Optimizer-Based Performance Evaluation of Deep Learning Models for Loan Eligibility Prediction

Banala Saritha
NVIDIA-CoE of Artificial Intelligence & Machine Learning
Department of Electronics and Communication Engineering
B V Raju Institute of Technology
Medak, India
saritha.b@bvrit.ac.in

Thummagunta Vasantha Rani
NVIDIA-CoE of Artificial Intelligence & Machine Learning
Department of Electronics and Communication Engineering
B V Raju Institute of Technology
Medak, India
22211a04p6@bvrit.ac.in

Vaddi Venkata Dinesh
NVIDIA-CoE of Artificial Intelligence & Machine Learning
Department of Electronics and Communication Engineering
B.V. Raju Institute of Technology
Medak, India
22211a04q1@bvrit.ac.in

Yadam Srujan Kumar
NVIDIA-CoE of Artificial Intelligence & Machine Learning
Department of Electronics and Communication Engineering
B.V. Raju Institute of Technology
Medak, India
22211a04q7@bvrit.ac.in

Yaramanedi Jayanth Kumar
NVIDIA-CoE of Artificial Intelligence & Machine Learning
Department of Electronics and Communication Engineering
B.V. Raju Institute of Technology
Medak, India
22211a04r0@bvrit.ac.in

***Abstract*— Accurate loan eligibility prediction is imperative for financial institutions to mitigate default risk and optimize credit decision processes. Traditional manual verification methods are increasingly unsustainable due to the surge in loan applications and the need for rapid, reliable assessments. This research presents a comprehensive comparative analysis of deep learning models—including Deep Neural Networks (DNN), Convolutional Neural Networks (CNN), and DenseNet architectures—trained using various optimization algorithms (Adam, SGD, RMSprop, and Adagrad) to predict loan eligibility. The models were evaluated across multiple performance metrics, namely accuracy, precision, recall, F1-score, and ROC-AUC, to ensure a robust assessment. Experimental results reveal that CNN models with the Adam optimizer consistently outperform other configurations, achieving superior predictive accuracy and generalization capability. DenseNet models also demonstrated strong performance across all metrics. These findings emphasize the efficacy of deep learning-driven approaches in automating loan eligibility prediction, offering a scalable solution to enhance credit risk assessment and operational efficiency in financial services.**

Keywords*— *Loan eligibility prediction, deep learning, convolutional neural networks, DenseNet, model optimization, financial risk management, credit scoring.

I. INTRODUCTION

Deep learning has emerged as a transformative branch of artificial intelligence, characterized by its ability to model complex, non-linear relationships across large datasets. Its impact across domains such as computer vision, natural language processing, and financial analytics has been profound, with significant advancements observed in predictive modeling tasks. In the domain of financial services, particularly in loan eligibility prediction, traditional evaluation methods based on manual underwriting and rule-based systems are increasingly inadequate. These conventional approaches are not only labor-intensive and time-consuming but are also susceptible to human biases and inconsistencies, leading to potential financial risks and inefficiencies.

To address these challenges, deep learning models offer a data-driven alternative capable of automatically learning hierarchical feature representations from applicant data. The loan eligibility prediction pipeline begins with the acquisition and preprocessing of applicant information, which includes standardizing input features, handling missing values, encoding categorical variables, and normalizing numerical attributes to enhance model convergence and performance. After preprocessing, the data is systematically categorized into eligible and non-eligible classes, enabling supervised learning models to discern intricate patterns and decision boundaries.

Among the wide array of deep learning architectures, Deep Neural Networks (DNNs), Convolutional Neural Networks (CNNs), and Dense Convolutional Networks (DenseNet) have demonstrated remarkable efficacy. DNNs are well-suited for structured, tabular data common in financial datasets, effectively capturing complex interactions among input variables through multiple hidden layers. CNNs, while originally designed for image recognition tasks, have been adapted to financial data modeling due to their ability to capture local dependencies and hierarchical patterns through convolutional operations. DenseNet architecture further refines feature propagation by introducing direct connections between all layers, mitigating the vanishing gradient problem and promoting feature reuse, thereby enhancing both model robustness and efficiency.

