https://doi.org/10.56093/JISAS.V78I2.11
Available online at http://isas.org.in/isa/jisas
JOURNAL OF THE INDIAN SOCIETY OF AGRICULTURAL STATISTICS 78(2) 2024 169–178

DDC: Deep Distribution Classifier, A Convolutional Neural Network-based Approach for Identifying Data Distributions

Samarth Godara, Avinash G, Rajender Parsad and Sudeep Marwaha

ICAR-Indian Agricultural Statistics Research Institute, New Delhi

Received 19 May 2024; Revised 23 July 2024; Accepted 02 August 2024

SUMMARY

In domains such as the stock market and manufacturing, there's a growing demand for faster and more accurate data distribution identification methods due to the rapid generation of vast volumes of data, highlighting the need for enhanced real-time decision-making capabilities. Traditional methods of identifying data distributions often rely on manual inspection, limited statistical tests and time-consuming analysis, leading to inefficiencies and inaccuracies in classification. In this scenario, the presented research offers a novel approach leveraging Deep Learning (DL) models to automate the process. The presented methodology also enables faster and more accurate identification of data distributions by the generation of synthetic data points and training of the DL model for identifying different distribution types. The primary objective of this study is to develop a DL model that categorizes data points into specific distributions based on an input dataset. Moreover, for model training and evaluation, a total of 1000 datasets are generated, each comprising 1000 data points. The study considers five distributions (Normal, Uniform, Exponential, Log-normal and Beta distribution), with 200 datasets generated (with randomly selected parameters) for each distribution. In the study, the DL model is trained first, and later, the model is evaluated on a separate test (unseen) dataset. Then, its performance in classifying the distributions is assessed based on metrics such as accuracy and loss. The study results demonstrate the effectiveness of the proposed approach in accurately classifying the distribution of data points, providing valuable insights into the application of DL for distribution classification tasks. The proposed method enhances scalability, robustness and efficiency by harnessing the power of convolutional neural networks and advanced preprocessing techniques.

Keywords: Distribution identification; Deep learning; Data distribution classification; Convolutional neural networks; Fast distribution detection.

1. INTRODUCTION

Distribution identification is indispensable across domains such as finance, healthcare, manufacturing, and more. In finance, understanding data distributions aids in risk assessment, portfolio management, and financial modelling [1, 11]. It facilitates disease diagnosis, treatment optimization, and epidemiological studies in healthcare [2, 12]. Furthermore, it optimizes production processes, ensures product quality, and minimizes downtime in manufacturing [3]. Across all domains, accurate distribution identification underpins informed decision-making, process optimization, and performance enhancement.

Understanding the data distribution helps identify underlying patterns by revealing the frequency and distribution of different values within the dataset. This understanding enables analysts to recognize trends, correlations, and relationships between variables. Identifying anomalies becomes easier when the expected data distribution is known, as any deviations from this distribution can signal potential errors, outliers, or unusual events. Moreover, optimizing processes relies on understanding the data distribution to identify areas for improvement, streamline operations, and enhance efficiency. Making informed decisions also benefits from knowledge of the data distribution, as it provides context for interpreting results, evaluating risks, and selecting appropriate strategies based on the dataset's characteristics.

The traditional approach to identifying data distribution involves statistical methods such as histograms, box plots and more. Analysts manually inspect data visualizations and perform hypothesis tests such as the Kolmogorov-Smirnov or Shapiro-

Corresponding author: Samarth Godara
E-mail address: samarth.godara@icar.gov.in

Wilk tests. This method relies on assumptions about the underlying distribution and may require multiple iterations to determine the best-fitting distribution [4]. However, it can be time-consuming and subjective, and complex patterns in the data may need to be captured more effectively. Moreover, the traditional method of distribution identification relies on assumptions about the underlying distribution, which may only sometimes hold true in real-world datasets. Manual inspection of data visualizations and statistical tests can be time-consuming and subjective, leading to potential biases and inaccuracies. Overall, these issues highlight the need for more efficient and automated approaches to distribution identification.

Moreover, there is a pressing demand for alternative methods in data distribution identification that offer both speed and accuracy. This need arises particularly in domains like the stock market and manufacturing plants, where vast volumes of data are generated rapidly. Traditional methods often fall short in handling the sheer scale and velocity of data production in these domains. As a result, there is a growing urgency to develop faster and more precise techniques to identify data distributions, enabling real-time decision-making and optimization processes in these critical sectors.

