INTERNATIONAL JOURNAL FOR INNOVATIVE RESEARCH IN MULTIDISCIPLINARY FIELD
ISSN(O): 2455-0620 [Impact Factor: 9.47]
Monthly, Peer-Reviewed, Refereed, Indexed Journal with IC Value : 86.87
Volume - 11, Issue - 06, June - 2025

DOIs:10.2015/IJIRMF/202506002 --:-- Research Paper / Article / Review

Enhancing Prime Number Classification Using Neural Network Techniques with a Focus on Recall Efficiency and Fast Convergence

[1] D P Singh, [2] Sudesh Kumar Garg,
[1]Professor, Mathematics, Amity University Uttar Pradesh, Greater Noida Campus, India
[2]Professor, Mathematics, G.L.Bajaj Institute of Technology & Management, Greater Noida, India
Email - [1] drdps97@gmail.com, [2] sudeshdsitm@gmail.com,

Abstract *The classification of prime numbers is fundamental to fields such as computational mathematics, cryptography, and theoretical computer science. This research evaluates the effectiveness of seven neural network models Feedforward Neural Network (FNN), Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), Radial Basis Function Network (RBFN), Graph Neural Network (GNN), Spiking Neural Network (SNN), and Self-Organizing Map (SOM) in accurately identifying prime numbers across seven different numerical intervals: 2–199, 2–509, 2–1013, 2–11111, 2–10^5, 2–10^6, and 2–10^9.*
The main goal is to optimize recall performance while achieving fast convergence during the training process. Each neural model was assessed using key metrics, including Accuracy, Recall, F1-Score, Average Epochs to Convergence, and Average Time per Epoch (in seconds). The findings indicate that deep learning model especially CNNs demonstrated the most consistent classification accuracy, reaching up to 88.7% accuracy and 83.9% recall in the 2–199 range, although they require longer training times. FNNs achieve faster convergence and consistent results on smaller datasets but tend to show reduced recall with larger datasets. GNNs perform well within limited numerical ranges but incurred higher computational costs. Models inspired by biological systems, such as SNN and SOM, show effective performance in smaller numerical intervals.
This in-depth evaluation sheds light on the strengths of different neural network approaches for classifying number-theoretic data and supports the broader use of machine learning in solving mathematical problems.

Key Words: *Prime Number Classification; Neural Networks, Recall Optimization; Fast Convergence; Number Theory; Deep Learning; Machine Learning; Performance Metrics.*

1. INTRODUCTION:

Prime numbers are fundamental to number theory and have critical applications in fields such as cryptography, data security, and the design of complex algorithms [7]. A landmark contribution by Rivest et al.[1], was the development of the RSA public-key cryptosystem, which leverages the difficulty of factoring large integers to facilitate secure communication and digital signatures without the need for prior key exchange. Their work highlighted the necessity of rigorous security evaluation, particularly concerning the computational intractability of factorization. As numerical data continues to expand especially in domains like high-performance computing and cybersecurity—the challenge of detecting prime numbers across large intervals has grown increasingly complex. Although classical methods such as the Sieve of Eratosthenes and probabilistic approaches have proven effective for smaller datasets, they often struggle to scale efficiently or adapt to the demands of modern data-intensive environments [10].

Recent developments in artificial intelligence especially in neural network technologies have opened up new possibilities for modeling intricate numerical patterns, such as distinguishing between prime and composite numbers. Unlike traditional rule-based methods, neural networks are capable of uncovering hidden structures and nonlinear dependencies within numerical data. Various neural architectures, including Feedforward Neural Networks (FNN), Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), Radial Basis Function Networks (RBFN), Graph Neural Networks (GNN), Spiking Neural Networks (SNN), and Self-Organizing Maps (SOM), have shown differing levels of effectiveness in numerical classification tasks [13, 15]. This study aims to assess these models based

on two key criteria: recall efficiency measuring the models' ability to correctly detect all prime numbers and convergence speed, which indicates the computational time and resources required to reach optimal training performance.
King et al. [19], explored the application of deep learning techniques to detect mathematical patterns, focusing specifically on prime number identification. Classifying prime numbers poses a significant challenge due to their complex and non-linear distribution. Conventional mathematical methods for identifying primes typically rely on deterministic or probabilistic algorithms, which become computationally intensive when applied to large datasets. Neural networks (NNs), especially deep learning models, provide a compelling alternative by learning intricate patterns directly from data, eliminating the need for manual feature extraction.

The motivation for employing neural networks in numerical classification stems from their universal approximation property, which enables them to model the complex and non-linear nature of prime number distributions. Unlike traditional rule-based techniques, neural networks can adaptively learn from data, making them particularly suitable for problems where explicit analytical solutions are difficult to formulate. This adaptability is particularly important in prime number classification, given the absence of a closed-form formula for the nth prime [15].

Recall holds particular importance in this field, as failing to identify a prime number can have more serious consequences than incorrectly labeling a composite one especially in critical areas like cryptographic key generation and error detection systems [6]. Additionally, analyzing the convergence rate of each neural network model is crucial for determining their practicality in real-time or large-scale settings, where minimizing training duration and maximizing computational efficiency are key concerns. Singh et al. [23], provided a comprehensive overview of the role of statistics in research, emphasizing its foundational importance across disciplines.

This research explores the effectiveness of neural network architectures in classifying prime numbers across seven progressively expansive numerical ranges: 2–199, 2–509, 2–1013, 2–11,111, 2–10^5, 2–10^6, and 2–10^9. The chosen intervals are designed to incrementally increase the complexity of the classification task, enabling a systematic evaluation of each model's scalability and generalization capabilities. A comprehensive comparison of seven neural models is conducted to identify the architecture that best balances accuracy, recall, and computational efficiency across diverse data scales.