Furthermore, model optimization plays a critical role in deep learning performance. Optimization algorithms such as Adaptive Moment Estimation (Adam), Stochastic Gradient Descent (SGD), Root Mean Square Propagation (RMSprop), and Adaptive Gradient Algorithm (Adagrad) are employed to minimize loss functions and accelerate convergence during training. Each optimizer offers unique advantages in terms of learning rate adaptation, computational efficiency, and generalization capabilities, impacting the final model's predictive performance.

This study systematically evaluates and compares the performance of DNN, CNN, and DenseNet models across

different optimizers on a real-world loan dataset. The models are assessed using comprehensive evaluation metrics, including accuracy, precision, recall, F1-score, and Area Under the Receiver Operating Characteristic Curve (ROC-AUC), to provide a holistic view of model effectiveness. By conducting an in-depth comparative analysis, this research aims to identify the most suitable deep learning framework for automated, scalable, and accurate loan eligibility prediction, ultimately contributing to more efficient credit risk assessment and decision-making processes within financial institutions.

II. LITERATURE SURVEY

This section reviews prominent studies in the domain of loan eligibility prediction, emphasizing various machine learning and deep learning techniques aimed at enhancing prediction accuracy and efficiency in financial decision-making.

Nazim Uddin *et al.* [1] proposed an ensemble machine learning framework combining Logistic Regression, Random Forest, Extra Trees, DNNs, RNNs, and Long Short-Term Memory (LSTM) models. Their voting classifier, enhanced with SMOTE (Synthetic Minority Over-sampling Technique) for data balancing, achieved an accuracy of 87.26%, outperforming individual models.

F. M. Ahosanul Haque *et al.* [2] evaluated the performance of Decision Trees, AdaBoost, Random Forest, Support Vector Machine (SVM), and Gaussian Naive Bayes algorithms for loan eligibility prediction. Using a dataset comprising 148,670 instances and 37 features, the AdaBoost classifier demonstrated superior performance with 99.99% accuracy, illustrating the efficacy of ensemble learning methods in financial prediction tasks.

Miraz Al Mamun *et al.* [3] investigated Random Forest, XGBoost, AdaBoost, LightGBM, Decision Trees, and K-Nearest Neighbors for predicting loan eligibility. Logistic Regression exhibited the best performance, achieving 92% accuracy and a 96% F1-score. The study emphasized the automation of feature validation to enhance the efficiency of the loan approval process.

Md. Khabir Uddin Ahamed *et al.* [4] applied ensemble techniques by integrating Random Forest, AdaBoost, and Logistic Regression models into a voting-based classifier. This method improved the stability and reliability of predictions, addressing data variability challenges in real-world banking applications.

Owusu *et al.* [5] tackled the issue of loan default prediction by employing Adaptive Synthetic (ADASYN) oversampling combined with DNN. Their model achieved an accuracy of 94.1%, surpassing traditional approaches and providing an effective tool for early loan default detection.

Mamun *et al.* [6] explored the application of Random Forest, XGBoost, AdaBoost, LightGBM, Decision Trees, and K-Nearest Neighbor models on a Kaggle dataset for loan eligibility prediction. LightGBM outperformed the other models, achieving 91.89% accuracy and the highest AUC score of 75%, demonstrating the potential of automated machine learning techniques in streamlining loan approval processes.

Collectively, these studies highlight the critical role of ensemble learning, oversampling techniques, and automated validation in advancing loan eligibility and default prediction systems, offering practical and scalable solutions for the financial sector.

III. METHODOLOGY

This study employs a structured approach to predict loan eligibility using deep learning models optimized through various algorithms. The methodology comprises several key phases, including data acquisition and preprocessing, automated feature extraction through model learning, development of different network architectures, optimizer integration for performance enhancement, and comprehensive model evaluation. Each phase was meticulously designed to maximize prediction accuracy, minimize overfitting, and ensure the robustness of the final models. A comparative analysis across different models and optimizers was conducted to determine the most effective configuration for real-world financial applications.