In this scenario, Machine Learning and Deep Learning (ML/DL) emerge as one potential solution. ML/DL models are increasingly utilized for classification tasks across diverse domains due to their ability to learn complex patterns from data. In healthcare, these models aid in disease diagnosis, medical image analysis, and personalized treatment recommendations [5]. In finance, they enable fraud detection, credit risk assessment, and algorithmic trading strategies [6]. In manufacturing, they optimize quality control processes, predictive maintenance, and supply chain management [7]. Across domains, ML/DL models empower automated decision-making, enhance efficiency, and drive innovation.

Generally, multi-class classification refers to an ML/DL task that aims to classify input data points into one of several predefined classes or categories. In this task, each data point can belong to only one class out of multiple possible classes. Inputs to a multi-class classification task typically consist of features or attributes that describe the characteristics of the data points. These features could be numerical values, categorical variables, or even more complex data structures like images or text. The output of a multi-class classification task is the predicted class label for each input data point. The model assigns a probability distribution over all possible classes, and the class with the highest probability is considered the predicted class for that data point. The output is often represented as a vector of probabilities, where each element corresponds to the probability of the corresponding class.

The proposed approach leverages ML/DL models to automate the classification of data distributions. By preprocessing data points, scaling them to a uniform range and employing Convolutional Neural Networks (CNNs) followed by densely connected layers, the model aims to accurately identify the underlying distribution among common options such as normal, uniform, exponential, log-normal and beta distributions. This automated approach offers scalability, efficiency and robustness, addressing the limitations of traditional manual methods and providing valuable insights across various industries.

The selection of only five distributions provides a diverse yet manageable set of common statistical distributions, ensuring the model is trained on a representative sample of distribution types. If researchers wish to expand the classification task, they must incorporate additional distributions and their respective synthetic datasets into the training process, potentially adjusting the model architecture to handle increased complexity. Conversely, reducing the classification task would involve excluding some distributions and retraining the model to ensure it maintains high accuracy with fewer classes, potentially simplifying the model's architecture.

The novel contributions of this research work include:

- Proposing an automated approach for identifying data distributions using ML/DL models.
- Implementing convolutional neural networks (CNNs) followed by densely connected layers for efficient and accurate classification of data distributions.
- Introduces a preprocessing step to scale data points to a uniform range and employs sorting techniques to enhance classification accuracy.
- Demonstrating the effectiveness of the proposed approach in overcoming limitations of traditional manual methods, providing

scalable and robust solutions for distribution identification across diverse domains.

In the current scenario, the challenges faced by current researchers in performing the task of identifying data distributions include the absence of a framework guiding data preparation to aid ML/DL models in predicting data distribution. Additionally, there's a lack of systems to train ML/DL models for distribution classification and generate data points for model development encompassing training, validation, and testing phases. These challenges were addressed by proposing an automated approach, introducing preprocessing steps, implementing CNNs followed by densely connected layers, and demonstrating the approach's effectiveness in overcoming the limitations of traditional manual methods.

2. METHODOLOGY

The proposed methodology automates the process of generating synthetic data points, performing distribution tests, preprocessing the data, training a neural network model, and evaluating its performance for the classification of data distributions. The whole study can be divided into two parts, i.e., the Data Preparation stage and the Model Development and Evaluation stage (Fig. 1).

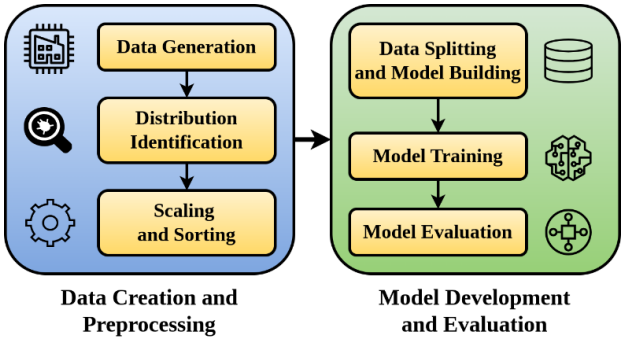

Fig. 1. Methodology undertaken to develop the proposed Deep Distribution Classifier (DDC)

1. Data preparation

   a. Data Generation: In the data preparation stage of the study, several steps are undertaken to generate synthetic data points, perform distribution tests, and preprocess the data for training the neural network model. Initially, the process begins with data generation, where empty lists are initialized to store input data points and corresponding p-values for distribution tests, respectively. The number of desired generations is set to 1000 (overall, 1000 datasets are generated in the simulation, where each dataset contains 1000 values, which belong to either one of the undertaken distributions). Moreover, a counter is initialized to track the number of generated data points. Within a loop, data points are generated for each distribution type (normal, exponential, uniform, log-normal, beta) using random parameters sampled from specified ranges [8, 13-15]. Description of the distributions undertaken in the presented study is as follows:

I. A normal distribution, also known as Gaussian distribution, is a symmetric probability distribution where data is symmetrically distributed around the mean. Its formula is given by eq. (1).

$$f(x|\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \quad (1)$$

where, μ represents the mean and σ represents the standard deviation. In general, the mean can range from negative infinity to positive infinity, covering all real numbers. The standard deviation must be a non-negative real number, so its range typically starts from zero and extends to positive infinity.

