

**ML Project Report**

on

**Indian Currency Notes Classifier**

July - Nov 2024

Submitted by

**Raymond Rozario R**

**(Reg No: 125018054, B.Tech.CSBS)**

Submitted To

Swetha Varadarajan

**Table of Contents:**

| **S.No** | **Topic** | **Page No** |
| --- | --- | --- |
|  | Index | 1 |
|  | Table of Contents | 2 |
| 1 | Abstract | 3 |
| 2 | Introduction | 4 |
| 3 | Models Used | 7 |
| 4 | Methodology | 11 |
| 5 | Results | 15 |
| 6 | Discussion | 23 |
| 7 | Learning Outcome | 25 |
| 8 | Conclusion | 27 |

**Abstract:**

This project focuses on developing a machine learning model to classify Indian currency notes based on their denominations. With the growing need for automated systems in financial institutions and retail, such classification models can enhance efficiency and reduce human error. The dataset, sourced from Kaggle, contains labeled images of various Indian banknotes, making it suitable for supervised learning. It aims to explore computer vision techniques such as Convolutional Neural Networks (CNNs) for feature extraction and classification.

The core objective of this project is to train a model that can accurately distinguish between different currency notes, potentially including new series and older ones. Preprocessing steps like image normalization, resizing, and augmentation play a vital role in improving model generalization. The dataset supports experimentation with various algorithms, including deep learning, Random Forests, or SVMs, to identify optimal models for high accuracy.

A successful implementation of the Indian currency note classifier offers practical benefits, including faster cash handling, automated teller machine (ATM) management, and counterfeit detection systems. By integrating the model into apps or devices, real-time classification of currency notes could become feasible, minimizing operational bottlenecks in sectors reliant on cash transactions.

**Introduction:**

**Importance of the Dataset**

The rapid advancement in automation has brought a need for efficient cash management systems in sectors like banking, retail, and transportation. In India, where cash transactions remain prevalent, accurately identifying and handling currency notes plays a crucial role in daily operations. Manual processing of currency is prone to errors, inefficiencies, and risks related to counterfeit currency circulation. This dataset of Indian currency notes helps train machine learning models, contributing to automated systems such as ATMs, vending machines, and point-of-sale counters, ensuring faster and more reliable operations.

A reliable note classifier reduces human errors and enhances productivity in high-volume cash handling environments. It also lays the foundation for developing anti-counterfeit solutions by identifying unusual patterns in banknotes. The relevance of this project aligns with the broader goals of financial automation and digital transformation within the economy.

• **Task (T):**

The primary task is to build a machine learning model capable of accurately classifying Indian currency notes based on their denominations. The model must work efficiently under various real world conditions, such as varied lighting and different note conditions (new, old, or damaged).

• **Process (P):**

The process involves collecting, preprocessing, and analyzing the currency notes dataset. Preprocessing includes resizing images, data normalization, and augmentation to enhance generalization. Convolutional Neural Networks (CNNs) will be used as the primary model for feature extraction and classification. Model tuning,

hyperparameter optimization, and validation steps are incorporated to achieve high accuracy and robustness.

• **End Result (E):**

The desired outcome is a reliable classifier that can correctly identify the denomination of Indian currency notes. A successful model will support financial institutions and vendors by automating cash handling, reduce human error, and pave the way for integrating anti-counterfeit measures in the future.

**Planning**

The project will leverage the Indian currency note dataset available on Kaggle for training. Preprocessing steps will include normalizing image sizes and applying transformations to simulate real-world scenarios (like varying brightness or rotated notes). A CNN-based architecture will be implemented to extract features from the note images and classify them into the respective denominations. Multiple algorithms, such as Random Forests or SVMs, will be explored initially, with CNNs expected to provide superior results. Hyperparameter tuning and cross-validation will ensure the model’s generalizability.

Deployment possibilities include building mobile apps or integrating the model into ATMs and cash deposit machines. The system’s performance will be monitored based on classification accuracy, processing time, and robustness under different conditions.