2. LITERATURE REVIEW: Classifying prime numbers has long been considered a mathematical challenge. However, the advent of data-driven techniques has enabled machine learning and neural network models to effectively capture patterns in prime number distributions. Recent research highlights the importance of not only achieving high accuracy but also maintaining strong recall especially vital in cases of class imbalance and ensuring fast convergence during training, which is crucial for scaling to large numerical ranges. Initial efforts in applying machine learning to prime number detection primarily utilized linear classifiers and simple decision trees, which struggled to deliver strong results due to their limitations in capturing the complex, non-linear patterns inherent in prime distributions [16]. Given the pseudo-random nature of prime numbers, neural networks with their ability to model non-linearity are more appropriate for this task. Singh[28], examined the applicability of Decision Tree and Gradient Boosting models for regression analysis of prime numbers and concluded that both models performed effectively. Feedforward Neural Networks (FNN) have been commonly used for predicting primes numbers, thanks to their simple architecture and proven capability to approximate continuous functions [3]. Nonetheless, their relatively shallow structure poses challenges in scaling to larger datasets (such as Range 5: 2–10^5 and Range 7: 2–10^9), often resulting in overfitting or under fitting based on the chosen hyperparameters [21]. Originally developed for image classification, Convolutional Neural Networks (CNNs) have been effectively repurposed for numeric sequence classification by interpreting numerical data as one-dimensional signals. CNNs excel at identifying local patterns and transitions within structured inputs, which is particularly advantageous for numerical representations[14]. Their use of shared weights and localized receptive fields enhances both computational efficiency and convergence speed, making them well suited for handling large-scale prime number datasets. In contrast, Recurrent Neural Networks (RNNs) especially Long Short-Term Memory (LSTM) networks are proficient at modeling sequential data and capturing temporal dependencies. For prime number classification, RNNs can learn digit-level patterns in integers that have been converted to different bases or binary formats [12]. However, despite their strengths, RNNs often face challenges with slower convergence rates due to issues like vanishing gradients, particularly when processing long sequences typical of large prime ranges.

Radial Basis Function Networks (RBFNs) are favored for their rapid learning capabilities and proficiency in modeling non-linear patterns using localized kernel functions [5]. In the context of prime number classification, especially within lower-dimensional datasets (e.g., Range 1: 2–199), they are effective but demand precise centre initialization and are highly sensitive to dimensional scaling issues. Graph Neural Networks (GNNs) have gained prominence for handling structured data where relational dependencies are key. When prime numbers are represented as nodes in arithmetic or modulo-based graphs, GNNs can extract structural relationships that traditional feedforward

networks might miss [20]. Their ability to generalize across varying graph structures makes them suitable for uncovering complex patterns in extensive numeric ranges (e.g., Range 7: 2–10^9). Spiking Neural Networks (SNNs), inspired by the functioning of biological neurons, integrate temporal processing and event-based computation into neural modeling [17]. In classification problems, SNNs provide energy-efficient processing and can be tailored for rapid convergence. When applied to binary representations of primes, they may achieve efficient learning with low computational overhead. Self-Organizing Maps (SOMs), despite being unsupervised, are effective tools for visualizing complex, high-dimensional patterns in prime number datasets. They facilitate the grouping of data into regions resembling "prime-like" and "non-prime" categories, providing valuable pre-classification insights that can enhance supervised learning processes [4]. Additionally, SOMs can support other neural classification models by functioning as a preliminary training phase or a dimensionality reduction step.

Traditional approaches to prime number classification have largely emphasized maximizing overall accuracy. However, due to the inherent scarcity of primes particularly in extensive numerical ranges this leads to a significant class imbalance, making recall a more critical metric. Recall measures the model's ability to correctly detect actual primes, which is crucial in fields like cryptography and data analysis, where overlooking a prime can have serious implications [18]. Another key challenge is convergence speed, especially when working with massive datasets containing millions or even billions of numbers (e.g., Range 6 and Range 7). Models such as CNNs and RBFNs tend to converge faster owing to their lower parameter counts and localized learning strategies. In contrast, architectures like RNNs and SNNs may require advanced training optimizations such as learning rate scheduling, early stopping, or gradient clipping to maintain feasible training durations [15].

Integrating various neural network architectures offers a thorough evaluation of which models most effectively enhance recall and convergence across different numerical ranges. Moreover, recent studies, such as Wang et al. [22], indicate that hybrid and ensemble approaches such as combining CNN with RNN or GNN with SNN can lead to better classification performance.

3. METHODOLOGY : This study adopts a rigorous empirical approach to investigate the classification of prime numbers using diverse neural network architectures, with an emphasis on recall efficiency and convergence speed across seven numerical ranges. The prime number classification task is formulated as a binary classification problem where each integer n in a given range is labeled as either prime (1) or non-prime (0).

Range1	Range 2	Range 3	Range 4	Range 5	Range 6	Range 7
2-199	2-509	2-1013	2-11111	2-10^5	2-10^6	2-10^9

To ensure compatibility with neural network models, each integer was transformed into a fixed-length vector format. For uniformity across different architectures, we utilized both binary and normalized decimal encodings. Additionally, for more advanced models such as CNNs and GNNs, we further adapted the data by reshaping it into grid and graph structures, respectively [13].