II. The exponential distribution describes the time between events in a Poisson process, where events occur continuously and independently at a constant average rate over time. Its probability density function (PDF) is given by eq. (2).

$$f(x;\lambda) = \lambda e^{-\lambda x} \quad (2)$$

where, x is the time, and λ is the rate parameter. Its range is limited to positive real numbers, as it dictates the rate at which events occur over time.

III. The uniform distribution represents outcomes where each value within a given range has equal probability. Its PDF is defined as given in eq. (3).

$$f(x;a,b) = \frac{1}{b-a}, \ for\ a \le x \le b \quad (3)$$

where, a and b are the lower and upper bounds of the range, respectively. Where a and b are real numbers and $a < b$.

IV. The log-normal distribution is characterized by data whose logarithms follow a normal

distribution, often representing skewed data. Its PDF is defined as given in eq. (4).

$$f(x;\mu,\sigma) = \frac{1}{x\sigma\sqrt{2\pi}} e^{-\frac{(\ln x-\mu)^2}{2\sigma^2}} \quad (4)$$

where, μ is the mean and σ is the standard deviation of the logarithm of the variable x. Here μ can take any real value, and σ must be greater than 0.

V. The beta distribution represents the variability of probabilities over a fixed interval, commonly used for modelling proportions or rates. Its PDF is defined as given in eq. (5).

$$f(x;\alpha,\beta) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{B(\alpha,\beta)} \quad (5)$$

where, α and β are shape parameters, and $B(\alpha,\beta)$ is the beta function. Both α and β must be positive real numbers greater than zero.

For each distribution type, the computer program invokes the `generate_test_values` function to perform distribution tests and obtain p-values. If the p-value exceeds the threshold of 0.05, indicating a valid distribution, the data points and corresponding p-values are appended to the dataset. This process iterates until the desired number of valid data points is generated.

In our study, random selection of population distribution parameters refers to the process of selecting parameters such as scale, location, shape and rate for each distribution from predefined ranges of (0-2), (0-1), (1-2), and (0-1), respectively. This randomness allows for the generation of diverse data points representing various distributions. For instance, in the normal distribution, a random selection of mean (location) and standard deviation (scale) produces data points with different central tendencies and spreads. Similarly, in the uniform distribution, random selection of minimum and maximum values (location and scale) generates data points uniformly distributed within the specified range. This approach ensures that each distribution is represented by a wide range of data points, enhancing the model's ability to effectively learn and classify different distribution patterns.

b. In the data generation process, the `generate_test_values` function is crucial in conducting distribution tests for the generated data points. It takes the data points along with parameters such as scale, loc, a, and b as inputs. A dictionary inside the function is initialized to store p-values obtained from different distribution tests. Necessary functions from `scipy.stats` python library are imported to perform tests for normal, uniform, exponential, log-normal, and beta distributions. For each distribution, the function executes the respective test and stores the resulting p-value in the dictionary. Finally, the function returns the dictionary containing p-values for each distribution test.

c. Preprocessing and Scaling: Following data generation and distribution testing, the data undergoes preprocessing and scaling to prepare it for training the neural network model. The input data points are scaled from 0 to 1 using the `MinMaxScaler`, a preprocessing technique commonly employed in machine learning tasks. The equation for Min-Max scaling to scale data between 0 and 1 is given by eq. (6).

$$X_{\text{scaled}} = \frac{X - X_{\min}}{X_{\max} - X_{\min}} \quad (6)$$

Where, X_{scaled} is the scaled value, X is the original value, $X_{\min}$ is the minimum value in the dataset and $X_{\max}$ is the maximum value in the dataset. Subsequently, the scaled data points are sorted to ensure uniformity and consistency in the input data. This preprocessing step is crucial as it standardizes the input data and enhances the convergence and performance of the neural network model during training. Overall, the data preparation stage lays the foundation for subsequent model training and evaluation by generating synthetic data, conducting distribution tests, and preparing the data for machine learning algorithms. After scaling the data points, the values are sorted in ascending order. Sorting is used as a preprocessing step to standardize the input data, making patterns within different distributions more apparent to the deep learning model. By arranging data points in a specific order, the inherent characteristics of each distribution become more distinguishable, aiding the model in learning and identifying distribution-specific features more effectively. This step enhances the model's ability to discern subtle

differences between distributions, thereby improving classification accuracy and robustness.

