1. Abstract
2. Introduction
3. Literature Survey
4. Proposed Work
5. **Model Building**

Model building refers to the process of creating a mathematical representation of a real-world system or problem, which can be used to make predictions, decisions, or solve problems. Once the model is built, it can be deployed and used to make predictions or solve problems in real-world scenarios.

The process of model building involves several steps

Step1: Gather Data: Identify and gather relevant data to train the model.

Step2: Pre-process Data: Resizing images, applying augmentations such as horizontal flips, zooming, rotating, shearing, shifting for the training phase.

Step3: Select Model: Choose the appropriate machine learning or deep learning model that is best suited’

Step4: Train Model: Use the prepared data to train the model by adjusting its parameters and hyperparameters. Once the model is trained, it can be used for predictions

Step5: Evaluate Model: Once the model is trained, evaluate its performance using a suitable evaluation metric such as confusion matrix, accuracy score for a test dataset.

Step6: Optimize Model: Fine-tune the model based on the evaluation results. This may involve tweaking the model's architecture, adjusting the hyperparameters, or using more or better data.

Step7: Deploy the model: Once the model is optimized, deploy it to a production environment where it can be used to make predictions or solve problems.

**CNN (with LeakyRelu)**

CNN stands for Convolutional Neural Network. In a CNN, the input is passed through a series of convolutional layers that extract relevant features from the image of currency note. These convolutional layers apply a set of filters to the input image, which detect specific features such as edges, corners, and shapes.

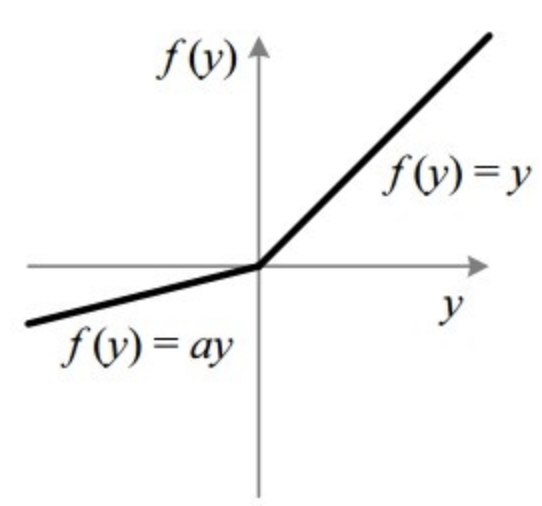
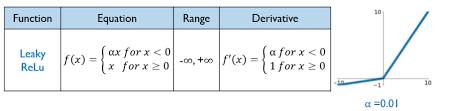
The output of the convolutional layers is then passed through a series of pooling layers that reduce the spatial size of the feature maps and extract the most important information from them. This helps to make the network more efficient by reducing the number of parameters and computational requirements.

After the convolutional and pooling layers, the output is flattened and passed through one or more fully connected layers that classify the input image into currency’s denomination.

For testing, a validation dataset is used to predict denominations and then the predictions are evaluated based on true values.

In traditional ReLU (Rectified Linear Unit), the activation function outputs 0 for any negative input and a linear function for any positive input. This can result in "dead neurons" in the network, where the neurons output 0 and do not contribute to the network's learning.

To address this issue, LeakyReLU introduces a small slope to the negative region of the activation function, which helps to prevent the "dead neurons" problem. The LeakyReLU function is defined as:



Fig[ ] : Plot of LeakyRely activation function.

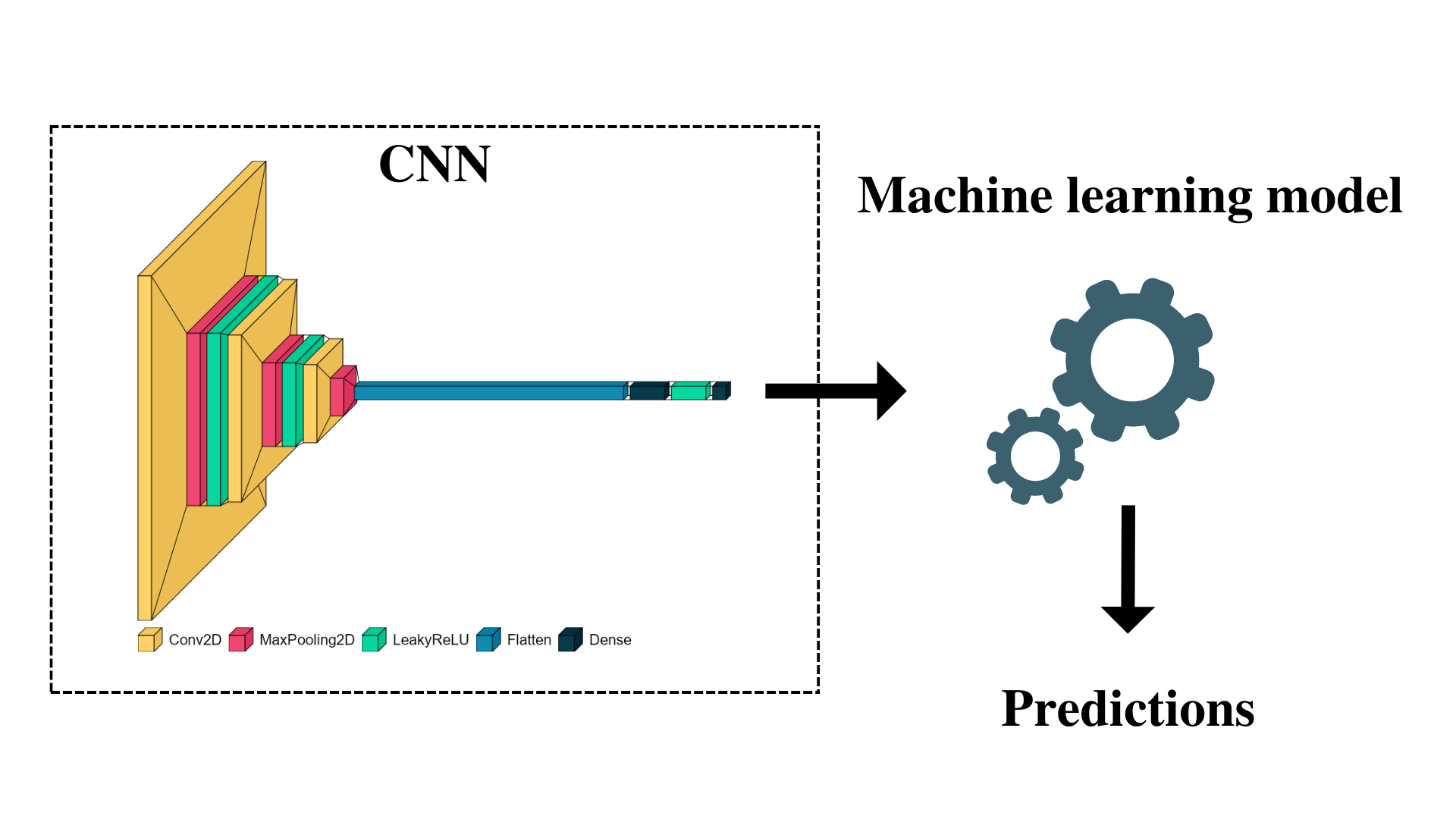
When used in a CNN, LeakyReLU can improve the network's performance by preventing overfitting and increasing the network's ability to learn from small amounts of data.