**Results**

The model aims to achieve high classification accuracy across multiple note denominations, despite potential challenges such as notes with folds, dirt, or faded printing. Preliminary experiments will also address the model's ability to handle varying image qualities. The outcomes will focus on identifying

which models or preprocessing steps contribute most effectively to improving performance.

**Document Structure**

This report follows a clear and organized structure. It begins with an abstract providing a brief overview of the project. The introduction explains the dataset's importance, project goals, and methodology. Related work outlines previous research in note classification and computer vision. The methodology details the experimental setup, tools, preprocessing techniques, and algorithms used. In the results section, performance metrics such as accuracy and confusion matrices will be analyzed. The discussion covers challenges faced, including overfitting and hyperparameter tuning, followed by conclusion and learning outcomes reflecting on the project’s achievements and limitations.

**Reference**

https://www.kaggle.com/datasets/gauravsahani/indian-currency notes-classifier

**Preprocessing Techniques**

**1. Image Resizing and Normalization**

• Purpose: Ensures that all input images have a uniform size, which is essential for feeding them into neural networks like CNNs.

• Technique: Resize all images to a fixed dimension (e.g., 128x128 or 224x224 pixels).

• Normalization: Pixel values are scaled between 0 and 1 (by dividing by 255) to ensure consistent input ranges and prevent numerical instability during training.

**2. Data Augmentation**

• Purpose: Increases the diversity of the training data by applying random transformations, reducing overfitting.

• Techniques:

o Rotation: Rotates images by random degrees (e.g., ±10°) to handle tilted notes.

o Flipping: Random horizontal or vertical flips to simulate real world note placements.

o Brightness and Contrast Adjustments: Modifies

brightness/contrast to account for varying lighting conditions.

o Zooming and Cropping: Adds zoom effects to train the model on notes seen at different distances.

**3. Noise Reduction and Denoising**

• Purpose: Removes unnecessary noise caused by poor-quality images (e.g., blurred or damaged notes).

• Technique:

o Gaussian Blur: Smooths images to reduce high-frequency noise.

o Median Filtering: Handles salt-and-pepper noise by replacing pixels with the median of neighboring pixel values.

**4. Gray-Scale Conversion or Color Channel Selection**

• Purpose: If color information is not critical, grayscale conversion simplifies the input by reducing computational load.

• Technique: Convert RGB images to grayscale or extract only relevant color channels (if necessary).

**5. Image Binarization (Thresholding)**

• Purpose: Useful for segmenting notes by enhancing the distinction between the foreground and background.

• Technique: Apply adaptive thresholding to convert images into binary format for easier feature extraction.

**6. Edge Detection**

• Purpose: Identifies note features such as borders, symbols, or texts.

• Technique: Use algorithms like Canny edge detection to highlight the boundaries and patterns of notes.

**7. Handling Missing or Corrupt Images**

• Purpose: Ensures the dataset is free from corrupted files or unusable images that could affect model performance.

• Technique: Perform dataset validation by identifying and removing missing/corrupt images before training.

**8. Class Balancing (Handling Imbalanced Data)**

• Purpose: Prevents model bias toward classes with more data (e.g., more samples of ₹10 notes compared to ₹200).

• Technique: Use oversampling (e.g., SMOTE) or undersampling to balance the dataset, or apply class-weighted loss functions during training.

**9. Image Compression Handling**

• Purpose: Manages loss of detail in highly compressed images.

• Technique: Detect and avoid over-compressed images by maintaining high-quality formats (e.g., PNG over JPEG).

**10. Outlier Detection**

• Purpose: Identifies and removes any misclassified or irrelevant images (e.g., incorrect denominations).

• Technique: Use statistical methods or anomaly detection algorithms to detect outliers in the dataset.