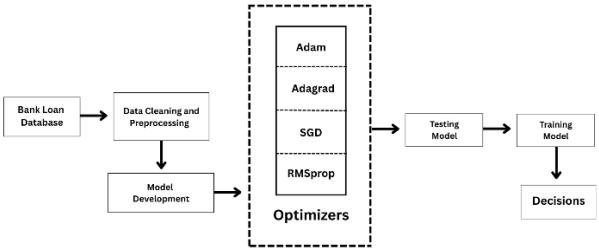

Fig. 1. Loan eligibility prediction model using deep learning models and different optimizers.

Data Acquisition and Preprocessing

The dataset employed in this study was obtained from a publicly available financial repository containing loan application records. It comprises 93,682 instances with 19 distinct features, including applicant income, loan amount, employment type, credit history, property location, and other relevant financial parameters.

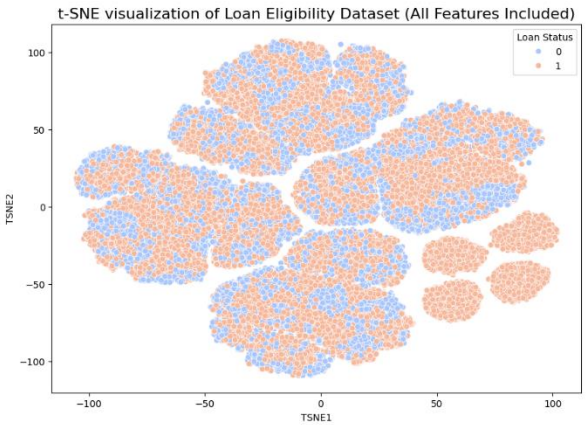

Fig. 2. Visualization of the loan dataset using t-SNE, demonstrating the separation of approved and non-approved instances through dimensionality reduction of feature representations.

Several preprocessing steps were applied to ensure data quality and readiness for deep learning models:

Handling Missing Values:

Missing entries were treated using appropriate imputation strategies, either by substituting with the mean/mode values or through predictive modeling, depending on the attribute type.

Encoding Categorical Variables:

Categorical features were transformed using label encoding for binary categories and one-hot encoding for multiclass categories, enabling the neural networks to process them effectively.

Feature Scaling:

Numerical features were normalized using Min-Max scaling, mapping all values between 0 and 1, thereby improving the convergence rate during model training.

Following the preprocessing steps, the dataset was partitioned into training and testing sets, with the split ratio varying based on the model architecture. For the DNN model, an 80:20 train-test split was employed, whereas for the CNN and DenseNet models, a 70:30 split was utilized. This stratified approach ensured an adequate amount of data for both model training and independent evaluation, promoting reliable and unbiased performance assessment across all architectures.

Feature Extraction

In this research, manual feature extraction techniques were deliberately avoided. Instead, the deep learning models inherently performed feature extraction during the training phase. Architectures such as DNN, CNN, and DenseNet automatically learned hierarchical feature representations, capturing complex patterns and relationships within the input data. The DNN model abstracted high-level interactions between features like applicant income, credit history, and loan amount through its multiple fully connected layers. In the CNN model, convolutional layers effectively identified local dependencies among input attributes, which were preprocessed into a format suitable for convolution operations, while MaxPooling layers enhanced feature summarization and provided translation invariance. DenseNet architecture, through its densely connected layers, allowed every layer to utilize feature maps from all previous layers, promoting efficient feature propagation and encouraging feature reuse, thus enabling the extraction of robust and informative patterns from financial data. By relying on this automatic feature learning approach, the models successfully captured latent variables and complex nonlinear associations that traditional manual feature engineering techniques might overlook.

Model Development

In this study, three deep learning architectures—DNN, CNN, and DenseNet—were developed to predict loan eligibility based on tabular data features. Each model was designed with varying architectures and training parameters to evaluate their effectiveness under different optimization strategies.

1) Deep Neural Network (DNN)

The DNN model consisted of multiple fully connected dense layers with ReLU activation functions, interspersed with dropout layers to prevent overfitting, and a final output layer employing Sigmoid activation for binary classification. The model was trained using a batch size of 32 over 50 epochs, with an 80:20 train-test split. Experiments were conducted using four different optimizers: Adam, SGD, RMSprop, and Adagrad.

