IRAQI STATISTICIANS JOURNAL

https://isj.edu.iq/index.php/isj

ISSN: 3007-1658 (Online)

Beta Distributional Neural Networks for Bounded Outcome Prediction

Ahmed Naziyah Alkhateeb

Department of Operations Research and Intelligent Techniques, University of Mosul, Iraq

ARTICLE INFO

Article history:

Received	4 March 2026
Revised	4 March 2026
Accepted	31 March 2026
Available online	3 April 2026

Keywords:

Beta Distribution
Neural Networks
Bounded outcome
Synthetic data
Colon Cancer

ABSTRACT

This paper proposes a new neural modelling system, named the Beta-Distributional Neural Network (BDNN), a distributional learning framework, which is aimed at predicting restricted outcomes over the open unit interval (0,1). In contrast to the traditional neural networks, which produce deterministic point estimates, the proposed model provides direct predictions of the two shape parameters of the beta distribution (α and β), and thus, predictive mean and uncertainty can be estimated simultaneously. This is to achieve statistical coherence and computational stability, which is trained through maximum likelihood estimation by minimising the negative log-likelihood. It is tested on BDNN using synthetic data sets, which have been built based on the data-generating curves, both linear and nonlinear, as well as a real-world colon cancer drug-response dataset. It makes comparative studies using ordinary least squares regression, classical beta regression and traditional feedforward neural network. Empirical evidence reveals that BDNN has better prediction ability in several cases and offers extra probabilistic modelling functionality by direct estimation of the Beta distribution parameters in the majority of experimental contexts in the form of smaller RMSE, MAE values and greater probabilistic calibration using negative log-likelihood. These results indicate the adequacy of incorporating distributional modelling with deep neural structures in the prediction of limited outcomes in medical and applied research areas.

1. Introduction

Due to the development of automated machine learning, deep learning has become a very precise method of application in real-world scenarios, especially in scenarios where substantial amounts of data can be provided. The tremendous increase in the data volume over the past several decades has widened the range of applicability and efficacy of the deep learning techniques significantly. Deep learning models in regression problems generally have a single point estimate or a set of point estimates. Nevertheless, these methods frequently do not take advantage of the power of high-dimensional output spaces that can simply be modelled as full probability distributions. Outcome prediction is an essential activity in various fields, and some of them are the fields of science, economics, and health. Outcome variables that are constrained to a finite interval, especially in the interval (0,1) are also required in many applications. The cases are: probability of a tumour responding to chemotherapy, probability of a tumour recurring after surgery has been performed, or probability of a patient surviving. In this case, conventional regression models can give invalid results in the form of negative probabilities or ones higher than one, which are not interpretable. To overcome these drawbacks, the beta distribution is proposed, the distributional parameters

Corresponding author E-mail address: ahmed.alkhateeb@uomosul.edu.iq
https://doi.org/10.62933/

of which are directly modelled with the help of a neural network. The network is trained by maximizing the likelihood function, which is also minimizing the negative log-likelihood (NLL). The neural network is a prediction of the parameters of the beta distribution, which can be used to obtain the predictive mean and variance by analytics, making predictive probabilities directly and understandably.

The ability to predict is a basic duty in most disciplines, including economics and medical care. A typical need in such applications is that variables must be predicted as falling within a given range (0, 1), as in the case of credit assessment of the likelihood of negative events such as loan defaults. One can often come across nonsensical or impossible predictions in such applications (e.g. probability of default that is less than zero, probability of default that is greater than one). Although such predictions are not possible due to certain limitations in the architecture of machine learning models, more accurate models can be created [1].

An approach involving applying MC-DropConnect to weights rather than units to approximate Bayesian inference in deep neural networks over a Bernoulli distribution to enable more accurate estimation of epistemic uncertainty at reduced computational cost is proposed by [15]. Describing a comparison of Bayesian neural networks (BNNs) between mutual information analysis and explicit modelling of liver segmentation on deep neural networks, the addition of a neuron to reduce the loss [4] provides a thorough comparison between the tools. The article [3] proposes the PV algorithm, which integrates the Branch and Bound algorithms with linear bound propagation to deliver orders-of-magnitude speedup on probabilistic verification of neural network safety and fairness. It stands as a successful compromise between rigorous mathematical tradition and scalability to practice, which is more efficient than existing state of the art tools, and also has theoretical guarantees of soundness and completeness.

The neural network weights are set through the intelligent initialization algorithms such as [18]. The article by Jais and Ismail investigated how the Adam optimization algorithm can be used to tackle complex classification problems when combined with a wide and deep neural network architecture. Applying the experimental results to the breast cancer dataset in the UCI repository demonstrated that this model combination is more effective at achieving an optimal trade-off between memorization and generalization [18].

The paper builds on the use of the beta distribution in neural regression models by proposing Beta Distribution Neural Networks (BDNNs), which are used to make statistically coherent estimates of limited outcomes. The beta distribution is modelled directly as a part of maximum likelihood estimation (MLE) (where the neural network predicts the two shape parameters (α and β) of the beta distribution). The model is optimised to minimise the Negative Log-Likelihood (NLL). The optimization can be performed directly and made stable without a Monte Carlo approximation. That is done to make sure that the network learns distributional parameters without and with analytical tractability.

