations, with similar NLL and coverage close to the expected 80%. Overall, metrics are slightly better for the quantile model, even when the data is a heteroscedatic noisy data. Interestingly, note that there is no significant change in performance for the different number of layers evaluated (2, 3, 5 and 10 layers) or for the low and high regularization levels used.

Box plots for the 2D results obtained for training with N=8000 samples evaluated over the test set (2000 samples) for 20 repetitions are shown in Figure 3. Note that both type of models exhibit good predictive accuracy (MSE comparable to training data, median $R^2 > 0.9$) and the uncertainty estimations are close to the 80% coverage specified, with slight under-prediction for the heteroscedastic case and the quantile with low regularization and small over-prediction

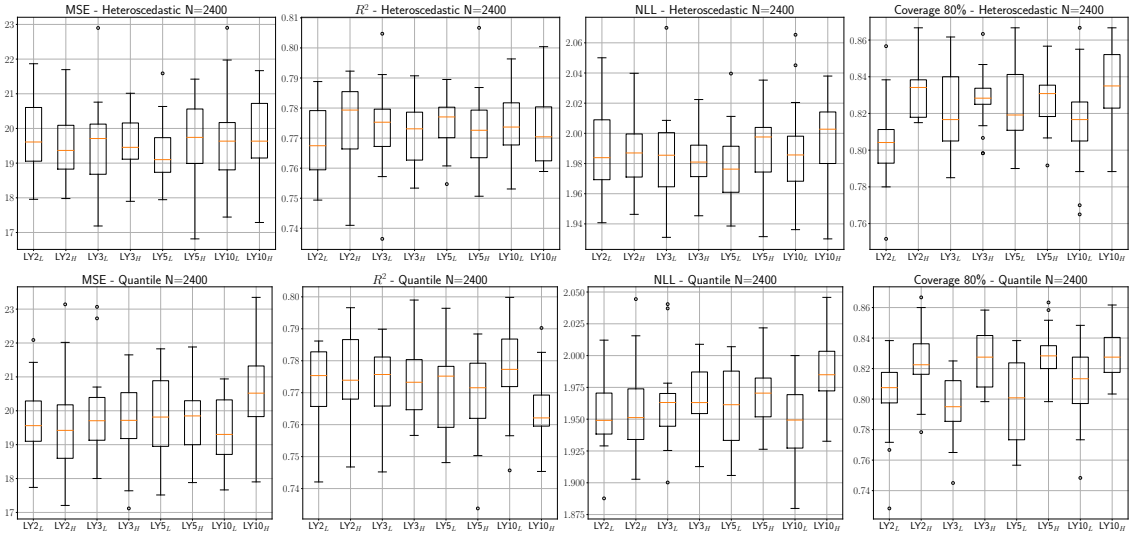

Figure 2: Statistics for models with low and high regularizations, for 1D data and N=2400 samples. LY denotes the number of layers, while the L or H sub-index is used to denote regularization level. **Top: Heteroscedastic. Bottom: Quantile**.

for the quantile with high regularization. Additionally, performance is similar for models between 1 and 3 layers, with performance degrading a little for models with 5 layers and high regularization or 10 layers.