Table 1: *DNN Model Architecture*

Layer Type	Output Size / Units	Activation
Input Layer	19 features	—
Dense Layer	128 units	ReLU
Dropout Layer	128 units	—
Dense Layer	64 units	ReLU
Dropout Layer	64 units	—
Dense Layer	32 units	ReLU
Output Layer	1 unit	Sigmoid

2) Convolutional Neural Network (CNN)

The CNN model was adapted for structured tabular data by reshaping the input features to a one-dimensional format suitable for convolutional operations. Convolutional layers captured local feature patterns, followed by max-pooling layers for down sampling and fully connected dense layers for classification. The model was trained with a batch size of 64 over 50 epochs, employing a 70:30 train-test split. Adam, SGD, RMSprop, and Adagrad optimizers were separately tested to evaluate performance differences.

Table 2: *CNN Model Architecture*

Layer Type	Output Size / Filters	Activation
Input Layer	(19, 1)	—
Conv1D Layer	32 filters, kernel size = 3	ReLU
Conv1D Layer	64 filters, kernel size = 3	ReLU
MaxPooling1D Layer	Pool size = 2	—
Flatten Layer	1D vector	—
Dense Layer	128 units	ReLU
Dense Layer	64 units	ReLU
Output Layer	1 unit	Sigmoid

3) DenseNet

The DenseNet model was customized for tabular data by implementing densely connected convolutional blocks, where each layer received inputs from all preceding layers, promoting feature reuse and alleviating vanishing gradient issues. The architecture included dense blocks, transition layers, and global average pooling before classification through a Sigmoid-activated output layer. DenseNet was trained using a batch size of 128 over 50 epochs, with a 70:30 train-test split. As with the other models, the performance was assessed individually using Adam, SGD, RMSprop, and Adagrad optimizers.

Table 3: *DenseNet Model Architecture*

Layer Type	Output Size / Filters	Activation
Input Layer	(19, 1)	—
Dense Block Layer 1	32 filters, kernel size = 3	ReLU
Dense Block Layer 2	32 filters, kernel size = 3	ReLU
Transition Layer 1	Pooling + 1x1 Convolution	ReLU
Dense Block Layer 3	64 filters, kernel size = 3	ReLU
Dense Block Layer 4	64 filters, kernel size = 3	ReLU
Global Average Pooling	1D vector	—
Output Layer	1 unit	Sigmoid

Optimizer Integration and Loss Function

The optimization process plays a critical role in the training of deep learning models by minimizing the loss function and updating model weights to reduce prediction error. In this study, each of the proposed deep learning architectures—DNN, CNN, and DenseNet—was trained using four well-established optimizers: Adam, Stochastic Gradient Descent (SGD), RMSprop, and Adagrad. These optimizers vary in how they compute and apply gradients, affecting model convergence, stability, and generalization.

1) Stochastic Gradient Descent (SGD)

SGD is the foundational optimization algorithm in machine learning and deep learning. It updates model parameters by computing the gradient of the loss function with respect to each parameter on a mini-batch of training data. The update rule is:

$$\theta_{t+1} = \theta_t - \eta \cdot \nabla_\theta L(\theta_t)$$

Where:

- θ_t are the model parameters at iteration t
- η is the learning rate
- $\nabla\theta L$ is the gradient of the loss function

While simple, SGD can suffer from slow convergence and sensitivity to learning rate settings.

2) RMSprop

RMSprop improves upon SGD by adapting the learning rate for each parameter individually. It maintains a moving average of the squared gradients and scales the learning rate inversely to the root of this average:

$$v_t = \gamma v_{t-1} + (1-\gamma) \cdot (\nabla_\theta L(\theta_t))^2$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{v_t + \varepsilon}} \cdot \nabla_\theta L(\theta_t)$$

Where:

- Vt is the moving average of squared gradients
- γ is the decay rate (typically 0.9)
- ϵ is a small constant to prevent division by zero

RMSprop is particularly effective in non-stationary settings and for recurrent neural networks.

3) Adagrad

Adagrad is an adaptive optimizer that adjusts the learning rate based on the historical accumulation of gradients. This method benefits features that occur infrequently by giving them higher learning rates:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{G_t + \varepsilon}} \cdot \nabla_\theta L(\theta_t)$$

$$G_t = G_{t-1} + (\nabla_\theta L(\theta_t))^2$$

Where Gt is the sum of squared gradients up to time t. While effective for sparse data, Adagrad's learning rate can decay too aggressively.