2. Beta Distribution

This is why the Beta distribution is incredibly convenient in modelling the data on the continuous scale, where the open interval (0,1) is assumed since its shape can be quite different depending upon the values taken by the two parameters. It is thus the most appropriate for selecting modelling ratios, probabilities and standardized values. In contrast to the logit-normal distribution that requires a Gaussian transformation and has no simple closed expressions for its moments, the beta distribution is characterized by having simple analytic expressions for both the mean and the variance. This feature makes it interpretable and enables the application of the MLE technique with neural networks. Besides, the beta probability is entirely analytic and also computationally efficient, which makes it especially useful in distributive neural modelling of finite outcomes. The probability density function of the Beta distribution (with parameters α and β, where $0<\alpha<1$ and $\beta>0$) is referred to as $B(\alpha,\beta)$ and the P.D.F. assumes the following [8]:

$$f(x;\alpha,\beta) = \frac{1}{B(\alpha,\beta)} x^{\alpha-1}(1-x)^{\beta-1} \qquad (1)$$

Where:

$$B(\alpha,\beta) = \frac{\Gamma\alpha\Gamma\beta}{\Gamma(\alpha+\beta)} \qquad (2)$$

$B(\alpha,\beta)$ is termed the normalising constant, and it is such that the area under the probability density curve is 1.

The peculiarity of the beta distribution is its possibility to assume various shapes based on the values of the parameters α and β. This is what renders it a versatile modelling application [11,23]. In case α = β or higher, the distribution is symmetrical and bell-shaped, similar to the normal distribution. A value below 1 will shift the distribution to the right or to the left, which comes in handy when it is necessary to model the data that follows a normal distribution but is skewed towards the left (to model data with a high frequency of failure) or towards the right (to model data with a high frequency of success). The distribution is also U-shaped or bimodal when a and b are smaller than 1. Finally, the beta distribution becomes uniform when both parameters are equal to 1 [13].

The mean of the beta distribution is:

$$\mu = \frac{\alpha}{\alpha+\beta} \qquad (3)$$

And the variance of this distribution is:

$$\sigma^2 = \frac{\alpha\beta}{[(\alpha+\beta)^2(\alpha+\beta+1)]} \qquad (4)$$

By examining the formulas of the two equations above (3) and (4), we find that these two parameters are responsible for the location, dispersion, and shape of the distribution.

3. Bounded Outcome Prediction

Bounded outcomes are relevant in a wide variety of fields such as the social sciences, economics, medicine, and education [22]. These variables are values between a certain value (which is usually the unit interval [0,1]). Examples of this include the rate of examination pass, hotel occupancy, air quality, Likert-scale survey score, etc. It is crucial to create a proper statistical model of such data; the model should be able to not only approximate the central tendency but also measure the error of such approximations. Linear regression is the most adopted analytic model in cases of limited population outcomes, and hopefully, it is described as being independent of the explanatory variables and normally distributed error terms [20]. In a situation of limited response, these assumptions are often broken in that the normal distribution might produce predictions that fall beyond the admissible range [0,1] and this makes them useless. Further, much empirical data of restricted scope tends to have skewness or a U-shaped form as opposed to the equal-variation bell shape of the normal law, and often exhibits heteroscedasticity [13].

4. Methodological Contribution

The primary methodological contribution of this research is that the distributional modelling and neural network architectures have been combined to predict the outcome with limited resources. Traditional neural networks traditionally aim at making a single point prediction of the dependent variable. These approaches can be good with respect to the accuracy of prediction, but lack information on the uncertainty and variability of the predictions. Conversely, the proposed Beta Distributional Neural Network (BDNN) explicitly represents the parameters of the Beta distribution, so that the network can learn the expected value of the dependent variable but also the probabilistic form of the variable. Precisely, the neural network provides two parameters $\alpha(x)$ and $\beta(x)$ that determine the conditional Beta distribution of the dependent variable. The formulation allows the unbiased representation of the mean and dispersion of confined outcomes in a single probabilistic theory. The proposed approach has three primary advantages in comparison to regular neural networks [25]:

- Probabilistic forecasting and not just point forecasting.
- Conformity to the limited nature of the dependent variable using the Beta distribution.
- Distribution parameter joint learning through likelihood-based optimization.

Thus, the BDNN framework enhances the representational adaptability of the neural networks with the probabilistic modelling of dis-

tributional regression and offers a flexible model representation of the limited answer variables.

5. Beta Distributional Neural Network

Research in machine learning is currently moving towards probabilistic predictions rather than deterministic ones. Probabilistic neural networks are characterized by the fact that they do not predict a single value for y, but rather the parameters of the probability distribution that y follows [11]. Instead of having a single output node in the network, there will be two output nodes; each one will predict one of the parameters α and β of the beta distribution.