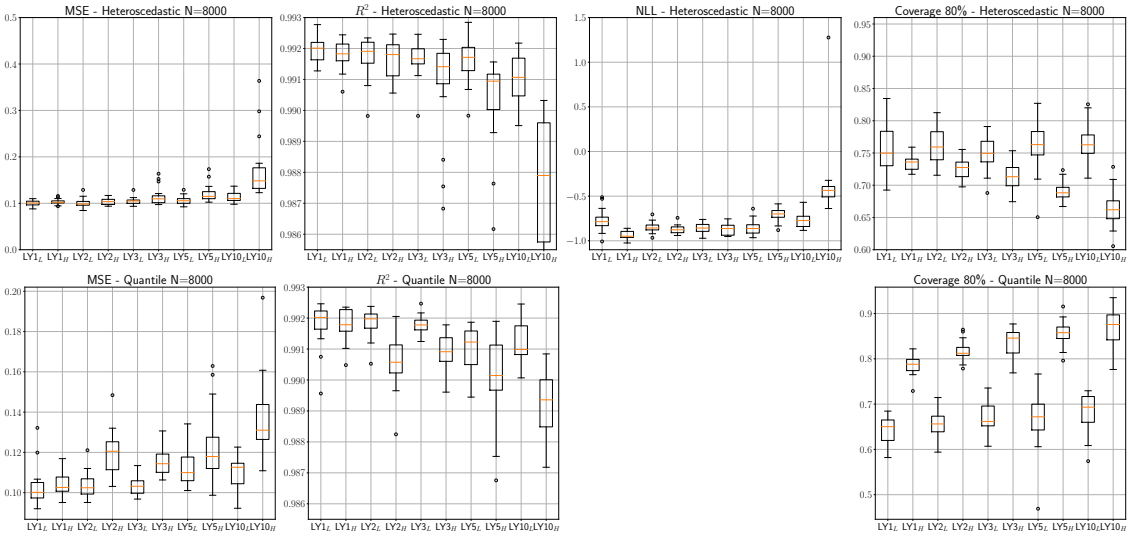

Figure 3: Statistics for models with low and high regularizations, for 2D data and N=8000 samples. LY denotes the number of layers, while the L or H sub-index is used to denote regularization level. **Top: Heteroscedastic. Bottom: Quantile**.

4.2.1 Low Data Limit in 1D

For evaluating the consistency between models trained with enough data and models trained in the low data limit, we repeat the training procedure but using $N = 30$ samples for training.

Note that in this case we let models with 10 layers to train during 1000 epochs.

Box plots for the 1D results obtained for training with $N = 30$ samples evaluated over the test set (2970 samples) for 20 repetitions are shown in Figure 4. MSE and R^2 for the heteroscedastic model have been framed in ranges comparable to quantile results so some boxes for the 2 layers neural network are cut. Note that despite the substantial reduction of the data for training, the model performance only narrowly degrades with respect to the case where enough samples are available. Also, some differences between the performance of the two main uncertainty quantification paradigms start to emerge, as well as some more notorious deviations with respect to the complexity of the architecture of the network or the regularization level.

The performance of the quantile formulation degrades less than the heteroscedastic one, with lower MSEs and NLLs, higher R^2 and coverage closer to the 80% expected value. Also, the quantile performance is consistent for 2 to 5 layers deteriorating for the 10-layers network with low regularization (labeled LY10_L in the plots). In general, the regularization seems to play a minor role, with little differences in the 80% coverage prediction between low and high regularization. In contrast, the heteroscedastic model with 2 layers has poor performance. Between 3 to 5 layers, low regularization seems moderately better than high regularization, but the opposite seems to apply for 10 layers. These results seem to indicate that the quantile formulation is more robust for the low data limit, requiring only marginal regularization, unless the network has a high complexity, in which case stronger regularization helps. The heteroscedastic formulation needs an architecture that has medium complexity to be effective in the low data limit, with slightly better performance for low regularization.