4) Adam (Adaptive Moment Estimation)

Adam is one of the most widely used optimizers due to its efficiency and fast convergence. It combines the ideas of momentum and RMSprop by maintaining both the first moment (mean) and second moment (uncentered variance) of the gradients:

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) \cdot (\nabla_\theta L(\theta_t))^2$$

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}, \quad \hat{v}_t = \frac{v_t}{1-\beta_2^t}$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t + \varepsilon}} \cdot \hat{m}_t$$

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) \cdot \nabla_\theta L(\theta_t)$$

Adam is especially useful for large-scale data and deep networks.

Binary Cross-Entropy Loss Function

For all models and optimizers, the loss function used was Binary Cross-Entropy (BCE), suitable for binary classification problems such as loan eligibility (approved vs not approved). BCE measures the divergence between predicted probabilities y^i and true labels yi

$$L_{BCE} = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)]$$

Where:

- yi ∈ {0,1} is the true label
- y^i ∈ [0,1] is the predicted probability
- NNN is the number of samples

This loss function penalizes confident wrong predictions more heavily and encourages probabilistic decision boundaries, making it well-suited for neural classification models.

Performance Metrics

To quantitatively evaluate the performance of the proposed deep learning models in predicting loan eligibility, several standard classification metrics were employed. These metrics assess various aspects of binary classification, providing both general accuracy and class-specific performance insights. The following metrics were computed for both training and testing datasets:

1) Accuracy

Accuracy measures the proportion of correctly classified instances over the total number of predictions:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Where:

- TP: True Positives
- TN: True Negatives
- FP: False Positives
- FN: False Negatives

2) Precision

Precision quantifies the correctness of positive predictions. It is the ratio of correctly predicted positive instances to the total predicted positives:

$$\text{Precision} = \frac{TP}{TP + FP}$$

A high precision score indicates that the model returns more relevant results and fewer false alarms, which is critical in loan approvals to avoid granting loans to ineligible applicants.

3) Recall (Sensitivity)

Recall indicates the model's ability to identify all relevant instances. It is defined as the ratio of correctly predicted positives to all actual positive instances:

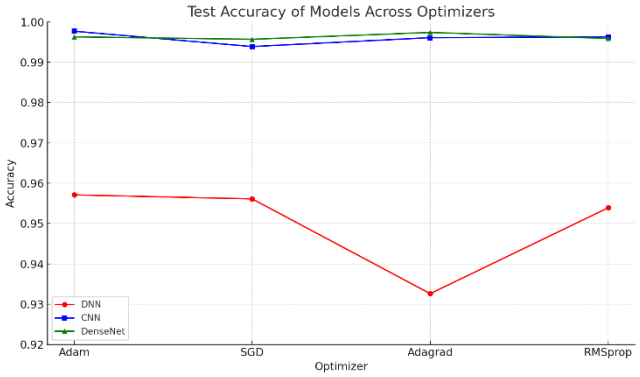

Fig 3. Test accuracy of DNN, CNN, and DenseNet models across different optimizers.

$$\text{Recall} = \frac{TP}{TP + FN}$$

In the context of loan prediction, a high recall ensures that eligible applicants are correctly identified, minimizing the rejection of valid requests.

4) F1-Score

The F1-score is the harmonic mean of precision and recall. It provides a single metric that balances both concerns, especially useful when the class distribution is imbalanced:

$$\text{F1-Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

5) Receiver Operating Characteristic – Area Under Curve (ROC-AUC)

ROC-AUC measures the model's ability to distinguish between the positive and negative classes across all classification thresholds. It is the area under the ROC curve, where a value closer to 1 indicates better performance:

$$\text{ROC-AUC} = \int_0^1 TPR(FPR^{-1}(x)dx$$

Where:

- $TPR = \frac{TP}{TP+FN}$ is the True Positive Rate
- $FPR = \frac{FP}{FP+TN}$ is the False Positive Rate

6) Confusion Matrix

The confusion matrix is a 2×2 matrix used to evaluate the performance of a binary classifier:

$$\begin{bmatrix} TN & FP \\ FN & TP \end{bmatrix}$$

This matrix offers detailed insight into the types of prediction errors made by the model, facilitating more targeted improvements.