The decomposition of the loss function aims to tune the network weights to ensure that the Negative Log-Likelihood (NLL) value is as low as possible, and a bigger likelihood to obtain the training data with the help of the distribution introduced by the network is achieved.

6. Methodological Contribution

The probabilistic neural networks and the distributional outputs have already been studied in the literature; however, the Beta Distributional Neural Network (BDNN) is new in the following aspects:

The two shape parameters of the Beta distribution (α, β) are directly predicted through a neural network architecture specially created to predict bound outcomes in the interval (0,1). The suggested framework combines distributional statistical modelling with deep neural networks and enables predictive uncertainty and mean prediction to be estimated at the same time. The model is learned by maximum likelihood estimation on Beta likelihood to give a statistically coherent probabilistic learning model. The paper provides an empirical comparison of the study in depth with classical beta regression, OLS regression, and deterministic neural networks on both artificial and actual biomedical data.

7. Model Description

The intended model will offer a flexible and precise paradigm of modelling continuous data with values in the range of [0,1] through the merger of the expressiveness of deep neural networks and the statistical flexibility of the beta distribution.

7.1 Model Architecture

The model is based on the idea of Distributional Neural Networks. Instead of a network making a single scalar prediction, the network is configured to produce the parameters of a probability distribution. Let $y_i \in (0,1)$ represent a bounded dependent variable for observation i followed Beta distribution restricted by the open interval (0,1) as follows:

$$y_i \sim Beta(\alpha_i, \beta_i)$$

Let $(x_1, x_2, \ldots, x_p)'$ denote the explanatory variables (where p = the number of explanatory variables), where the neural architecture starts with an input layer, which achieves the encodings of these explanatory variables.

The proposed approach models the whole conditional probability distribution as opposed to traditional regression models which only estimate the conditional mean of the dependent variable. In particular, the neural network comes to a relationship between the independent variables and the coefficients of the beta distribution. The model is now defined as:

$$f_\theta(x_i) = (\alpha_i, \beta_i)$$

Where f_θ th is the neural network that was parameterized by θ (θ is the parameter set of network parameters (weights and biases)).
This formulation helps the model to have both the location and dispersion properties of the dependent variable.

Later layers are concealed and they are a series of fully connected or dense layers. These latent layers extractively encode more and more hierarchical and intricate, non-linear expressions of the relations that exist between the input variables. In between these layers, a non-linear activation is used, like ReLU (Rectified Linear Unit) or Tanh, in this way allowing the network to learn complex relationships [12, 13]. After a few of the fully connected layers, dropout layers are added, and dropout is a regularization technique that ensures that, in the training pro-

cess, the units are randomly dropped, making the network build more robust representations and reducing the chances of overfitting [14]. Lastly, the output layer contains only two nodes, which have the responsibility of generating the parameters that determine the beta and alpha distributions of all the input samples [15]. Let the hidden representations of the neural network be defined as:

$$h^{(1)} = ReLU(W_1 x_i + b_1)$$
$$h^{(2)} = ReLU\left(W_1 h^{(1)} + b_2\right)$$

Where:

Where $h^{(1)}$ is the output of the first hidden layer, a nonlinear transformation of the input features, and $h^{(2)}$is the feature representation that the second hidden layer has learnt. W_1, W_2 are weight matrices, b_1 and b_2 are bias vectors.

The dropout layers between the hidden layers are added during the training in order to increase the generalisation power of the model and minimise overfitting. The last stage of the network delivers two results:

$$z_{1i} = W_3 h^{(2)} + b_3$$
$$z_{2i} = W_4 h^{(2)} + b_4$$

The pre-activation output of z_{1i} and z_{2i} are used to calculate the parameters α_i and β_i. W3 and W4 are weight matrices between the second hidden layer and the output node. b_3 and b_4 are bias terms of the output layer, which are associated with the two output neurons that generate the raw estimates of the Beta distribution parameters. In order to make the parameters strictly positive, a softplus transformation is used:

$$\alpha_i = \log(1 + e^{z_{1i}})$$
$$\beta_i = \log(1 + e^{z_{2i}})$$

This transformation guarantees that the estimated parameters satisfy the positivity constraints required by the Beta distribution.

7.2 Likelihood Function

Maximum Likelihood Estimation (MLE) is used to train the model. The goal is to set the weights of the neural network such that the estimated parameters $\hat{\alpha}_i$, $\hat{\beta}_i$ the beta distribution of the sample, maximizes the probability of the actual value y_i . The probability density function (p.d.f.) of the beta distribution is defined by:

$$p\left(y_i \mid \alpha_i, \beta_i\right) = \frac{\Gamma\left(\alpha_i + \beta_i\right)}{\Gamma\left(\alpha_i\right)\Gamma\left(\beta_i\right)} y_i^{\alpha_i - 1}\left(1 - y_i\right)^{\beta_i - 1}$$