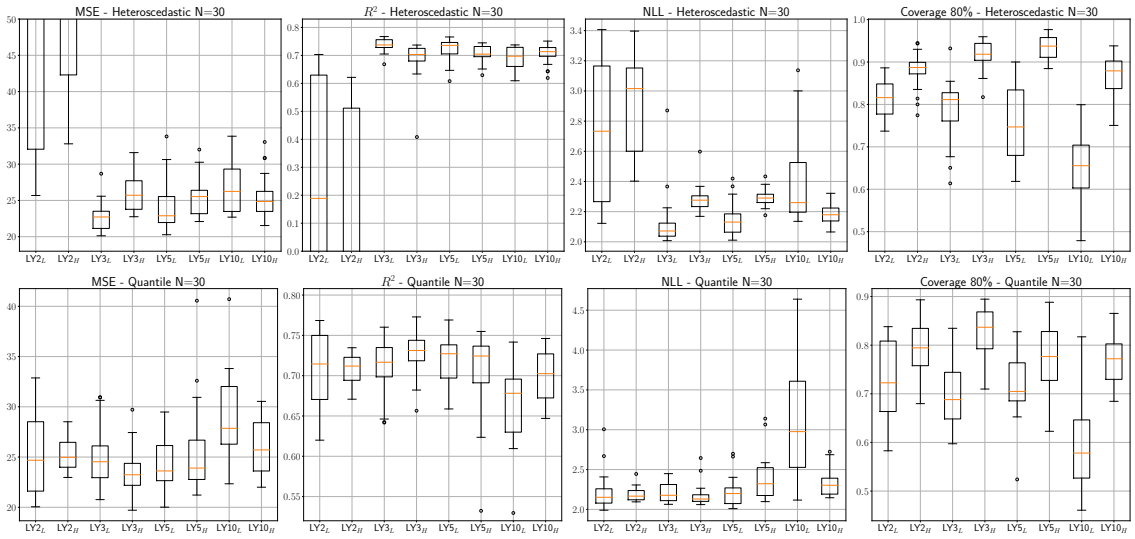

Figure 4: Statistics for models with low and high regularizations, for 1D data and N=30 samples. LY denotes the number of layers, while the L or H sub-index is used to denote regularization level. **Top: Heteroscedastic. Bottom: Quantile**.

Additionally, we include results in the low data limit for models trained to minimize the MSE loss, i.e., without considering any uncertainty quantification formalism. Box plots for the 1D results obtained for training with $N = 30$ samples evaluated over the test set (2970 samples) for 20 repetitions are shown in Figure 5. The homoscedastic label denotes that no specific noise model is used for training. Note again that the degradation observed for heteroscedastic and quantile formulations, is in the order of the one for the homoscedastic case, in the small data

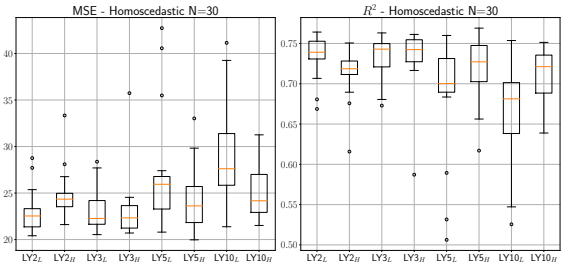

Figure 5: Statistics for models with low and high regularizations, for 1D data and N=30 samples, trained without uncertainty quantification. LY denotes the number of layers, while the L or H sub-index is used to denote regularization level.

limit. Therefore, adding uncertainty quantification does not sacrifice model performance, even in the low data limit. On the contrary, it provides a complementary level of information and a small reduction of the model dispersion, specially for cases where the MSE loss has worse performance.

Nevertheless, in the low data limit it makes more sense to also assume that the amount of data for testing is low. Hence, a more practical comparison seems to be to also evaluate model performance in a small testing set. We evaluate this criterion in the models with 3 layers. In this comparison, we randomly select 40 samples and split them randomly into two subset: $N = 30$ samples for training and the remainder 10 samples for testing. We sweep over a logarithmic 10×10 grid of the regularization parameters. For dropout in the range: $[1 \times 10^{-3}, 2 \times 10^{-1}]$ and for Reg ℓ_2 in the range $[1 \times 10^{-8}, 1 \times 10^{-4}]$. For each of the different regularization combinations in the grid, we train a model for 500 epochs and evaluate the performance with respect to all the metrics in the testing set of 10 samples. We compute 20 repetitions over the grid sweeping and report the test median in Figure 6. Not all the metrics are completely consistent with each other, but it seems that the performance for the heteroscedastic model is slightly better on the lower left corner (i.e., very low dropout and ℓ_2 regularization), while there does not seem to be a clear trend in the quantile model.

4.2.2 Low Data Limit in 2D

We proceed as in the 1D case and train models for the 2D data in the low data limit using $N = 60$ and low and high regularizations. Note than in this case the heteroscedastic model requires more iterations to achieve reasonable performance (i.e., about 10000 epochs). Likewise, we train the highly regularized quantile model for 3000 epochs due to slight oscillations. Box plots for the 2D results obtained for training with $N = 60$ samples evaluated over the test set (9940 samples) for 20 repetitions are shown in Figure 7. Overall the performance of the low data limit models is consistent with the models trained with enough samples. The main differences can be found in the 80% coverage results, with both formulations exhibiting better performance for models with low regularization. As a reference, box plot results for training with MSE loss are included in Figure 8. Again, uncertainty quantification formulations, do not degrade performance while slightly reducing model dispersion and providing a measure of the confidence in the machine learning model predictions.