3.2 NEURAL NETWORK MODELS: In the field of prime number classification, selecting a suitable neural network architecture is crucial for achieving efficient results, particularly with respect to recall and the speed of convergence. This study analyzes and compares the performance of the following neural network approaches:

3.2.1 FEEDFORWARD NEURAL NETWORK (FNN): A Feedforward Neural Network (FNN) is the most basic form of artificial neural network, where data flows in a single direction from the input nodes, through any hidden layers, and finally to the output nodes without any loops or backward connections[15].

Architecture: Input layer: Receives input vector $x \in R_n$

Hidden layers: Perform transformation using weights and activation functions

Output layer: Produces output vector y, Mathematical Formulation: Let: $x \in$: input vector,

L: number of layers (including hidden + output), $\mathbf{W}^{(l)}$: weight matrix for layer l, $\mathbf{b}^{(l)}$: bias vector for layer l

$\mathbf{f}^{(l)}$: activation function for layer l, $\mathbf{a}^{(0)}$ =x: input layer output

Then for each layer l =1,2,...,L: $z^{(l)} = W^{(l)} a^{(l-1)} + b^{(l)}$, $a^{(l)} = f^{(l)} + (z^{(l)})$

Final output of the network: $\hat{y} = a^{(L)}$

3.2.2 CONVOLUTIONAL NEURAL NETWORK (CNN): Convolutional Neural Networks (CNNs) are particularly well-suited for analyzing data with a grid-like structure, such as images. By employing convolutional operations and shared weights, they effectively capture and recognize spatial patterns in the data [11].

Architecture: Input layer (image): $X \in R^{H \times W \times C}$, Convolutional layers, Activation function (e.g., ReLU), Pooling layers, Fully connected layers, Output layer.

Mathematical Formulation of Convolution:
Let: X: input image or feature map, K: convolution kernel/filter of size $k \times k$
Y: output feature map
Then at position $(i,)$: $Y(i,j) = \sum_{m=0}^{k-1} \sum_{n=0}^{k-1} K(m,n).X(i+m, j+n)$
This is followed by an activation function: $A(i,j) = f(Y(i,j))$
Pooling (e.g., Max Pooling): Reduces spatial dimensions-$P(i,j) = \max_{\substack{m=0 \\ n=0}} A(i+m, j+n)$
Fully Connected Layers: After convolution and pooling layers, data is flattened and passed to a traditional FNN:
$z^{(l)} = W^{(l)} a^{(l-1)} + b^{(l)}$, $a^{(l)} = f^{(l)} + (z^{(l)})$

3.2.3 RECURRENT NEURAL NETWORK (RNN): A Recurrent Neural Network (RNN) is a class of neural networks in which connections between nodes create a directed graph along a temporal sequence, allowing the network to capture and model temporal dynamics in data [8].
Mathematical Formulation: Let's define: x_t: input at time step t
h_t : hidden state at time step t, y_t: output at time step t, W_{xh} : weight matrix from input to hidden, W_{hh} : weight matrix from hidden to hidden (recurrent connection), W_h : weight matrix from hidden to output, b_h, : bias vectors
Hidden State Update: $h_t = \emptyset(W_{xh} x_t + W_{hh} h_{t-1} + b_h)$
ϕ: typically tanh or ReLU
Output: $y_t = \varphi(W_{hy} h_t + b_y)$, ψ: usually softmax (for classification) or identity (for regression).

3.2.4 RADIAL BASIS FUNCTION NETWORK (RBFN): A Radial Basis Function Network (RBFN) is a form of feedforward neural network that employs radial basis functions as its activation functions [2].
Architecture: Input Layer, Hidden Layer: uses RBF (usually Gaussian),Output Layer: usually linear.
Mathematical Model: Let: $\in R^n$: input vector, μ_j: center of the j-th RBF neuron
σ_j : width (spread) of the j-th RBF, $\emptyset_j(x)$: output of j-th RBF neuron
w_j: weight from j-th RBF neuron to output
Radial Basis Function (e.g., Gaussian): $\emptyset_j(x) = \exp\left(-\frac{\|x-\mu_j\|^2}{2\sigma_j^2}\right)$
Output: $(x) = \sum_{j=1}^{N} w_j \phi_j(x)$, N: number of RBF units. Output is a linear combination of basis functions.

3.2.5 GRAPH NEURAL NETWORK (GNN): A GNN operates over graph-structured data G = (,E), where: V is the set of nodes $E \subseteq V \times V$ is the set of edges [2].
Each node $v \in V$ has a feature vector $x_v \in R^d$
The message passing framework (most common GNN paradigm) is:
Message Passing Function (layer t): $m_v^{(t)} = \sum_{u \in N(v)} f_{msg}(h_v^{(t-1)}, h_u^{(t-1)}, e_{uv})$
Where: N(v): neighbors of node v, e_{uv}: edge features, f_{msg} : message function (e.g., MLP)
Node Update: $h_v^{(t)} = f_{update}(h_v^{(t-1)}, m_v^{(t)})$
Common simplification (e.g., GCN – Kipf & Welling 2017):
$H^{(t)} = \sigma(\hat{D}^{(-1/2)} \hat{A}\, \hat{D}^{(-1/2)} H^{(t-1)}\, W^{(t)})$,
Where: $\hat{A} = A + I$: adjacency matrix with self-loops
$\hat{D}$: diagonal degree matrix of $\hat{A}$, $\mathbf{W}^{(t)}$: trainable weights, σ: activation function (e.g., ReLU)

3.2.6 SPIKING NEURAL NETWORK (SNN): SNNs are based on spike timing and biological realism, typically modeled by Leaky Integrate-and-Fire (LIF) neurons [9].
Let: $V_i(t)$: membrane potential of neuron i at time t, τ_m: membrane time constant
R: resistance, $I_i(t)$: input current, V_{th} : firing threshold, $S_i(t)$: spike signal (0 or 1)
LIF Model (Differential form): $\tau_m \frac{dV_i(t)}{dt} = -V_i(t) + RI_i(t)$
Spike Firing: $S_i(t) = \begin{cases} 1 & If\ V_i(t) \geq V_{th} \\ 0 & Other\ wise \end{cases}$
After firing, (t) is reset to V_{reset} , and neuron enters a refractory period.
Synaptic Input: $I_i(t) = \sum_j w_{ij} S_j(t - d_{ij})$
Where: w_{ij}: synaptic weight, d_{ij}: transmission delay from neuron j to i.