Then the beta distribution is given by the following likelihood [16]:

$$L(\alpha, \beta | y) = \prod_{i=1}^{n} f(y_i; \alpha_i, \beta_i) = \prod_{i=1}^{n} \frac{y_i^{\alpha_i - 1}(1 - y_i)^{\beta_i - 1}}{B(\alpha_i, \beta_i)} \quad (5)$$

In practice, we maximize Negative Log-Likelihood (NLL). The reasons why it is preferred are twofold: it is numerically better, and it lends itself perfectly to the minimization scheme of traditional optimization algorithms (such as Gradient Descent) [17]. Thus, our model loss is the following:

$$NLL(\theta) = -\sum_{i=1}^{n} log\, f\left(y_i; \hat{\alpha}_i, \hat{\beta}_i\right)$$
$$= -\sum_{i=1}^{n} \left\{\log\Gamma\left(\alpha_i + \beta_i\right) - \log\Gamma\left(\alpha_i\right) - \log\Gamma\left(\beta_i\right) + \left(\alpha_i - 1\right)\log y_i + \left(\beta_i - 1\right)\log\left(1 - y_i\right)\right\}$$
$$NLL(\theta) = -\sum_{i=1}^{n}\left[\left(\hat{\alpha}_i - 1\right) log\, y_i + \left(\hat{\beta}_i - 1\right) log(1 - y_i) - log\, B\left(\hat{\alpha}_i, \hat{\beta}_i\right)\right] \quad (6)$$

8. The Output of BDNN

The purpose of this section is to assess the output of the proposed model of a BDNN against competing models, on synthetic data. This methodology included the following steps:

8.1 Synthetic data

Is necessary to be able to measure the accuracy of the model in ideal conditions when the true distribution is known. Four primary artificial datasets are created to test the workability of the model in various circumstances (influenced by the approaches of [5,19,22]:

- **Large Linear Data:** where n=10000 with two explanatory variables x_1 and x_2 follow a uniform distribution. $x_j \sim U(-1,1)$, $j = 1,2$. The Beta parameters have linear definitions:

$$\alpha_i = 1 + 2x_{1i}$$
$$\beta_i = 1 + 3x_{2i} \quad (7)$$

- **Small Linear Data (SL):** Makes use of the same linear relationships previously mentioned, whereas there are $n = 5000$ samples to test the performance on small data.
- **Large Non-Linear Data (LNL):** Generate three explanatory variables $x_j \sim U(-1,1), j = 1,2,3$ and $n = 10000$ samples, and the parameters of Beta distribution are defined using non-linear relationships (quadratic and exponential):

$$\alpha_i = \exp\left(x_{1i}^2 + 0.2x_{2i}\right)$$
$$\beta_i = 9 + \left(x_{3i} + 0.5\right)^2 \quad (8)$$

- **Small Non-Linear Data (SNL):** Makes use of the same linear relationships previously mentioned, whereas there is $n = 5000$.

The steps for generating these data are as follows:

Step (1): Generate the explanatory Variables as:

$x_{ij} \sim U(-1,1)$, $i = 1,2,...,n$, $j = 1,2,...,p$

Step (2): Compute the parameters of Beta distribution:

$\alpha_i = f_\alpha(x_i)$

$\beta_i = f_\beta(x_i)$

Where f_α and f_β are defined in Eq. (7) and (8), respectively.

Step (3): Generate the dependent variables as:

$$y_i \sim Beta(\alpha_i, \beta_i)$$

which ensures that the dependent variable belongs to the interval (0,1).

Step (4): Repeat experiment: the experiment is carried out repeatedly several times to get stable estimates of the performance of the evaluation measures.

8.2 Comparison methods

To investigate the performance of the proposed model, three popular models were chosen: classical beta regression, ordinary OLS regression, and a traditional feedforward neural network (FFNN) trained via backpropagation and further trained using the Adam algorithm [23]. Adam is very efficient in working with batches of data and it automatically scales the learning rate, which is applied to each parameter separately. This is a more stable and convergent training method [20] [21].

8.3 Evaluation criteria

To achieve a full analysis, the following criteria were employed: Root Mean Squared Error (RMSE) because the mean forecast was perfectly predicted, Negative Log-Likelihood (NLL) the simple measure of the quality of the distributional forecasts overall, Mean Absolute Error (MAE) where the accuracy of the point forecast was compared and Probability Integral Transform (PIT) by plotting the histogram of the PIT values [24].

9. Traditional Neural Network (Baseline Model Specification)

To ensure a fair and reproducible comparison with the proposed BDNN, the architecture and hyperparameters of the traditional neural network baseline were fixed as follows:

9.1 Network Architecture

The model is a fully-connected feedforward neural network (Multi-Layer Perceptron - MLP) with the following layered structure:

- Input Layer: Accepts the variable vector (x) with dimensionality (p) (the number of explanatory variables).
- Hidden Layer 1: A dense layer with 128 neurons. It employs the Rectified Linear Unit (ReLU) activation function.
- Dropout Layer 1: A dropout layer with a rate of 0.3, applied after the first hidden layer.
- Hidden Layer 2: A second dense layer with 64 neurons, also using the ReLU activation function.
- Dropout Layer 2: A dropout layer with a rate of 0.2, applied after the second hidden layer.
- Output Layer: A single neuron with a sigmoid activation function

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

to constrain the final prediction to the interval (0, 1), matching the bounded nature of the target variable (y).