2. Model Development and Evaluation Stage

a. Data splitting and Model building: In this step, the data is split into training, validation, and testing sets, a common practice in ML/DL that assesses model performance on unseen data. The training dataset is used to train the model's parameters, the validation dataset is used to tune hyperparameters and prevent overfitting, and the testing dataset is used to evaluate the model's performance on unseen data. In our study, a ratio of 80:10:10 was used for training, validation, and testing datasets, respectively, to ensure adequate model training, tuning, and evaluation while maintaining a balance between dataset sizes.

Subsequently, a convolutional neural network (CNN) model is defined, which allows for the sequential stacking of layers. The architecture of the CNN model comprises a convolutional layer, a max-pooling layer to downsample feature maps, a flattening layer to convert the multidimensional data into a vector, densely connected layers to perform classification, and a softmax output layer for multi-class classification [9]. The CNN model utilized in the study is designed for feature extraction and hierarchical learning from input data. Its core equation involves convolving input data with learnable filters, followed by activation functions and pooling operations to extract relevant features. This process is represented in eq. (7).

$$\text{Convolution: } Z^{[l]} = W^{[l]} * A^{[l-1]} + b^{[l]} \quad (7)$$

Where, $Z^{[l]}$ represents the output of the convolution operation at layer l, $W^{[l]}$ denotes the learnable filters (also called kernels) specific to layer l, $A^{[l-1]}$ is the activation map from the previous layer, $l-1$, $b^{[l]}$ is the bias term associated with the convolutional layer l and $*$ denotes the convolution operation, where the filter is applied to the input data.

b. Model Training: Following the model architecture definition, the model is compiled using the Adam optimizer and categorical cross-entropy loss function, which is suitable for multi-class classification tasks. With the compiled model, training commences on the training data for a specified number of epochs, in this case, 100 epochs. During training, the model learns to map input data to their corresponding output classes, iteratively adjusting its parameters to minimize the loss function. Here, backpropagation is a crucial algorithm used to train the compiled CNN model. It calculates the gradient of the loss function with respect to the network weights, enabling weight updates to minimize the error. The algorithm operates in two phases: the forward pass, where inputs are fed through the network to make predictions, and the backward pass, where errors are propagated backwards through the network to update the weights. The formula for updating weights using backpropagation is given by eq. (8).

$$\Delta w_{ij} = -\eta \frac{\partial E}{\partial w_{ij}} \quad (8)$$

Where, Δw_{ij} is the change in weight between neuron i and neuron j, η is the learning rate, $\frac{\partial E}{\partial w_{ij}}$ is the partial derivative of the error with respect to the weight w_{ij}.

An activation function in a neural network defines the output of a neuron given an input or set of inputs. It introduces non-linearity into the model, allowing it to learn complex patterns and make accurate predictions. The model undertaken in the study uses two widely used activation function, i.e. ReLU activation function and the softmax activation function.

The ReLU (Rectified Linear Unit) activation function is defined as $\text{ReLU}(x) = \max(0, x)$. It outputs the input directly if it is positive; otherwise, it outputs zero, aiding in faster and more effective training by mitigating the vanishing gradient problem. The softmax activation function is defined as given in eq. (9).

$$\text{softmax}(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}} \quad (9)$$

It converts a vector of raw scores into probabilities, with each value ranging between 0 and 1, and their sum equal to 1, making it suitable for multi-class classification tasks.

c. Model Evaluation: After training, the model's performance is evaluated on the unseen test data, which computes the loss and accuracy metrics. These metrics provide insights into how well the model generalizes to new, unseen data. The obtained loss and accuracy metrics are displayed

to quantitatively assess the model's performance. In multi-class classification CNN models, the loss metric commonly used is categorical crossentropy, which calculates the difference between predicted and actual class probabilities across all classes [10]. Its formula is given by eq. (10).

$$\text{Categorical Crossentropy Loss} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C} y_{ij}\log(p_{ij}) \quad (10)$$

Here, N is the number of samples, C is the number of classes, y_{ij} is the indicator function (1 if sample i belongs to class j, 0 otherwise), and p_{ij} is the predicted probability of sample i belonging to class j. Moreover, accuracy is a common metric used to evaluate model performance, representing the proportion of correctly classified samples out of the total samples. Its formula is given by eq. (11).