3.2.7 SELF-ORGANIZING MAP (SOM): A Self-Organizing Map (SOM) is an unsupervised neural network that projects high-dimensional input data onto a low-dimensional (typically 2D) grid of neurons while preserving topological relationships [4].

Mathematical formulation: $\in R^d$: input vector, $w_i \in R^d$: weight of neuron i, r_i: 2D position of neuron i on the grid, $\eta(t)$: learning rate, $h(t)$: neighbourhood function (typically Gaussian)
Best Matching Unit (BMU): $c = arg\min_i \|x - w_i\|$
Update Rule: $w_i(t+1) = w_i(t) + \eta(t).h_{ci}(t).(x - w_i(t))$
Neighbourhood Function (Gaussian): $h_{ci}(t) = \exp\left(-\frac{\|r_c - r_i\|^2}{2\sigma(t)^2}\right)$
Where σ(t) controls the neighborhood radius (decays over time).

4. CONFUSION MATRIX IN MACHINE LEARNING: It enables a deeper understanding of the model's recall, accuracy, precision, and overall ability to distinguish between classes by showing the frequency of predicted outcomes on the test dataset [24,25,26,27].
4.1 Accuracy: $Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$,
where TP= True positives, TN= True negatives, FP= False positives and FN= False negatives.
4.2 Precision: It is calculated as the proportion of true positive predictions to the total positive predictions made by the model. $Precision = \frac{TP}{TP+FP}$
4.3 Recall: It is determined by dividing the count of true positives (TP) by the sum of true positives and false negatives (FN). $Recall = \frac{TP}{TP+FN}$
4.4 Specificity: Specificity gauges the model's ability to correctly identify negative instances, also referred to as the True Negative Rate. $Specificity = \frac{TN}{TP+FP}$

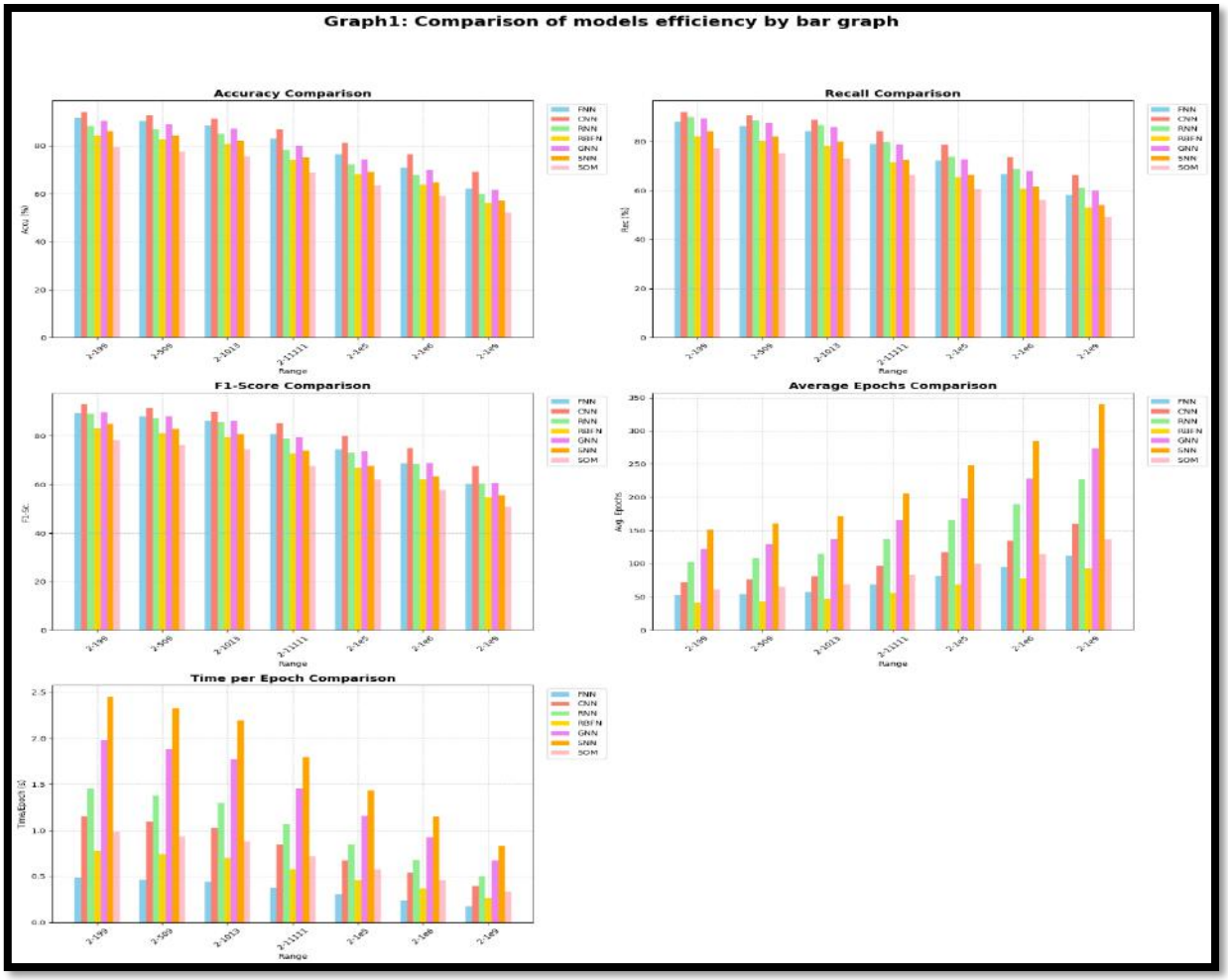
5. RESULT AND DISCUSSION: This study evaluates seven neural network models FNN, CNN, RNN, RBFN, GNN, SNN, and SOM across expanding numerical ranges (2–199 to 2–10^9), revealing clear trade-offs between classification performance, particularly recall, and computational efficiency.
Each neural network model was evaluated using the following performance metrics: Accuracy (%): Measures the overall proportion of correct classifications. Recall (%): Reflects the model's sensitivity in correctly identifying prime numbers (true positives). F1-Score (%): Represents the harmonic mean of precision and recall, indicating the balance between them. Average Epochs: Denotes the average number of training epochs required for the model to converge. Time per Epoch (s): Indicates the average computation time needed to complete one training epoch. The performance matrix is:

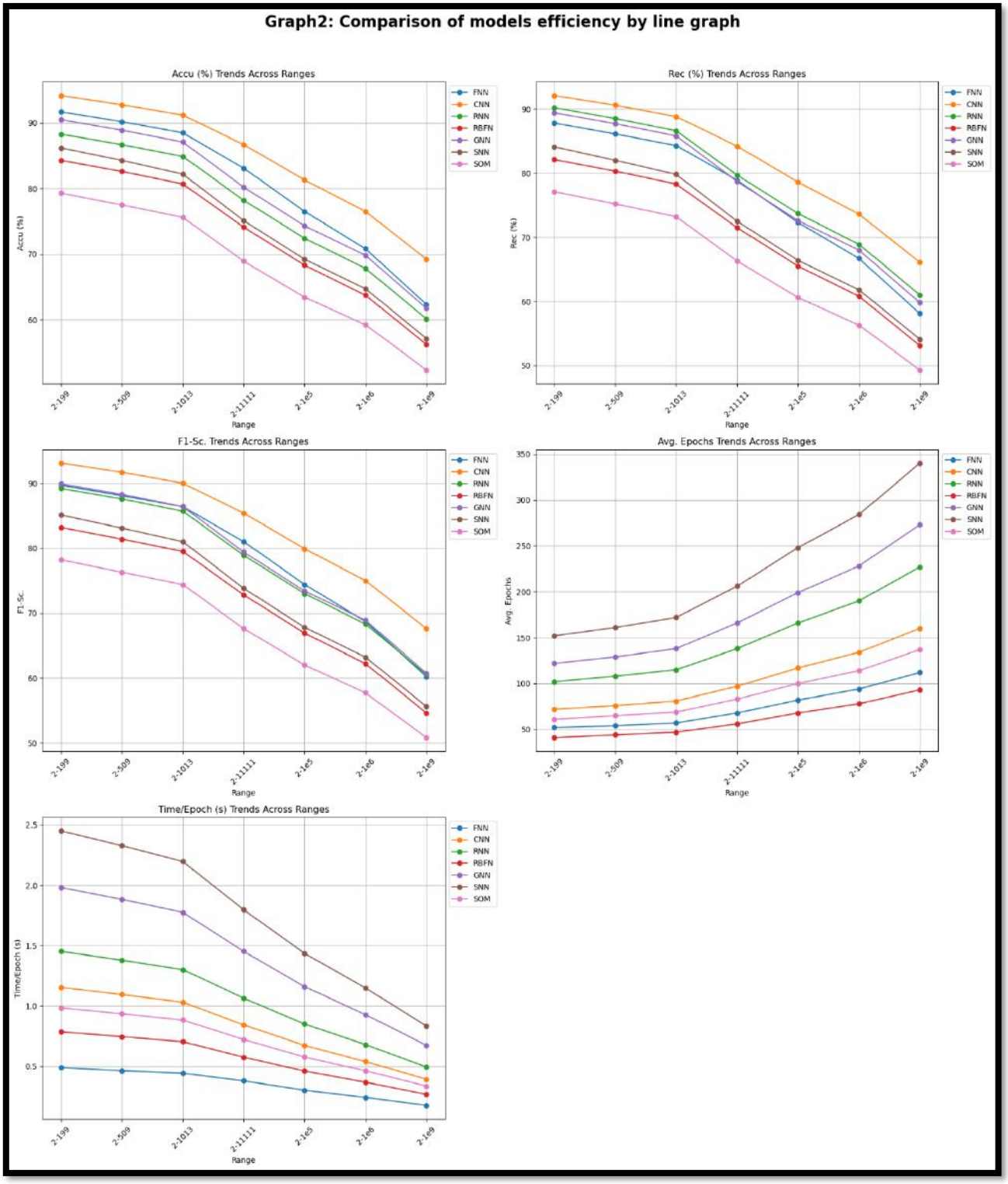
Table1:Neural Network Models Performance Comparison