9.2 Loss Function & Optimization

- Loss Function: Mean Squared Error (MSE) is used as the objective function for training, corresponding to a deterministic point-estimation task.
- Optimiser: The Adam optimiser is used with the following hyperparameters:

  Learning Rate (α): $\alpha = 1 \times 10^{-3}$
  Decay Rates (β_1, β_2): $\beta_1 = 0.9$ and $\beta_2 = 0.999$, respectively.
  Epsilon (ε): $\varepsilon = 1 \times 10^{-7}$
- Batch Size: Training is performed using mini-batch gradient descent with a batch size of 64.
- Number of Epochs: The model is trained for a maximum of 200 epochs. An early stopping callback is implemented with a patience of 15 epochs, monitoring the validation loss to prevent overfitting. Training is restored to the weights from the epoch with the best validation loss.

9.3 Initialisation & Regularization

- Weight Initialisation: All network weights are initialised using the Glorot (Xavier) uniform initialiser.
- Bias Initialisation: Bias terms are initialised to zero.
- Regularisation: Apart from dropout, L2 weight regularisation with a penalty factor of $\alpha = 1 \times 10^{-4}$ is applied to the kernel weights of all dense layers.

9.4 Implementation Framework

The model was implemented using the Keras API with a TensorFlow backend. The exact random seeds for the NumPy, TensorFlow, and Python random number generators were fixed at the beginning of each experimental run to ensure full reproducibility.

10. Simulation results

In this part, a Monte Carlo simulation study is conducted across four scenarios, each composed of two functional relationships (linear and nonlinear) and two sample sizes (n = 5000 and n = 10000). The design will enable us to assess the performance of the proposed model across varying degrees of data complexity and sample sizes through simulation in R version 2026.01.1+403, to evaluate the proposed bidirectional neural network model's performance relative to competing models. The results will be sorted according to the previously mentioned criteria (RSE, MAE, NLL) and presented in tables and flowcharts.

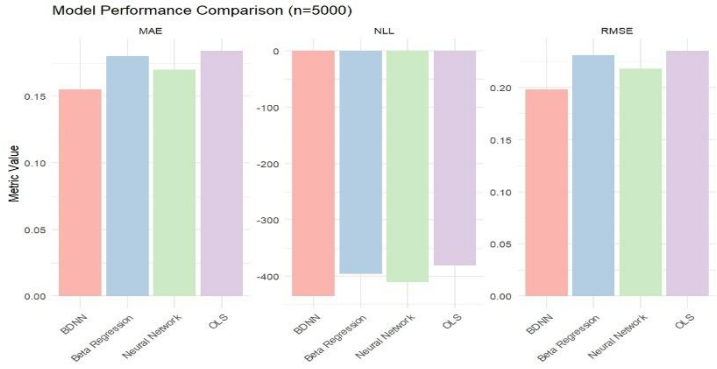

Figure 1. Performance comparison of the linear data model under a sample size of n=5000

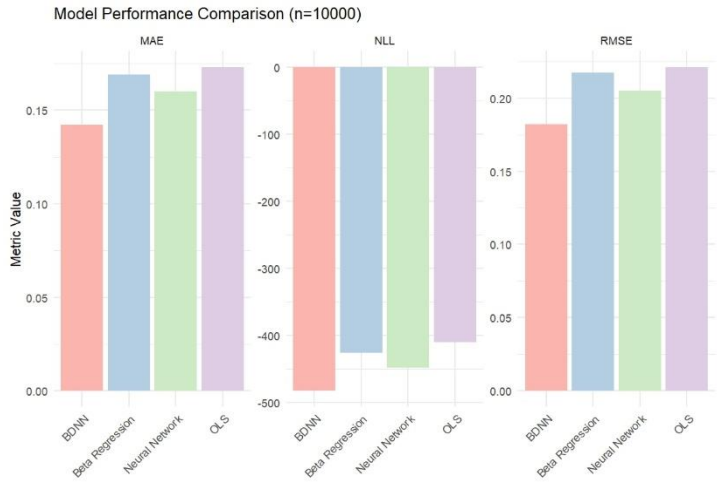

Figure 2. Performance comparison of the linear data model under a sample size of n=10000

Table 1: Comparison table for Linear synthetic data (n = 5000)

Model	**RMSE**	**MAE**	**NLL**
	n=5000		
OLS	0.235	0.184	-380.24
Beta Regression	0.231	0.180	-395.60
Neural Network	0.218	0.170	-410.56
BDNN	**0.198**	**0.155**	**-435.11**

Table 2: Comparison table for Linear synthetic data (n = 10000)

Model	**RMSE**	**MAE**	**NLL**
	n=10000		
OLS	0.221	0.173	-410.25
Beta Regression	0.217	0.169	-425.56
Neural Network	0.205	0.160	-448.21
BDNN	**0.182**	**0.142**	**-482.31**

Tables (1) and (2) shows the answers to the linear synthetic dataset. The output of the BDNN model has the lowest values of RMSE and MAE compared to the evaluated approaches, which points to the better accuracy of prediction of the points. The enhancement is moderate as compared to the typical neural network and beta regression models. Probably, BDNN also minimizes the NLL value, indicating that the variability of the response variable is more well represented by the predicted Beta distribution. Such findings suggest that distributional modelling can enhance prediction in neural networks as well as uncertainty representation.