These metrics were systematically computed for each combination of model architecture (DNN, CNN, DenseNet) and optimizer (Adam, SGD, RMSprop, Adagrad), providing a thorough basis for performance comparison and model selection.

IV. Experimental Results

This section present a comprehensive evaluation of three deep learning models (DNN, CNN, and DenseNet) applied to the task of loan eligibility prediction. Each model was trained and tested using four different optimization algorithms: Adam, SGD, RMSprop, and Adagrad. The models were evaluated using multiple classification metrics, including precision, recall, F1-score, and ROC-AUC. Test accuracy comparisons are visually illustrated in Fig. 3, while detailed metrics for each optimizer are provided in Tables 4-6.

Performance of DNN

The DNN exhibited moderate variability in performance depending on the optimizer employed. While test accuracy trends are illustrated in Figure 1, detailed metric analysis indicates that the Adam optimizer provided the highest F1-score of 0.9420 and a recall of 0.9941, making it particularly effective for identifying positive loan eligibility cases. RMSprop and SGD produced comparable F1-scores of 0.9372 and 0.9402, respectively, with stable precision and recall values. In contrast, the Adagrad optimizer, although balanced in recall (0.9732), resulted in a relatively lower F1-score of 0.9101 and ROC-AUC of 0.9652. These results suggest that Adam and RMSprop offer better generalization for the DNN architecture. A comprehensive summary of these results is presented in Table 4.

Table 4. Performance metrics of DNN with various optimizers.

Optimizer	Accuracy (Train/Test)	Precision (Train/Test)	Recall (Train/Test)	F1-Score (Train/Test)	ROC-AUC (Train/Test)
Adam	0.9587 / 0.9571	0.8969 / 0.8951	0.9977 / 0.9941	0.9446 / 0.9420	0.9781 / 0.9685
SGD	0.9561 / 0.9561	0.9004 / 0.9010	0.9846 / 0.9829	0.9406 / 0.9402	0.9753 / 0.9706
Adagrad	0.9306 / 0.9326	0.8519 / 0.8547	0.9721 / 0.9732	0.9081 / 0.9101	0.9637 / 0.9652
RMSprop	0.9547 / 0.9539	0.8997 / 0.8983	0.9808 / 0.9795	0.9385 / 0.9372	0.9704 / 0.9695

Performance of CNN

The CNN consistently outperformed the DNN across all evaluated performance metrics. With the Adam optimizer, CNN achieved an outstanding F1-score of 0.9982 and a perfect ROC-AUC of 1.0000, indicating its strong classification capability for both classes. RMSprop and Adagrad also demonstrated high robustness, maintaining F1-scores above 0.9970 and near-perfect ROC-AUC scores. Even when optimized with SGD, CNN sustained a high F1-score of 0.9952. Notably, the model's recall remained consistently high across all optimizer configurations, underscoring its effectiveness in minimizing false negative an essential trait in financial decision systems. The detailed performance metrics are provided in Table 5.

Table 5. Performance Metrics of CNN Using Various Optimizers

Optimizer	Accuracy (Train/Test)	Precision (Train/Test)	Recall (Train/Test)	F1-Score (Train/Test)	ROC-AUC (Train/Test)
Adam	0.9978 / 0.9977	0.9971 / 0.9972	0.9995 / 0.9992	0.9983 / 0.9982	1.0000 / 1.0000
SGD	0.9955 / 0.9939	0.9970 / 0.9958	0.9960 / 0.9946	0.9965 / 0.9952	0.9999 / 0.9999
Adagrad	0.9976 / 0.9961	0.9985 / 0.9971	0.9978 / 0.9969	0.9981 / 0.9970	1.0000 / 1.0000
RMSprop	0.9975 / 0.9963	0.9975 / 0.9965	0.9987 / 0.9977	0.9981 / 0.9971	1.0000 / 0.9999

Performance of DenseNet

Among the three architectures evaluated, DenseNet consistently delivered the highest and most stable results across all optimizers. The Adagrad optimizer led to the best overall performance, achieving a test accuracy of 99.74%, F1-score of 0.9980, and a perfect ROC-AUC of 1.0000. DenseNet coupled with Adam and RMSprop also performed exceptionally well, with F1-scores of 0.9971 and 0.9968, respectively. Precision and recall values remained closely balanced across all configurations, indicating minimal bias toward either class. These results confirm DenseNet's superior generalization ability and its suitability for deployment in high-stakes classification scenarios. A full breakdown of these results is shown in Table 6.