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}} \quad (11)$$

For further validation and demonstration purposes, the code predicts the output for the first 5 test samples using the trained model and prints both the true and predicted values. This step allows for a qualitative assessment of the model's predictive capabilities.

d. Model Storage: Lastly, to ensure the model's reusability and accessibility, it is saved to a file. This enables easy retrieval and utilization of the trained model for future tasks without retraining it from scratch. Additionally, there is also the functionality to load the saved model back into memory, facilitating seamless integration into other projects or applications. Overall, these steps encompass the model development, evaluation, and management processes, ensuring the robustness and usability of the trained neural network model.

3. RESULTS AND DISCUSSION

The architecture of the model developed in the study consists of a sequential stack of layers (Fig. 2). The initial layer is a 1-dimensional convolutional layer with 64 filters, a kernel size of 3, and ReLU activation function. It takes input data of shape (1000, 1), where 1000 represents the number of data points and 1 represents the number of features. Following the convolutional layer is a max-pooling layer with a pool size of 2, which reduces the spatial dimensions of the input. The output of the pooling layer is flattened into a one-dimensional array using the `Flatten` layer. Then, two densely connected layers are added, the first with 32 units and ReLU activation function, and the second with 5 units and softmax activation function, representing the output classes. The model is compiled using the Adam optimizer, categorical cross-entropy loss function, and accuracy metric for evaluation.

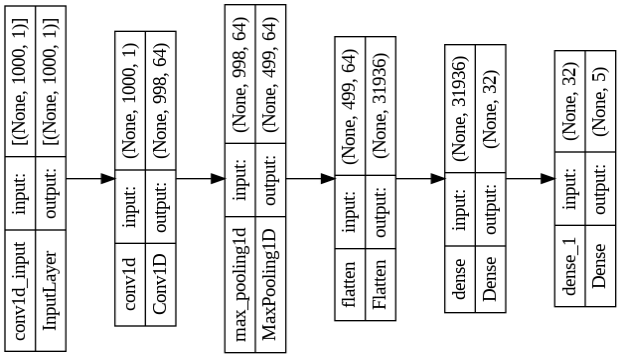

Fig. 2. Architectural details of the developed CNN-based Distribution Classification Model

Fig. 3 represents the performance metrics of the CNN-based model trained on a dataset over multiple epochs. The X-axis corresponds to a specific epoch during the training process, and the y-axis presents the loss and accuracy values achieved by the model on the training dataset at each epoch. From the figure, it is observed that, initially, at epoch 10, the model has a relatively high loss value of 1.2468 and a low accuracy of 0.3827, indicating poor performance. However, as training progresses, the loss steadily decreases, reaching significantly lower values by epoch 50 (0.056) and continuing to decrease thereafter. Similarly, the model's accuracy improves over time, starting from 0.3827 at epoch 10 and reaching a high of 0.9988 by epoch 60. Towards the end of training, the loss plateaus, indicating that the model has converged to a stable state where further training may not significantly improve performance. Similarly, the accuracy also stabilizes, indicating that the model has learned to classify the training data accurately.

The figure also presents the performance metrics evaluated on a validation dataset across different epochs during the training process. From the figure, it is observed that at the beginning of training (epoch 10), the model's loss is relatively high (0.7236), indicating that the model's predictions deviate significantly from the actual values. However, the accuracy is