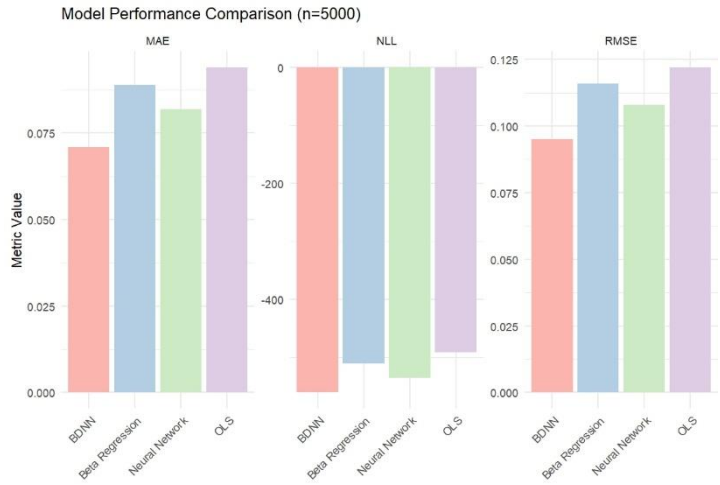

Figure 3. Comparison Figure of non-linear data model when n=5000

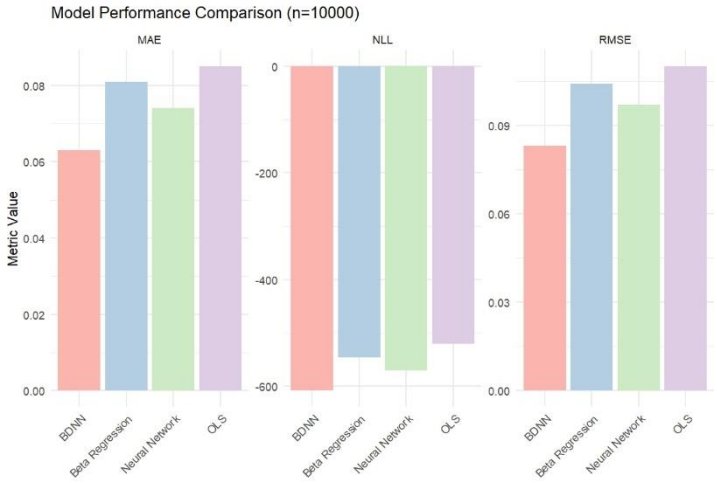

Figure 4. Comparison Figure of non-linear data model when n=10000

Table 3: Comparison table for Linear synthetic data (n = 5000)

Model	**RMSE**	**MAE**	**NLL**
	n=5000		
OLS	0.122	0.094	-490.81
Beta Regression	0.116	0.089	-510.07
Neural Network	0.108	0.082	-535.42
BDNN	**0.095**	**0.071**	**-560.31**

Table 4: Comparison table for Linear synthetic data (n = 10000)

Model	**RMSE**	**MAE**	**NLL**
	n=10000		
OLS	0.110	0.085	-520.09
Beta Regression	0.104	0.081	-545.47
Neural Network	0.097	0.074	-570.37
BDNN	**0.083**	**0.063**	**-608.24**

The results of nonlinear synthetic dataset are reported in tables (3) and (4). As anticipated, the linear models like OLS and Beta regression fail in performance when there exist nonlinear relationships. These are because the neural network model is better at predicting because of their capacity to pick nonlinear trends in the data. Nonetheless, the BDNN model has the highest overall performance as the values of RMSE and MAE are lower than the competing methods. Moreover, the BDNN has the lowest value of NLL, which shows that the predicted Beta distribution is a more accurate probabilistic model of the response variable. This finding shows the benefit of integrating neural networks with distributional modelling in situations where the response variables are finite.

11. Real Data

It used a multimodal dataset with multiple targets, namely drug discovery in colon cancer, comprising 7,643 patients. The data consisted of genomic, transcriptomic, and proteomic data, and it was proposed to create an integrated predictive model of treatment response and survival. The main characteristics were the level of gene expression of 50 significant genes related to the formation of colon cancer, the binary nature of 10 oncogene mutations, quantitative scores of protein interactions, and the level of the regulation of molecular pathways (high, moderate, low). There were also the measurements of drug response, methods of resistance, classification of metabolic pathway, level of toxicity, the data on survival, and the level of drug dosage.