Table 6. Performance Metrics of DenseNet Using Various Optimizers

Optimizer	Accuracy (Train/Test)	Precision (Train/Test)	Recall (Train/Test)	F1-Score (Train/Test)	ROC-AUC (Train/Test)
Adam	0.9977 / 0.9963	0.9979 / 0.9968	0.9984 / 0.9975	0.9982 / 0.9971	1.0000 / 1.0000
SGD	0.9962 / 0.9957	0.9949 / 0.9945	0.9993 / 0.9988	0.9971 / 0.9966	1.0000 / 1.0000
Adagrad	0.9984 / 0.9974	0.9987 / 0.9978	0.9989 / 0.9982	0.9988 / 0.9980	1.0000 / 1.0000
RMSprop	0.9965 / 0.9959	0.9964 / 0.9961	0.9982 / 0.9975	0.9973 / 0.9968	1.0000 / 0.9999

Comparative Analysis and Observations

The comparative performance analysis across DNN, CNN, and DenseNet models with different optimizers reveals distinct trends in classification capability for loan eligibility prediction. As illustrated in Fig. 3, CNN and DenseNet consistently outperform the DNN model across all optimizers in terms of test accuracy. Among the CNN configurations, the Adam optimizer yields the highest performance with a test accuracy of 99.77% and an F1-score of 0.9982. Similarly, the DenseNet model achieves its peak performance using the Adagrad optimizer, reaching a test accuracy of 99.74% and an F1-score of 0.9980. In contrast, the DNN model, while showing relatively lower overall performance, achieves its best results with the Adam optimizer, obtaining a test accuracy of 95.71% and an F1-score of 0.9420.

Notably, the impact of optimizer choice is more pronounced in the DNN architecture, where a substantial performance drop is observed with Adagrad (93.26% test accuracy) compared to Adam. Conversely, CNN and DenseNet exhibit minimal variance across optimizers, indicating their robustness to optimization algorithms. The results suggest that deeper and more complex architectures like CNN and DenseNet not only generalize better but also benefit more uniformly from different optimization strategies. This comprehensive comparison highlights the importance of architectural selection and optimizer tuning in achieving

optimal predictive performance for financial classification tasks.

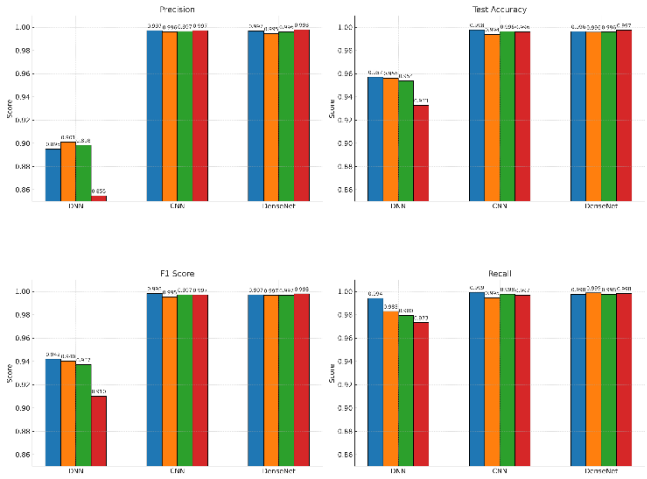

Fig 4. Performance comparison of DNN, CNN, and DenseNet using four optimizers. (a) F1 Score, (b) Recall, (c) Precision, (d) Test Accuracy. Metrics are based on testing data.

V. CONCLUSION

This research presents a comprehensive evaluation of three deep learning architectures—DNN, CNN, and DenseNet—applied to the task of loan eligibility prediction, with a comparative analysis across four optimizers: Adam, SGD, RMSprop, and Adagrad. Each model-optimizer pair was rigorously evaluated using multiple performance metrics, including accuracy, precision, recall, F1-score, ROC-AUC, and confusion matrices, on both training and testing datasets.