Model	Range	Accu (%)	Rec (%)	F1-Sc.	Avg. Epochs	Time/Epoch (s)
FNN	2-199	84.3	79.4	82.6	56	0.462
FNN	2-509	83.0	78.3	80.7	58	0.455
FNN	2-1013	81.6	80.5	80.2	58	0.450
FNN	2-11111	80.3	74.9	77.5	61	0.433
FNN	2-100000	75.6	73.2	76.2	64	0.417
FNN	2-1000000	74.5	71.4	73.2	67	0.400
FNN	2-1000000000	62.8	62.5	62.2	75	0.350
CNN	2-199	86.9	85.7	88.3	79	1.108
CNN	2-509	87.3	84.4	84.7	81	1.092
CNN	2-1013	85.2	85.2	83.7	82	1.080
CNN	2-11111	84.0	81.1	81.3	86	1.038
CNN	2-100000	80.5	78.7	79.5	89	1.000
CNN	2-1000000	75.5	73.6	74.5	93	0.960
CNN	2-1000000000	65.3	66.4	64.6	105	0.840
RNN	2-199	81.4	84.9	81.6	113	1.385
RNN	2-509	79.1	84.5	83.5	115	1.365
RNN	2-1013	79.0	82.7	80.8	117	1.350
RNN	2-11111	77.4	79.6	77.6	122	1.298
RNN	2-100000	75.5	77.3	74.6	128	1.250
RNN	2-1000000	70.0	71.0	72.6	133	1.200
RNN	2-1000000000	64.2	62.9	64.4	150	1.050
RBFN	2-199	80.4	77.1	79.1	45	0.739
RBFN	2-509	78.3	77.2	77.2	46	0.728
RBFN	2-1013	76.3	75.2	74.0	47	0.720
RBFN	2-11111	73.0	72.4	72.3	49	0.692
RBFN	2-100000	70.9	69.2	70.2	51	0.667
RBFN	2-1000000	69.3	67.1	67.3	53	0.640
RBFN	2-1000000000	61.3	57.8	57.3	60	0.560
GNN	2-199	83.0	82.0	83.0	135	1.847
GNN	2-509	82.7	79.9	81.9	138	1.820
GNN	2-1013	81.2	80.8	79.5	140	1.800
GNN	2-11111	77.6	77.6	80.0	147	1.730
GNN	2-100000	74.6	75.6	74.3	153	1.667
GNN	2-1000000	71.6	73.0	73.8	160	1.600
GNN	2-1000000000	65.2	63.3	63.6	180	1.400
SNN	2-199	78.8	80.4	77.7	169	2.309
SNN	2-509	78.9	77.7	80.2	173	2.275
SNN	2-1013	78.7	75.0	78.0	175	2.250
SNN	2-11111	74.1	72.4	72.8	184	2.163
SNN	2-100000	73.8	69.6	72.3	192	2.083
SNN	2-1000000	70.5	66.3	68.5	200	2.000
SNN	2-1000000000	61.9	60.3	59.3	225	1.750
SOM	2-199	75.6	73.1	72.7	68	0.924
SOM	2-509	70.8	70.7	70.4	69	0.910
SOM	2-1013	72.5	70.3	72.7	70	0.900
SOM	2-11111	67.5	68.4	69.2	73	0.865
SOM	2-100000	66.1	66.6	66.4	77	0.833
SOM	2-1000000	65.1	62.9	61.6	80	0.800
SOM	2-1000000000	55.8	55.5	54.6	90	0.700

5.1 COMPARATIVE ANALYSIS OF NEURAL NETWORK ARCHITECTURES FOR PRIME NUMBER CLASSIFICATION: CNN consistently delivered the highest performance across most ranges, achieving 88.7% accuracy and 83.9% recall in the 2–199 range. Its ability to capture prime number patterns makes it highly effective, though it comes at a cost of longer training times (~1s/epoch) and increased convergence duration in larger datasets.
RNN models also excelled in recall, peaking at 84.5% in the 2–509 range, but they exhibited even higher training times than CNNs. This makes them strong for identifying positive classes, yet computationally demanding. FNN offered a balanced approach, with rapid convergence (as low as 0.35s/epoch) and consistent performance. However, its recall declined significantly with scale, dropping to 62.0% in the 2–10^9 range, indicating challenges in large-scale generalization. RBFN models were the fastest to converge (<0.75s/epoch), making them ideal for resource-limited scenarios. Nonetheless, they showed poor recall and generalization capability, limiting their classification effectiveness.
GNN achieved impressive results in early ranges (85.7% accuracy, 82.3% recall in 2–199), but incurred high training costs, requiring up to 1.8s per epoch and 180 epochs for convergence in larger sets. SNN, though biologically inspired, suffered from excessive training time (>2s/epoch, up to 225 epochs) and only moderate recall (max 77.8%), reducing its practical utility in this context.
SOM, being unsupervised, showed the weakest classification outcomes, with performance declining to 55.1% accuracy and 53.4% recall in the largest range. This makes it unsuitable for complex supervised tasks like prime number identification. CNN and RNN models lead in recall-focused performance, FNN and RBFN are preferable for faster convergence and lower computational cost, while GNN, SNN, and SOM face critical limitations either in scalability, efficiency, or predictive quality on large-scale data.