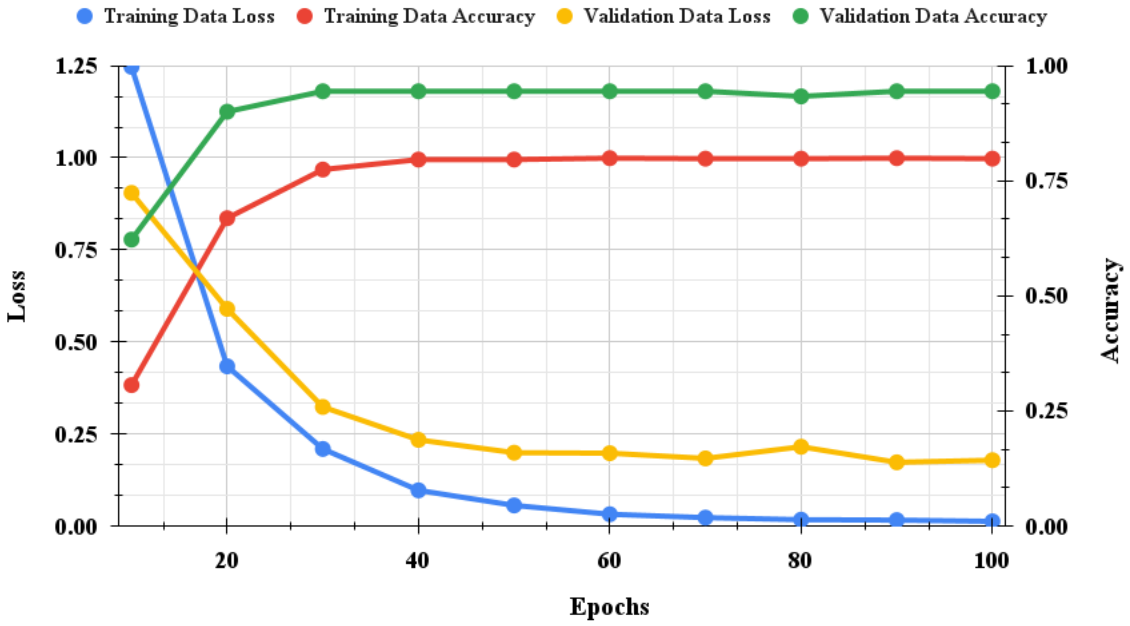

Fig. 3. The Accuracy and Loss across the 100 epochs employed for training the model

moderate (0.6222), indicating that a substantial portion of the data points is correctly classified. As training progresses, both loss and accuracy improve. By epoch 30, the loss decreases to 0.2587, indicating better model performance in minimizing prediction errors. Additionally, the accuracy increases to 0.9444, indicating that a higher proportion of data points is correctly classified. Beyond epoch 30, the loss decreases gradually, reaching its lowest value at epoch 90 (0.1384), indicating further improvement in predictive accuracy. However, there is a slight fluctuation in loss and accuracy values in later epochs, suggesting some instability in the model's performance. Overall, the validation dataset's loss decreases over epochs while accuracy remains relatively stable, indicating that the model is improving its ability to make accurate predictions. The consistency in accuracy values suggests that the model's performance is robust and generalizes well to unseen data.

Fig. 4 presents the performance metrics of a trained CNN-based model on three different datasets: training, validation, and testing. From the figure, it is observed that the loss on the training dataset is 0.0127, which indicates the average discrepancy between the model's predictions and the actual values in the training dataset. The accuracy on the training dataset is 0.9975, indicating the proportion of correctly classified data points out of the total number of data points in the training dataset. A high accuracy value indicates that the model makes accurate predictions on the training data. The loss on the validation dataset is 0.143, with the accuracy on the validation dataset being 0.9444, indicating the proportion of correctly classified data points out of the total number of data points in the validation dataset. A high accuracy value indicates that the model performs well on the validation data, although it may not be as high as the training accuracy.

The loss on the testing dataset is 0.0551; the loss value is intermediate between the training and validation losses, indicating that the model performs reasonably well on unseen data. The accuracy on the testing dataset is 0.9901, indicating the proportion of correctly classified data points out of the total number of data points in the testing dataset. A high accuracy value suggests that the model makes predictions on unseen data, demonstrating its ability to generalize well. Overall, the CNN model demonstrates high accuracy and low loss on both the training and testing datasets, indicating its effectiveness in making accurate predictions.

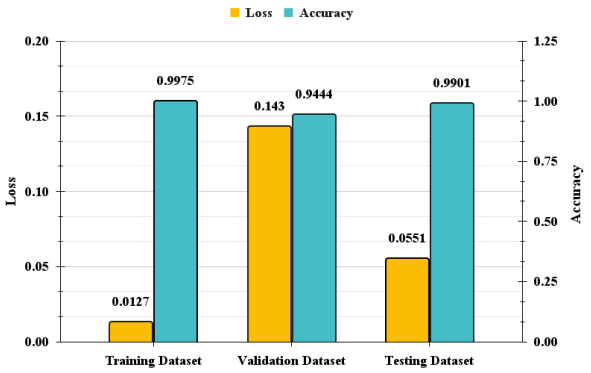

Fig. 4. Accuracy and Loss values of the trained model on the training, validation and testing dataset

Table 1 represents the CNN-based model's desired output and predicted output for different distributions: Normal, Uniform, Exponential, Log-Normal, and Beta. From the table, it is observed that for the desired output of 1 (indicating a normal distribution), the predicted outputs are relatively high, ranging from 0.993 to 0.952. This suggests the model correctly identifies data points in the normal distribution with high confidence. Similarly, for the desired output of 1 (indicating an exponential distribution, log-normal and beta distribution), the predicted outputs are relatively high, close to 1 (0.994, 0.986 and 0.828, respectively). This suggests that the model correctly identifies data points of these distributions with high confidence. Overall, the model performs well in identifying the different types of distributions, with high confidence in the cases.