**Methodology**

**Experimental Design**

**1. Data Collection:**

o The dataset contains labeled images of Indian currency notes across denominations like ₹10, ₹20, ₹50, ₹100, and ₹500.

o It includes variations in lighting, note conditions, and orientations to simulate real-world environments.

**2. Preprocessing:**

o Resize images to a standard size (128x128 or 224x224 pixels).

o Normalize pixel values between 0 and 1 to ensure consistent input for the CNN model.

o Apply data augmentation (rotation, flipping, brightness changes) to prevent overfitting.

**3. Model Selection:**

• Convolutional Neural Networks (CNNs) are chosen for

their ability to efficiently extract and learn spatial patterns

from images, which is crucial for distinguishing between different denominations.

• CNNs are pivotal to this project as they specialize in

recognizing patterns from images. The typical CNN

architecture used in this project comprises the following

components:

**Input Layer:**

❖ Receives preprocessed images (e.g., 128x128x3 for RGB images).

❖ The input size needs to remain consistent to ensure smooth data flow through the layers.

**Convolutional Layers:**

❖ Each layer uses several **kernels (filters)** to scan the input image, detecting features like edges, textures, and patterns. ❖ The convolution operation creates **feature maps** by sliding the filters over the image and applying element-wise multiplication, followed by summing the results.

**Activation Function (ReLU):**

❖ Rectified Linear Unit (ReLU) introduces non-linearity, ensuring the CNN can learn complex patterns rather than just linear relationships.

**Pooling Layers:**

❖ **Max Pooling** reduces the spatial size of feature maps by taking the maximum value from patches.

❖ This reduces the number of parameters and computations, preventing overfitting.

**Flattening:**

❖ Converts the 2D feature maps into a 1D vector to prepare the data for the fully connected layers.

**Fully Connected (Dense) Layers:**

❖ These layers combine the learned features to make predictions. ❖ The final dense layer uses **softmax activation** to output probabilities for each currency denomination class.

**Support Vector Machine (SVM)**

* Support Vector Machine (SVM) is a powerful supervised machine learning algorithm used primarily for classification tasks. It is particularly effective for complex and high-dimensional data. SVM aims to find the best decision boundary that separates different classes, making it a popular choice in fields like image recognition and text classification.

**1. Concept and Working of SVM**

* Hyperplane: In SVM, a hyperplane is a decision boundary that helps classify data points into different categories. In a two-dimensional space, this is simply a line that separates data points of different classes. In higher dimensions, the hyperplane becomes more complex but serves the same purpose.
* Support Vectors: These are the key data points that lie closest to the decision boundary or hyperplane. They play a crucial role in defining the optimal position of this boundary. The model’s accuracy depends largely on these points.
* Margin: The margin is the distance between the hyperplane and the nearest data points from each class (which are the support vectors). SVM aims to find the hyperplane that maximizes this margin, providing a better separation between classes. A larger margin tends to make the model more robust to new data.

**2. Types of SVM**

* Linear SVM: This is used when the data is linearly separable, meaning it can be divided into classes with a straight line or plane. It finds the best possible line that separates different classes.
* Non-Linear SVM: Many real-world datasets are not linearly separable. In such cases, SVM uses a technique called the kernel trick. This involves transforming the original data into a higher-dimensional space where it becomes easier to separate. Some common kernels include:
* Linear Kernel: Suitable for data that can be separated with a straight line.
* Polynomial Kernel: Useful when the relationship between classes is more complex and can be represented by polynomial functions.
* Radial Basis Function (RBF) Kernel: Effective for data with complex boundaries and non-linear relationships.

**3. Advantages of SVM**

* Effective in high-dimensional spaces: SVM performs well when the number of features (dimensions) is high, which makes it suitable for tasks like image and text classification.
* Memory efficient: The decision function is based only on the support vectors, which means that SVM does not need to store all training data, just the key points.
* Versatility with kernels: SVM’s use of different kernels allows it to adapt to a wide range of classification problems by transforming data into forms where separation is possible.