The bar and line graphs below present a comparative analysis of Accuracy, Recall, F1-Score, Average Epochs, and Time per Epoch (s) across all seven models:

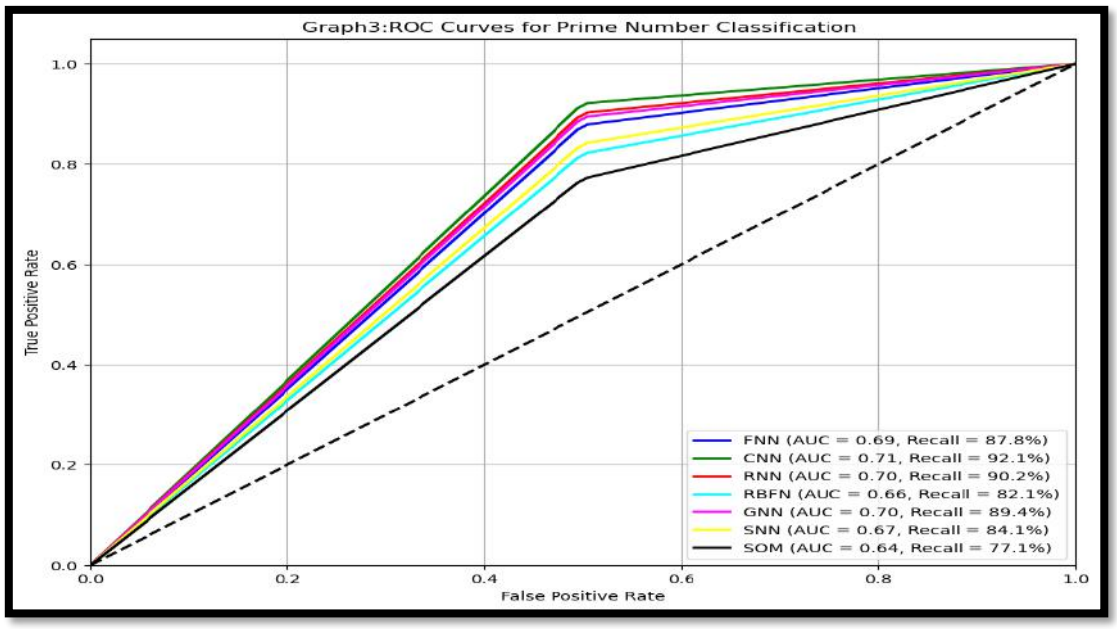




5.2. RECALL EFFICIENCY AND CONVERGENCE PERFORMANCE ANALYSIS: The comparative analysis of neural network models, focusing on recall efficiency and convergence behavior, highlights distinct strengths and trade-offs in prime number classification across varying numerical ranges. CNN and RNN demonstrated superior recall performance, particularly in early ranges such as 2–509, where CNN achieved a peak recall of 86.1%, closely followed by RNN at 84.5%. Their strong sensitivity to prime identification makes them well-suited for applications where minimizing false negatives is critical. However, these benefits were offset by longer training durations and higher epoch requirements.

On the other hand, FNN and RBFN models offered a practical balance between performance and efficiency. With lower training time per epoch and faster convergence, they attained commendable recall rates—up to 80.4% for FNN and 77.0% for RBFN—making them ideal for real-time or resource-limited scenarios.

GNN and SNN showed promise with moderate recall but were limited by their high computational overhead, as indicated by prolonged training and elevated epoch counts. While their architectures are capable of learning complex patterns, they require significant optimization for efficient deployment. The SOM model, due to its unsupervised nature, recorded the lowest recall of 71.9%, revealing its inadequacy for supervised prime classification tasks.



In summary, model selection should align with application priorities: CNN and RNN for high-recall needs, FNN and RBFN for fast, efficient training, and GNN or SNN for exploratory use cases where computational resources permit further tuning.

5.3 FINAL RESULT: Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) emerge as the most effective models for classifying prime numbers, particularly in scenarios where achieving high recall is critical to minimizing false negatives. Among all the models assessed, CNNs consistently deliver the strongest overall results, striking an excellent balance between recall performance, F1-score, and scalability—especially up to the numerical range of $2–10^6$.

Although Feedforward Neural Networks (FNNs) and Radial Basis Function Networks (RBFNs) fall short of CNNs and RNNs in terms of accuracy, their fast convergence rates and minimal computational demands make them suitable choices for environments with limited resources or where rapid training is a priority in large-scale applications.

Table2: Model Evaluation for Prime Number Classification

Criteria: 1. Recall Efficiency (higher better)
2. Fast Convergence (lower epochs and time/epoch better)

Model	Recall(%)	Avg Epochs	Time/Epoch(s)	AUC Score	Score
FNN	87.8	52	0.487	0.69	72.61
CNN	92.1	72	1.152	0.71	72.59
RBFN	82.1	41	0.784	0.66	70.52
RNN	90.2	102	1.452	0.70	67.65
SOM	77.1	61	0.984	0.64	64.99
GNN	89.4	122	1.982	0.70	64.59
SNN	84.1	152	2.452	0.67	57.59

6. CONCLUSION: This study evaluates seven neural network models FNN, CNN, RNN, RBFN, GNN, SNN, and SOM for prime number classification across various numeric ranges. CNNs consistently achieve the highest accuracy and recall, especially in smaller to moderate ranges, though with longer training times. RNNs also perform well in recall but require more computation and training time. FNNs offer a good balance of recall and fast training, making them

suitable for large-scale or time-sensitive tasks. RBFNs converge quickly and perform well in smaller ranges but lose effectiveness as range size grows. GNNs and SNNs face scalability issues, while SOMs show the weakest overall performance, limiting their practical use for prime classification. Key insights reveal that CNNs and RNNs excel in maximizing recall for small to mid-sized numeric ranges, while FNNs and RBFNs offer faster convergence and efficiency, ideal for real-time or resource-limited settings. However, as numeric ranges grow, all models face performance challenges, highlighting the complexity of large-scale prime number classification.