Analogously to the 1D case, we also evaluate the model performance in a small testing set using models with 3 layers. Therefore, we randomly select 80 samples and split them randomly into two subset: $N = 60$ samples for training and the remainder 20 samples for testing. We

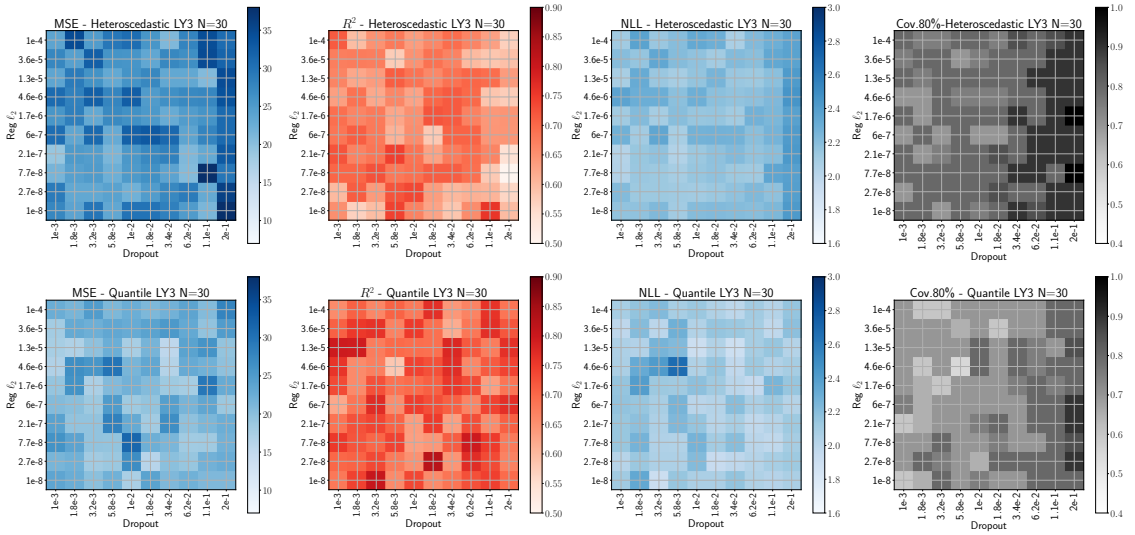

Figure 6: Statistics for models with 3 layers over a grid of different dropouts and ℓ_2 norm regularizations for 1D data and N=30 samples. When blue colormap is used, lower values correspond to better performance; when red colormap is used, higher values correspond to better performance; when gray colormap is used, values closer to 0.8 correspond to better performance. **Top: Heteroscedastic. Bottom: Quantile**. Same scale used in both cases.