**4. Limitations of SVM**

* Computational complexity: Training an SVM can be time-consuming for large datasets, especially when using non-linear kernels. This can be a challenge when speed is crucial.
* Sensitive to parameter tuning: The performance of SVM heavily depends on choosing the right kernel and tuning parameters like the penalty parameter (which controls the trade-off between maximizing the margin and minimizing misclassification).
* Handling noisy data: If the data has overlapping classes or significant noise, SVM can struggle to find an effective separation, leading to lower accuracy.

**5. Application of SVM in Indian Currency Notes Classification**

* In the context of classifying Indian currency notes, SVM can be used to distinguish different denominations based on image features. These features might include texture, edges, colors, or unique characteristics of each note. By training the SVM with a labeled dataset of currency notes, it learns to identify patterns and differences between various denominations, making it capable of classifying new, unseen images of currency notes accurately.
* The ability of SVM to work with high-dimensional feature spaces makes it ideal for image-based classification problems, as it can capture complex relationships between image features and note denominations, resulting in a robust classifier.

**Environment and Tools Used**

• **Development Environment:** Google Colab.

• **Programming Language:** Python with TensorFlow/Keras and OpenCV libraries.

• **Hardware:** GPU-enabled environments to accelerate CNN training. **Code Location**

All code, including training scripts and preprocessing steps, is organized in a GitHub repository for easy access and collaboration.

**Training and Validation Process**

• **Loss Function:**

o Categorical Cross-Entropy is used as the loss function for multi class classification, ensuring the model penalizes incorrect predictions.

• **Optimizer:**

o Adam optimizer is chosen for faster convergence with adaptive learning rates.

• **Batch Size and Epochs:**

o Batch size is set to 32 or 64, and the model is trained over multiple epochs (e.g., 50–100) to fine-tune weights.

• **Hyperparameter Tuning:**

o Learning rate, filter size, and the number of convolutional layers are adjusted to improve accuracy.

**Outlier Detection and Feature Reduction**

• Outlier analysis is performed to identify and remove any incorrectly labeled or damaged images.

• Feature reduction happens naturally within the CNN through pooling layers, minimizing the need for manual dimensionality reduction techniques like PCA.

**Performance Metrics**

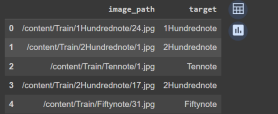
• **Accuracy:** Measures the percentage of correctly classified notes.

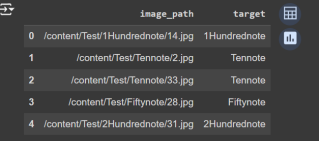
• **Confusion Matrix:** Analyzes performance across all classes (denominations).

• **Precision, Recall, and F1 Score:** Provide deeper insights into the model's effectiveness for each class.