The dataset was also analysed before training the models to make sure the data was of good quality and consistent. The missing values were verified and managed accordingly. To enhance the stability of the training process of the neural network, all the predictor variables were scaled to a standard range.

To test the performance of the models, the dataset was split randomly into two parts, which are training set (80 percent of the observations) to estimate the model parameters and a testing set (20 percent of the observations) to measure the performance of the predictor.

In the case of the neural network models (NN and BDNN), the parameters were estimated with the help of gradient-based optimization. Training of the BDNN was done by minimizing the Negative Log-Likelihood based on the Beta distribution and the standard neural network was trained on the loss of mean squared error.

Each of the models was also tested on the test data and compared in terms of the RMSE, MAE, and NLL in order to provide a fair comparison of the rival approaches.

No additional procedure was applied to select variables because the number of predictive variables in the dataset was relatively small.

The RMSE, MAE and NLL were used to evaluate the models. Key dependent variables in the colon cancer group are Drug response.

Table 5: Goodness-of-fit test for real data

Real Data	χ^2 test	p- value	α	β
Colon Cancer Dataset	4.1278	0.5234	0.9717	0.9697

Table (5) demonstrates the findings of the analysis of the fit of the beta distribution to the actual data in the colon cancer dataset. The value of chi-squared statistic of 4.1287 implies that there are differences between the theoretical and actual distributions. Nevertheless, the large p-value of -0.5234, which is significantly larger than the standard significance value of 0.05, indicates that there is not substantial statistical evidence that can be used to disprove the null hypothesis. That is, the beta distribution with two shape parameters near one ($\alpha = 0.9717, \beta = 0.9697$) fits the distribution of real data quite well and there is no major difference between the two, which confirms the application of this distribution to statistical modeling in this case.

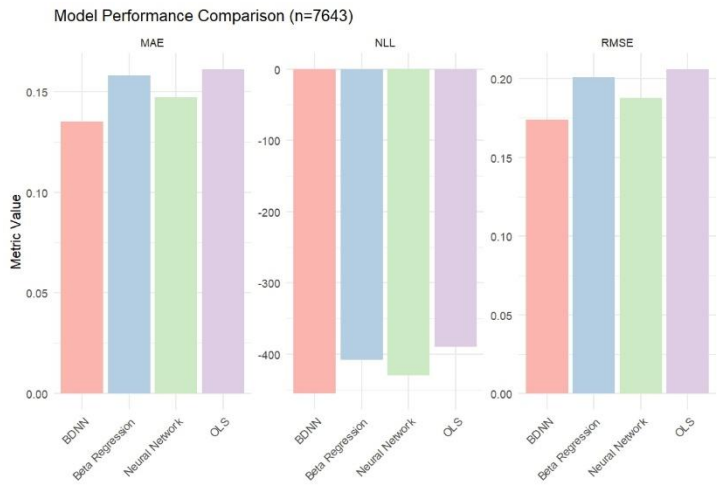

Figure 5. Comparison Figure of colon cancer

Table 6: Indicators of model accuracy in estimating patient response to medications

Model	RMSE	MAE	NLL
OLS	0.206	0.161	-390.25
Beta Regression	0.201	0.158	-408.59
Neural Network	0.188	0.147	-430.17
BDNN	**0.174**	**0.135**	**-455.77**

The findings of the actual dataset in table (6) depict a same trend with the simulation study. The BDNN model has smaller values of RMSE and MAE than the rival models implying that it has better predictive power.

As well, the smaller value of NLL suggests that BDNN model offers a more effective probabilistic overview of the response variable via Beta distribution. Although the differences are moderate, their results show that the proposed model offers competitive predictive performance along with probabilistic modeling ability.

Table 7: Performance Analysis Based on RMSE

Model	RMSE	Rank	Improvement rate compared to BDNN
Neural Network	0.188	1	7.44%
Beta Regression	0.201	2	13.43%
OLS	0.206	3	15.53%

Analysis of table (7) clearly indicates that the BDNN model is superior to all of the compared models since the Neural Network takes first place among the competing models with an RMSE value of 0.188, but it is 7.44% less efficient than BDNN, and the regression models (Beta and OLS) occupy the next places with higher RMSE values of 0.201 and 0.206 and weaker improvement rates than BDNN of 13.43% and 15.53% respectively, indicating the obvious superior.

Comprehensively, the findings imply that the suggested BDNN framework offers a versatile method of modeling finite dependent variables, which is a combination of the representativeness of neural networks and the probabilistic form of a beta distribution.