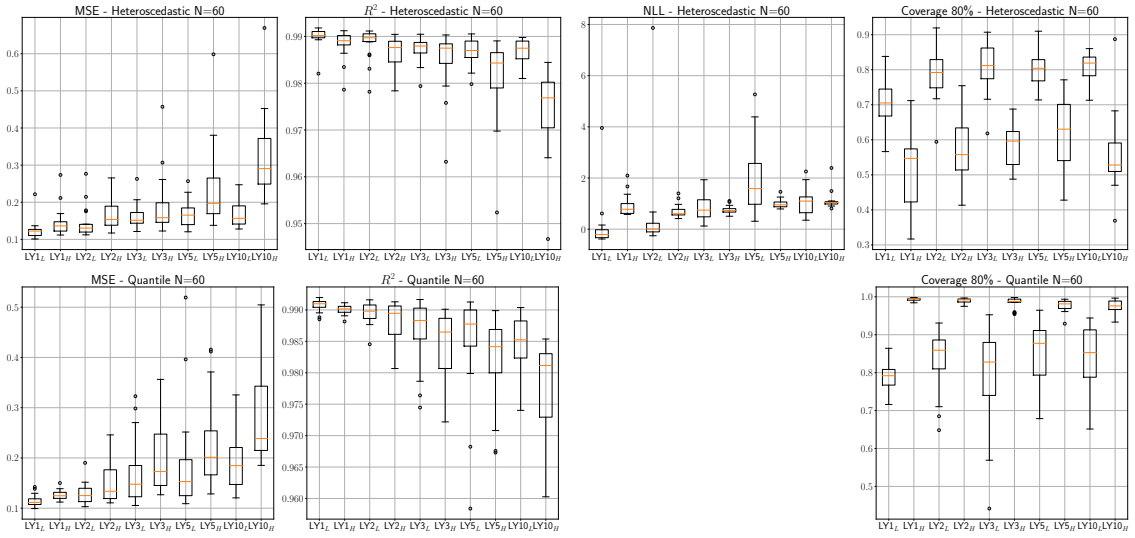

Figure 7: Statistics for models with low and high regularizations, for 2D data and N=60 samples. LY denotes the number of layers, while the L or H sub-index is used to denote regularization level. **Top: Heteroscedastic. Bottom: Quantile**.

sweep over the same 10×10 logarithmic grid of the regularization parameters than in the 1D case. For each of the different regularization combinations in the grid, we train a model for 5000 epochs for the heteroscedastic case and 1000 epochs for the quantile case, and evaluate the performance with respect to all the metrics in the testing set of 20 samples. We compute 20 repetitions over the grid sweeping and report the test median in Figure 9. It seems clear that in both formulations better results are achieved for small dropout regularization.

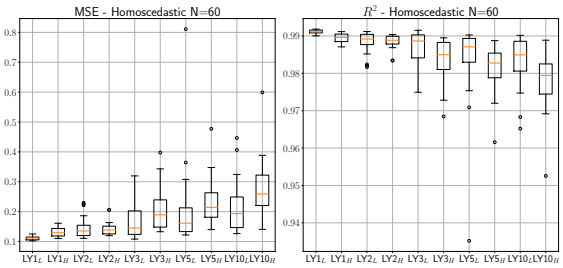

Figure 8: Statistics for models with low and high regularizations, for 2D data and N=60 samples, trained without uncertainty quantification. LY denotes the number of layers, while the L or H sub-index is used to denote regularization level.

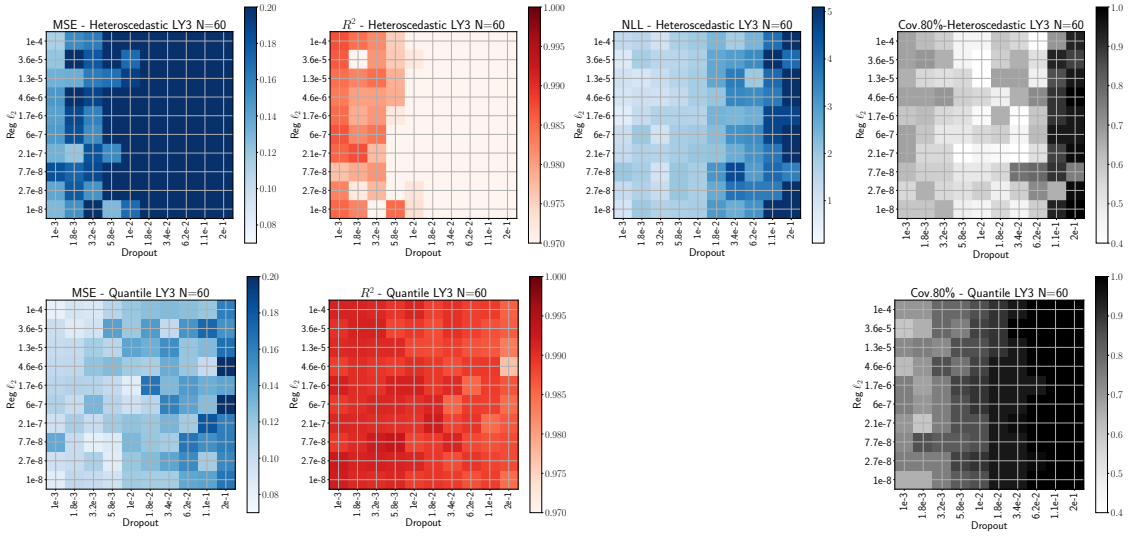

Figure 9: Statistics for models with 3 layers over a grid of different dropouts and ℓ_2 norm regularizations for 2D data and N=60 samples. When blue colormap is used, lower values correspond to better performance; when red colormap is used, higher values correspond to better performance; when gray colormap is used, values closer to 0.8 correspond to better performance. **Top: Heteroscedastic. Bottom: Quantile**. Same scale used in both cases, losing some resolution.