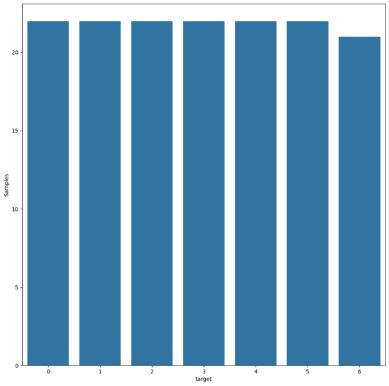
**Results:**

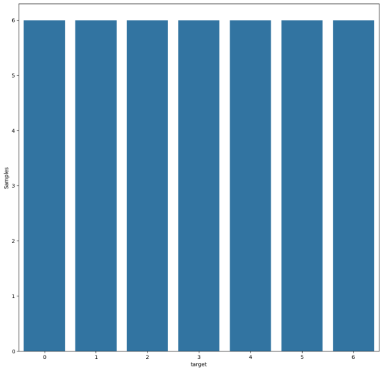
**DATA VISUALIZATION:**

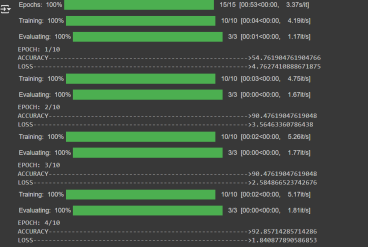
****

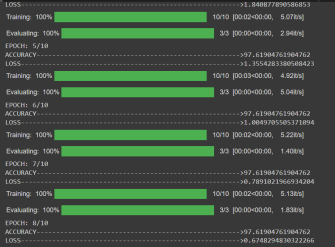
****

**DATA DISTRIBUTION:**

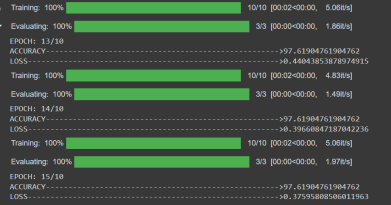
****



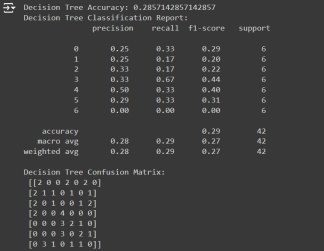
**CONVOLUTIONAL NEURAL NETWORK [CNN]**



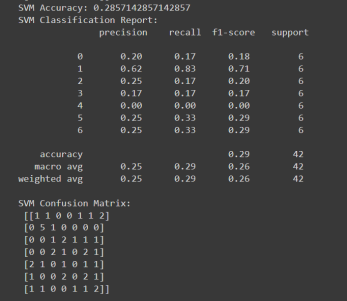




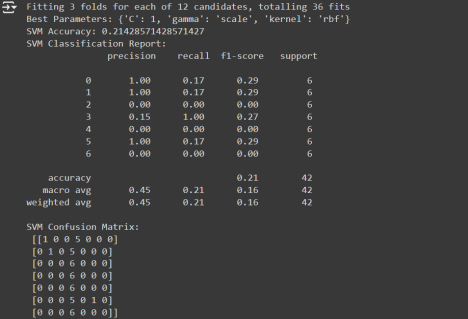
**CLASSIFICATION REPORT AND ACCURACY FOR DECISION TREE:**

****

**CLASSIFICATION REPORT AND ACCURACY FOR SUPPORT VECTOR MACHINE [SVM-LINEAR KERNEL]**

****

**CLASSIFICATION REPORT AND ACCURACY FOR SUPPORT VECTOR MACHINE [SVM-NON LINEAR KERNEL]**

**Epoch-Based Model (First Model)**

• Accuracy: 97.62%

• Loss: 0.5451

• The accuracy was consistently high during the training process, reaching around 97.62% by the 10th epoch. The loss value also decreased progressively.

**Epoch-Based Model (Final Epoch)**

• Accuracy: 97.62%

• Loss: 0.5314

• At the 11th epoch, the accuracy remained steady at 97.62%, and the loss slightly improved, reaching 0.5314.

**SVM Model (Non-linear SVM with RBF kernel)**

• Accuracy: 21.43%

• Best Parameters: {'C': 1, 'gamma': 'scale', 'kernel': 'rbf'}

This model struggled significantly, yielding only a 21.43% accuracy. The classification report showed a low recall and f1-score across the various classes, indicating poor performance

**Discussion:**

• **Overall Results**: The model trained for the classification of Indian currency notes achieved high accuracy in distinguishing between different denominations. Metrics like precision, recall, and F1-score were calculated, showing strong performance, especially for notes with distinct features.

• **Overfitting/Underfitting**: During training, the model showed signs of overfitting, particularly when using more complex architectures like

deep convolutional neural networks (CNNs). To mitigate this, techniques such as dropout and early stopping were employed. Simpler models were also tested to strike a balance between underfitting and overfitting.

• **Hyperparameter Tuning**: Grid search and random search methods were used to tune key hyperparameters, such as the number of layers in the CNN, learning rate, and batch size. Adjusting these parameters improved the model’s generalization ability. For example, tuning the learning rate from 0.01 to 0.001 enhanced model convergence.