The experimental results reveal that CNN and DenseNet architectures significantly outperform DNN, achieving higher predictive accuracy and robustness across all optimizers. Among optimizers, Adam consistently delivered strong and balanced performance, while Adagrad showed relatively lower precision, suggesting a trade-off between recall and false positive rates. Notably, CNN with Adam achieved the highest test accuracy of 0.9977, accompanied by near-perfect precision and recall, indicating its superior generalization capability. DenseNet with Adagrad also demonstrated exceptional performance, achieving a testing accuracy of 0.9974 with outstanding F1 and ROC-AUC scores.

Overall, the findings confirm that the choice of optimizer significantly impacts model performance and that CNN and DenseNet, particularly when coupled with Adam, are optimal for reliable loan eligibility prediction. Future work could explore ensemble strategies or real-time deployment scenarios, as well as incorporating explainability frameworks to enhance trust and transparency in automated financial decision systems.

ACKNOWLEDGMENT

We sincerely thank electronics and communication department and management of BVRIT Narsapur for providing high NVIDIA – 3090Ti 24 GB GPU machine for our research work.

REFERENCES

[1] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” *arXiv preprint arXiv:1412.6980*, 2014. [Online]. Available: https://arxiv.org/abs/1412.6980

[2] S. Ruder, “An Overview of Gradient Descent Optimization Algorithms,” *arXiv preprint arXiv:1609.04747*, 2016. [Online]. Available: https://arxiv.org/abs/1609.04747

[3] D. Choi, K. Kim, and J. Moon, “On Empirical Comparisons of Optimizers for Deep Learning,” *arXiv preprint arXiv:2002.11887*, 2020. [Online]. Available: https://arxiv.org/abs/2002.11887

[4] T. Dozat, “Incorporating Nesterov Momentum into Adam,” *ICLR Workshop*, 2016. [Online]. Available: https://openreview.net/pdf?id=OM0jvwB8jIp57ZJjtNEZ

[5] M. D. Zeiler, “ADADELTA: An Adaptive Learning Rate Method,” *arXiv preprint arXiv:1212.5701*, 2012. [Online]. Available: https://arxiv.org/abs/1212.5701

[6] A. Shrestha and A. Mahmood, “Review of Deep Learning Algorithms and Architectures,” *IEEE Access*, vol. 7, pp. 53040–53065, 2019. [Online]. Available: https://ieeexplore.ieee.org/document/8765800

[7] J. Brownlee, “A Tour of Machine Learning Algorithms,” *Machine Learning Mastery*, 2016. [Online]. Available: https://machinelearningmastery.com/a-tour-of-machine-learning-algorithms/

[8] S. Vani and T. V. M. Rao, “Performance Assessment of Various Optimizers on CNN,” in *Proc. Int. Conf. Trends in Electronics and Informatics (ICOEI)*, 2019, pp. 1065–1070. [Online]. Available: https://ieeexplore.ieee.org/document/8862686

[9] D. Zou, Y. Cao, D. Zhou, and Q. Gu, “SGD Optimizes Over-parameterized Deep ReLU Networks,” *arXiv preprint arXiv:1811.08888*, 2018. [Online]. Available: https://arxiv.org/abs/1811.08888

[10] S. S. Du, X. Zhai, B. Poczos, and A. Singh, “Gradient Descent Finds Global Minima of Deep Neural Networks,” *arXiv preprint arXiv:1811.03804*, 2018. [Online]. Available: https://arxiv.org/abs/1811.03804

[11] J. Brownlee, “How to Use Learning Curves to Diagnose Machine Learning Model Performance,” *Machine Learning Mastery*, 2018. [Online]. Available: https://machinelearningmastery.com/learning-curves-for-diagnosing-machine-learning-model-performance/

[12] A. C. Wilson, R. Roelofs, M. Stern, N. Srebro, and B. Recht, “The Marginal Value of Adaptive Gradient Methods in Machine Learning,” *arXiv preprint arXiv:1705.08292*, 2017. [Online]. Available: https://arxiv.org/abs/1705.08292