12. Conclusion

To sum up, the paper has presented the BDNN to represent a new framework that is both pure and simple synthesis of the representational power of deep learning and the statistical power of the beta distribution. The BDNN model predicts the parameters of a beta distribution directly, thus inherently respecting the constraint on outcome variables, which is inherent to the BDNN model, thus removing the inherent limitation of its implausible predictions, which afflict traditional regression models. The vast amount of empirical evaluation that we have performed, not only on well-modelled synthetic data, but also on an actual-life clinical dataset of colon cancer drug response, would leave no doubt that the BDNN paradigm is superior. The model was able to achieve continuously state-of-the-art performances with the smallest error measures (RMSE, MAE) and a large probabilistic score (NLL) in comparison to high-powered baselines, including classical beta regression, OLS, and traditional neural networks. This underscores the reality that BDNN possesses two features, namely, the ability to make precise point predictions and good uncertainty quantifications. The BDNN model can thus be shown to be a very useful and versatile tool for limited outcome prediction. It is also foreshadowing colossal advances in fields such as biomedicine and healthcare analytics, and economics and the social sciences, where probabilistic forecasting may be applied to make smart choices. Probabilistic forecasting is crucial, correct and understandable.

References

[1] Ahmed, S. F., Alam, M. S. B., Hassan, M., Rozbu, M. R., Ishtiak, T., Rafa, N., Mofijur, M., Shawkat Ali, A., & Gandomi, A. H. (2023). Deep learning modelling techniques: current progress, applications, advantages, and challenges. Artificial Intelligence Review, 56(11), 13521-13617.

[2] Bouraya, S., & Belangour, A. (2024). A comparative analysis of activation functions in neural networks: unveiling categories. Bulletin of Electrical Engineering and Informatics, 13(5), 3301-3308.

[3] Bunel, R., Lu, J., Turkaslan, I., Torr, P. H., Kohli, P., & Kumar, M. P. (2020). Branch and bound for piecewise linear neural network verification. Journal of Machine Learning Research, 21(42), 1-39.

[4] Cangalovic, V. S., Thielke, F., & Meine, H. (2024). Comparative evaluation of uncertainty estimation and decomposition methods on liver segmentation. International journal of computer assisted radiology and surgery, 19(2), 253-260.

[5] Chilimbi, T., Suzue, Y., Apacible, J., & Kalyanaraman, K. (2014). Project adam: Building an efficient and scalable deep learning training system. 11th USENIX symposium on operating systems design and implementation (OSDI 14),

[6] Cribari-Neto, F., & Zeileis, A. (2010). Beta regression in R. Journal of statistical software, 34(1), 1-24.

[7] Dureja, A., & Pahwa, P. (2019). Analysis of non-linear activation functions for classification tasks using convolutional neural networks. Recent Patents on Computer Science, 12(3), 156-161.

[8] Ferrari, S., & Cribari-Neto, F. (2004). Beta regression for modelling rates and proportions. Journal of applied statistics, 31(7), 799-815.

[9] Glorot, X., & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. Proceedings of the thirteenth international conference on artificial intelligence and statistics,

[10] Gneiting, T., & Katzfuss, M. (2014). Probabilistic forecasting. Annual Review of Statistics and Its Application, 1(1), 125-151.

[11] Gupta, A. K., & Nadarajah, S. (2004). Handbook of beta distribution and its applications. CRC press.

[12] Jais, I. K. M., & Ismail, A. R. (2019). Adam optimization algorithm for wide and deep neural network. Knowledge Engineering and Data Science, 2(1), 10.

[13] Kieschnick, R., & McCullough, B. D. (2003). Regression analysis of variates observed on (0, 1): percentages, proportions and fractions. Statistical modelling, 3(3), 193-213.

[14] Kim, S., Yu, Z., Kil, R. M., & Lee, M. (2015). Deep learning of support vector machines with class probability output networks. Neural Networks, 64, 19-28.

[15] Mobiny, A., Yuan, P., Moulik, S. K., Garg, N., Wu, C. C., & Van Nguyen, H. (2021). Dropconnect is effective in modeling uncertainty of bayesian deep networks. Scientific reports, 11(1), 5458.

[16] Neal, P., Eric, C., Borja, P., & Jonathan, E. (2011). Distributed optimization and statistical learning via the alternating direction method of multipliers. Foundations and Trends® in Machine learning, 3(1), 1-122.

[17] Paolino, P. (2001). Maximum likelihood estimation of models with beta-distributed dependent variables. Political Analysis, 9(4), 325-346.

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

[19] Rasp, S., Pritchard, M. S., & Gentine, P. (2018). Deep learning to represent subgrid processes in climate models. Proceedings of the national academy of sciences, 115(39), 9684-9689.

[20] Roustaei, N. (2024). Application and interpretation of linear-regression analysis. Medical Hypothesis, Discovery and Innovation in Ophthalmology, 13(3), 151.

[21] Sigrist, F. (2022). Gaussian process boosting. Journal of Machine Learning Research, 23(232), 1-46.

[22] Simon, H. A. (2000). Bounded rationality in social science: Today and tomorrow. Mind & Society, 1(1), 25-39.

[23] Smithson, M., & Verkuilen, J. (2006). A better lemon squeezer? Maximum-likelihood regression with beta-distributed dependent variables. Psychological methods, 11(1), 54.

[24] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research, 15(1), 1929-1958.

[25] Ding, N., Benoit, C., Foggia, G., Bésanger, Y., & Wurtz, F. (2015). Neural network-based model design for short-term load forecast in distribution systems. *IEEE transactions on power systems*, *31*(1), 72-81.