5 CONCLUSIONS

In this work, we applied uncertainty quantification models, namely heteroscedastic and quantile formulations, to synthetic data with one and two-outputs. We trained neural-network models with different complexities and evaluated their performance in the low data limit, where just a handful of data (tens of samples) is available. Through numerical experiments we demonstrate that both heteroscedastic and quantile formulations are robust and good at uncertainty estimation even in the low data limit. Furthermore, we find that in these formulations, very small 'true regularization' strategies, such as dropout or weighting of the ℓ_2-norm of the parameters, are required to produce good results even for complex models. Also, we note that the quantile formulation seems to have better performance and is more stable than the heteroscedastic case, i.e., the quantile models do not degrade as fast when model complexity is increased. Overall, our studies pave the way towards practical design of deep learning models that provide actionable predictions with quantified uncertainty using accessible volumes of data.

ACKNOWLEDGEMENTS

This work has been supported in part by the Joint Design of Advanced Computing Solutions for Cancer (JDACS4C) program established by the U.S. Department of Energy (DOE) and the National Cancer Institute (NCI) of the National Institutes of Health, and was performed under the auspices of the U.S. Department of Energy by Los Alamos National Laboratory under Contract DE-AC5206NA25396. YTL acknowledges partial support from LDRD (Laboratory Directed Research and Development) program under project 20210043DR (Uncertainty Quantification for Robust Machine Learning). Approved for public release LA-UR-21-22482.

REFERENCES

[1] E. Begoli, T. Bhattacharya, and D. Kusnezov, "The need for uncertainty quantification in machine-assisted medical decision making," *Nature Machine Intelligence*, vol. 1, pp. 20–23, Jan. 2019.

[2] S. Jain, G. Liu, J. Mueller, and D. Gifford, "Maximizing overall diversity for improved uncertainty estimates in deep ensembles," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, pp. 4264–4271, Apr. 2020. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/5849

[3] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight uncertainty in neural networks," in *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ser. ICML'15. JMLR.org, 2015, p. 1613–1622.

[4] Y. Gal and Z. Ghahramani, "Dropout as a Bayesian approximation: Representing model uncertainty in deep learning," in *Proceedings of Machine Learning Research*, M. F. Balcan and K. Q. Weinberger, Eds., vol. 48. New York, New York, USA: PMLR, 20–22 Jun 2016, pp. 1050–1059. [Online]. Available: http://proceedings.mlr.press/v48/gal16.html

[5] A. Kendall and Y. Gal, "What uncertainties do we need in Bayesian deep learning for computer vision?" in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017, pp. 5574–5584. [Online]. Available: https://proceedings.neurips.cc/paper/2017/file/2650d6089a6d640c5e85b2b88265dc2b-Paper.pdf

[6] S. Abeywardana, "Deep quantile regression," 2018, https://towardsdatascience.com/deep-quantile-regression-c85481548b5a.

[7] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017, pp. 6402–6413. [Online]. Available: https://proceedings.neurips.cc/paper/2017/file/9ef2ed4b7fd2c810847ffa5fa85bce38-Paper.pdf

[8] T. S. Ferguson, *Mathematical Statistics: A Decision Theoretic Approach*. New York and London: Academic Press, 1967.

[9] P. Chaudhuri, "On a geometric notion of quantiles for multivariate data," *Journal of the American Statistical Association*, vol. 91, no. 434, pp. 862–872, Jun. 1996.

[10] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.

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

[12] F. Chollet, *Keras documentation*, 2018. [Online]. Available: https://faroit.com/keras-docs/2.1.2