While the model demonstrates proficiency, it's essential to acknowledge that real-world datasets often exhibit characteristics of mixture models, emphasizing the need for nuanced approaches in subsequent research endeavours. The future scope of this research includes exploring the application of more complex neural network architectures to enhance classification accuracy. Additionally, integrating techniques for handling mixed distributions could improve the model's performance in real-world scenarios. Further investigations could also expand the study to encompass a broader range of distribution types and dataset sizes for comprehensive analysis.

4. DISCUSSION

In the presented study, the preprocessing step (scaling and sorting of the data points) has played a vital role in the classification process. Scaling the data points to a uniform range (between 0 and 1) prevents certain features from dominating the learning process merely due to their scale, thereby improving the stability and convergence of the model during training. Moreover, scaling the data points helps mitigate the effects of varying magnitudes and units among different features, making the optimization process more efficient. This, in turn, can lead to improved model performance, faster convergence, and better generalization to unseen data.

In addition, sorting the data points further enhances the training process; when the data points are sorted, similar patterns and relationships are grouped together, allowing the model to learn more effectively from the sequential structure of the data. This can result in faster convergence and more accurate classification of different distributions. Furthermore, sorting the data points can also help reduce overfitting by exposing the model to a more structured data representation. By presenting the data in a sorted order, the model is less likely to memorize noise or irrelevant patterns and

Table 1. Samples of the desired and predicted output values corresponding to the unseen data points

S.No.	Output type	Normal Distribution	Uniform Distribution	Exponential Distribution	Log-Normal Distribution	Beta Distribution
1	Desired output	1	0	0	0	0
	Predicted output	0.993	<0.001	<0.001	<0.001	<0.001
2	Desired output	0	0	1	0	0
	Predicted output	<0.001	<0.001	0.994	<0.001	<0.001
3	Desired output	1	0	0	0	0
	Predicted output	0.952	<0.001	<0.001	<0.001	<0.001
4	Desired output	0	0	0	0	1
	Predicted output	<0.001	<0.001	<0.001	<0.001	0.828
5	Desired output	0	0	0	1	0
	Predicted output	<0.001	<0.001	<0.001	0.986	<0.001

instead focuses on learning the underlying distributional characteristics.

The performance analysis of the developed CNN-based model over multiple epochs reveals notable trends. Initially, the model shows high loss and low accuracy, but as training progresses, both metrics improve steadily. The convergence of loss and stabilization of accuracy indicates the model's ability to make accurate predictions on the training dataset. Evaluation of the validation dataset further confirms the model's performance, with gradual improvements observed over epochs. Moreover, the performance metrics on different datasets—training, validation, and testing—demonstrate the model's effectiveness in making accurate predictions. The high accuracy and low loss values on both training and testing datasets underscore the model's robustness and ability to generalize well to unseen data. Additionally, the model's proficiency in identifying various distributions, as evidenced by the desired and predicted output values, showcases its capabilities in distribution classification tasks.

In our study, the 1D CNN-based model was deemed a better fit for several reasons. The input data represented sequential information, where each data point had a linear relationship with its neighbouring points. A 1D CNN is well-suited for processing sequential data, making it an appropriate choice for our dataset. Moreover, the 1D convolutional layer in the CNN model performs feature extraction by convolving filters over the input sequence. This capability allows the model to capture patterns and relationships within the data, which is crucial for identifying different distributions. Compared to Recurrent Neural Networks [16] or Long Short-Term Memory networks [17], 1D CNNs typically have fewer parameters, making them computationally efficient. This efficiency is beneficial for training on large datasets and allows for faster experimentation and model iteration.

Although the proposed method demonstrates effectiveness on synthetic datasets, it may face challenges in generalizing to real-world data, which often contain noise, outliers, and more complex structures. Therefore, the authors intend to validate the proposed methodology on real-world data in future studies. Additionally, the study's focus on five specific distributions limits the model's ability to accurately classify data points from other, untrained distributions.