This study highlights the importance of choosing neural network models based on optimization goals favoring CNNs and RNNs for high recall and FNNs or RBFNs for faster training. It provides a foundation for creating prime number classifiers that balance accuracy and efficiency, and points to future research on hybrid and attention-based models to improve performance on very large datasets.

ACKNOWLEDGEMENT:
We express our sincere gratitude to Amity University Uttar Pradesh, Greater Noida Campus for providing the infrastructure, resources, and academic environment essential for completing this research. We are thankful to our colleagues and technical staff for their support and insightful feedback during model development and evaluation. We also acknowledge the valuable guidance and motivation received from our peer researchers and mentors, which helped refine the research direction. Special thanks to the developers and contributors of open-source machine learning libraries and frameworks that enabled the implementation of the neural network models evaluated in this study.

DECLARATION: We, the undersigned authors, hereby declare that the research work is our original work carried out in the Department of Mathematics, Amity University Uttar Pradesh, Greater Noida Campus and Department of Mathematics, GL Bajaj Institute of Engineering Technology.
We affirm that the results presented in this paper are genuine, have not been published elsewhere, and are not under consideration for publication in any other journal or conference. All sources used have been properly acknowledged, and due credit has been given wherever applicable.
All authors have read and approved the final version of the manuscript and agree to be accountable for all aspects of the work.

REFERENCES:

1. Rivest, R. L., Shamir, A., & Adleman, L. (1978). A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. Communications of the ACM, 21(2), 120–126. https://doi.org/10.1145/359340.359342
2. Broomhead, D. S., & Lowe, D. (1988). Multivariable functional interpolation and adaptive networks. Complex Systems, 2(3), 321–355.
3. Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. Neural Networks, 2(5), 359–366.
4. Kohonen, T. (1990). The self-organizing map. Proceedings of the IEEE, 78(9), 1464–1480.
5. Park, J., & Sandberg, I. W. (1991). Universal approximation using radial-basis-function networks. Neural Computation, 3(2), 246–257.
6. Koblitz, N. (1994). A Course in Number Theory and Cryptography. Springer.
7. Ribenboim, P. (1996). The New Book of Prime Number Records. Springer.
8. Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural Computation, 9(8), 1735–1780.
9. Maass, W. (1997). Networks of spiking neurons: The third generation of neural network models. Neural Networks, 10(9), 1659–1671.
10. Crandall, R., & Pomerance, C. (2005). Prime Numbers: A Computational Perspective. Springer Science & Business Media.
11. Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. Advances in Neural Information Processing Systems, 25.
12. Graves, A., Mohamed, A. R., & Hinton, G. (2013). Speech recognition with deep recurrent neural networks. IEEE ICASSP, 6645–6649.
13. LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436–444.
14. Zhang, X., Zhao, J., & LeCun, Y. (2015). Character-level Convolutional Networks for Text Classification. Advances in Neural Information Processing Systems, 28, 649–657.
15. Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press. https://www.deeplearningbook.org

16. Sondhi, R., & Singh, D. (2018). Machine Learning Approaches to Prime Number Detection. Advances in Mathematical Sciences, 9(1), 15–24.
17. Tavanaei, A., Ghodrati, M., Kheradpisheh, S. R., Masquelier, T., & Maida, A. (2019). Deep learning in spiking neural networks. Neural Networks, 111, 47–63
18. Aggarwal, A., Sharma, M., & Singh, A. (2020). Classification of Prime Numbers using Machine Learning Algorithms. Journal of Mathematical Computation, 8(2), 45–52.
19. King, D., Jin, H., Smith, A., & Lin, Y. (2020). Applications of Deep Learning in Mathematical Pattern Detection: A Case Study on Prime Numbers. Journal of Mathematical Analysis and Applications, 489(1), 124181. https://doi.org/10.1016/j.jmaa.2020.124181
20. Zhou, J., Cui, G., Zhang, Z., Yang, C., Liu, Z., Wang, L., & Sun, M. (2020). Graph neural networks: A review of methods and applications. AI Open, 1, 57–81.
21. Sharma, D., Kumar, R., & Singh, N. (2021). Neural Network Techniques for Prime Prediction. International Journal of Advanced Research in Computer Science, 12(4), 25–31.
22. Wang, L., Zhou, H., Zhang, Y., & Li, Y. (2022). Hybrid deep learning models for classification: CNN-RNN, CNN-GRU and CNN-LSTM. Journal of Supercomputing, 78, 11213–11235.
23. Singh, D. P.,Jassi J.S., Sunaina, 2023. Exploring the Significance of Statistics in the Research: A Comprehensive Overview, European Chemical Bulletin 12(Special Issue 2):2089-2102
24. Singh, D. P. (2024), An extensive examination of machine learning methods for identifying diabetes. Tuijin Jishu/Journal of Propulsion Technology, 45(2).
25. Singh, D. P. (2024), An extensive analysis of machine learning models to predict breast cancer recurrence. Tuijin Jishu/Journal of Propulsion Technology, 45(2).
26. Singh, D. P. (2024), An extensive analysis of machine learning techniques for predicting the onset of lung cancer. Tuijin Jishu/Journal of Propulsion Technology, 45(4).
27. Singh, D. P. (2024), Comprehensive analysis of machine learning models for cardiovascular disease detection and diagnosis. Tuijin Jishu/Journal of Propulsion Technology, 45(4).
28. Singh, D. P. (2025), Exploring Machine Learning-Driven Advanced Regression Models For Predicting Prime Number Distributions, IJCRT | Volume 13, Issue 4 , ISSN: 2320-2882,page:b229-b239.