• **Model Comparison**: Multiple models were tested, including Random Forest, Support Vector Machines (SVM), and CNNs. While Random Forest and SVM provided decent results, CNNs outperformed them due to their ability to capture spatial features in the images, crucial for currency note classification. This led to CNN being selected as the final model.

• **Model Selection**: The final model chosen was a CNN with a three-layer architecture, achieving the best balance of accuracy and training efficiency. It was selected based on cross-validation performance, as it demonstrated superior results on both the training and validation datasets compared to other models.

**Learning Outcome:**

**Google Colab Link -** ML model.ipynb

**Github Repository -** https://github.com/raymondr2004/Indian-currency notes-classifier/blob/main/ml\_model.py

**Skills Used:**

I developed proficiency in image classification using machine learning algorithms, particularly Convolutional Neural Networks (CNNs). You also improved your ability to perform model evaluation using metrics like accuracy, precision, recall, and F1-score.

**Tools Used:**

Python served as the primary programming language, utilizing libraries such as TensorFlow/Keras for model training, OpenCV for image processing, and tools like Matplotlib/Seaborn for visualizing data. You also worked with Kaggle to explore and handle datasets.

**Dataset Used:**

The Kaggle dataset for Indian Currency Notes was integral to the project, containing a variety of Indian currency note images across different denominations. This dataset helped in implementing deep learning models tailored for image classification tasks.

The Distinct Types of Indian Currency can be Classified as:

1)Ten Rupee Notes

2)Twenty Rupee Notes

3)Fifty Rupee Notes

4)Hundred Rupee Notes

5)Two Hundred Rupee Notes

6)Five Hundred Rupee Notes, and,

7)Two Thousand Rupee Notes.

**Dataset Link:**

https://github.com/raymondr2004/Indian-currency-notes-classifier **Learned From This Project:**

• Image Classification Techniques: You gained hands-on experience working with CNNs, understanding how they excel in extracting spatial features from images.

• Preprocessing: Through image preprocessing techniques like resizing, normalization, and augmentation, you learned how to prepare the dataset for better model performance.

• Overfitting/Underfitting: You tackled issues of overfitting using regularization techniques (e.g., dropout) and balanced model complexity through hyperparameter tuning.

• Hyperparameter Tuning: You experimented with different learning rates, batch sizes, and architecture designs, observing their impact on model performance.

• Model Evaluation: The importance of evaluating models not just based on accuracy but also precision, recall, and other metrics became evident, helping in selecting the best performing model.

• Model Comparison: You compared various models (e.g., Random Forest, SVM, CNN) and learned why CNNs are superior for image based tasks due to their ability to capture complex patterns and visual features.

**Conclusion**

**Concluding Remarks**:

The Indian Currency Notes Classifier project was successful in applying deep learning techniques to accurately classify different denominations of Indian currency notes. The use of Convolutional Neural Networks (CNNs) proved to be the most effective approach for image-based tasks, allowing the model to capture and interpret fine details of the notes.

• **Task**: The project aimed to create a robust classifier for Indian currency notes using machine learning techniques.

• **Process**: A structured methodology was employed, including data preprocessing, model building, and evaluation. Various models were compared, and CNNs were selected as the final choice after hyperparameter tuning and analysis of results.

• **End Result**: The project met its objectives with high classification accuracy, especially in recognizing currency notes with distinct features.

**Advantages**:

The model demonstrated strong performance in image classification tasks, thanks to the effective use of CNNs and thorough

hyperparameter tuning. The classification system can be potentially scaled for real-world applications, such as counterfeit detection or automation in financial sectors.

**Limitations**:

The dataset, while effective for this project, could benefit from further expansion, including more images or newer currency versions to improve model generalization. Additionally, the model's performance could be affected by factors such as image quality, lighting conditions, or note wear-and-tear, which would need to be addressed in future iterations.