In conclusion, while our study demonstrates the effectiveness of CNN-based models for distribution classification tasks, future research should explore more complex neural network architectures and techniques for handling mixed distributions to enhance model performance in real-world scenarios. Expanding the study to encompass a broader range of distribution types and dataset sizes could provide more comprehensive insights into distribution identification across diverse domains.

5. CONCLUSION

In various domains, there is a pressing need for faster methods of identifying data distributions to enable timely decision-making and enhance operational efficiency. Traditional approaches to data distribution identification are often time-consuming and labour-intensive, hindering real-time analysis and decision-making processes in various domains. In this direction, the study aimed to develop a DL-based model for classifying the distribution of data points. The study offers novel insights by employing the DL algorithm to classify data distributions, addressing the need for automated and efficient identification methods across various domains. Additionally, it introduces a comprehensive approach that combines data generation, preprocessing, model development, and evaluation to accurately classify different distribution types. Through the generation of synthetic data points and training of the CNN model, the study evaluates the performance of the classification algorithm in identifying different distribution types (Normal, Uniform, Exponential, Log-normal, and Beta distribution). Overall, the results demonstrated that the DL-based approach showed promising results in classifying the distribution of data points accurately. The model achieved high accuracy on both training and validation datasets, indicating its ability to generalize well to unseen data. The study highlights the potential of using ML/DL techniques for automated distribution identification tasks, which can have significant applications in various domains such as finance, manufacturing, and research. Further research could focus on refining the model architecture, exploring additional features, and testing the model on larger and more diverse datasets to enhance its performance and applicability.

ACKNOWLEDGEMENTS

The authors are grateful to the reviewers for significant improvement of the articles.

REFERENCES

AbouRizk, Simaan M., Daniel W. Halpin, and James R.(1994). Wilson. "Fitting beta distributions based on sample data." Journal of Construction Engineering and Management 120, No. 2. 288-305.

Adcock, Christopher, Martin Eling, and Nicola Loperfido.(2015). "Skewed distributions in finance and actuarial science: a review." The European Journal of Finance 21, no. 13-14. 1253-1281.

Aghdam, Hamed Habibi, and Elnaz Jahani Heravi.(2017). "Guide to convolutional neural networks." New York, NY: Springer 10, no. 978-973. 51.

Davis, Paula M.(2020). "Statistics for describing populations." In Handbook of sampling methods for arthropods in agriculture, pp. 33-54. CRC Press, 2020.

Fraile, Roberto, and Eduardo García-Ortega.(2005). "Fitting an exponential distribution." Journal of Applied Meteorology and Climatology 44, No. 10. 1620-1625.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. Deep learning. MIT press, 2016.

Habehh, Hafsa, and Suril Gohel.(2021). "Machine learning in healthcare." Current genomics 22, No. 4: 291.

Huang, Jian, Junyi Chai, and Stella Cho.(2020. "Deep learning in finance and banking: A literature review and classification." Frontiers of Business Research in China 14, No. 1 : 13.

Medsker, Larry R., and Lakhmi Jain.(2001). "Recurrent neural networks." Design and Applications 5, No. 64-67. 2.

Mohammed, Noman, Benjamin C.M. Fung, Patrick C.K. Hung, and Cheuk-Kwong Lee.(2010). "Centralized and distributed anonymization for high-dimensional healthcare data." ACM Transactions on Knowledge Discovery from Data (TKDD) 4, No. 4: 1-33.

Rai, Rahul, Manoj Kumar Tiwari, Dmitry Ivanov, and Alexandre Dolgui.(2021). "Machine learning in manufacturing and industry 4.0 applications." International Journal of Production Research 59, No. 16: 4773-4778.

Ramberg, John S., Edward J. Dudewicz, Pandu R. Tadikamalla, and Edward F. Mykytka.(1979). "A probability distribution and its uses in fitting data." Technometrics 21, No. 2 : 201-214.

Stedinger, J.R.(1980). "Fitting log normal distributions to hydrologic data." Water Resources Research 16, No. 3: 481-490.

Thas, Olivier. Comparing distributions. Vol. 233. New York: Springer, 2010.

Vose, David.(2016). "Fitting distributions to data." Retrieved March 8, 2016.

Wang, Junliang, Chuqiao Xu, Jie Zhang, and Ray Zhong.(2022). "Big data analytics for intelligent manufacturing systems: A review." Journal of Manufacturing Systems 62, 738-752.

Yu, Yong, Xiaosheng Si, Changhua Hu, and Jianxun Zhang.(2019). "A review of recurrent neural networks: LSTM cells and network architectures." Neural computation 31, No. 7 1235-1270.