**CNN (with Machine Learning models)**

In a CNN, features are extracted through a process called feature mapping. The input image is passed through a series of convolutional layers that apply a set of filters to the image, resulting in a set of feature maps. These feature maps capture different aspects of the input image, such as edges, textures, and shapes.

After the convolutional layers, the feature maps are typically passed through one or more pooling layers that reduce the spatial size of the feature maps and extract the most important information from them. This helps to make the network more efficient by reducing the number of parameters and computational requirements.

Once the feature maps have been extracted, they can be flattened. The output of the last fully connected layer can be used as input to a machine learning model, which can then be trained to perform predictions.



Fig[ ] : Hybrid model of CNN and Machine Learning model with CNN architecture

**CNN with KNN (K-nearest neighbors)**

CNN with KNN hybrid model is a machine learning model that combines the feature extraction power of a convolutional neural network (CNN) with the classification ability of a K-nearest neighbors (KNN) algorithm.

The advantage of using a hybrid model like this is that the CNN is able to extract highly discriminative features from the input data of Indian currency images, which can improve the accuracy of the classification performed by the KNN algorithm. Additionally, the KNN algorithm is able to handle complex decision boundaries and is relatively simple to implement.

The hybrid model works as follows:

Step1: CNN is trained to extract features from the input data.

Step2: The output of the CNN is a set of feature vectors that represent the input data.

Step3: The feature vectors are then used as input to the KNN algorithm, which assigns a label to each vector based on the K-nearest neighbors in the feature space.

Step4: The label assigned to each feature vector is then used to classify the input data.

During the testing, an image is passed through CNN to extract features which are passed to KNN model for predicting the denomination.

**CNN with XGBoost**

CNN with XGBoost hybrid model is a machine learning model that combines the feature extraction power of a convolutional neural network (CNN) with the classification ability of an XGBoost model.

The CNN is used to extract features from the input data of Indian currency images and then the feature vectors are passed to the XGBoost model (which is a decision tree-based algorithm) for classification. In this approach, the CNN acts as a feature extractor, while the XGBoost model acts as a classifier.

CNNs can be slow to train and make predictions, especially for large datasets. By using XGBoost for classification, the computational cost of the CNN is reduced, making the overall model faster to train and evaluate.

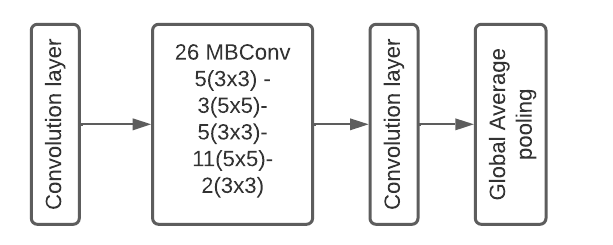
The output of the CNN is a set of feature vectors that represent the input data of Indian currency image. The feature vectors are then used as input to the XGBoost model, which assigns a label to each vector based on the learned decision rules.

**EfficientNetB3**

EfficientNetB3 is a deep convolutional neural network architecture that has shown state-of-the-art performance on a wide range of computer vision tasks, including image classification. It is an efficient and highly scalable network that can be easily trained on large datasets, making it an attractive option for applications such as currency recognition.

In the context of Indian currency classification, EfficientNetB3 can be trained to accurately classify banknotes based on their denomination. The architecture consists of multiple layers of convolutional and pooling operations, followed by several fully connected layers that perform the final classification.

One of the main advantages of EfficientNetB3 is its ability to scale efficiently to different input resolutions and network sizes. This makes it possible to train the model on a range of input image sizes, allowing it to perform well on images of different resolutions and aspect ratios.



Fig[ ] : EfficientNetB3 architecture

**ResNet**

ResNet (Residual Network) is a deep convolutional neural network architecture that was introduced by Microsoft researchers in 2015.

The working principle of ResNet is based on the idea of residual learning. In traditional deep neural networks, the output of each layer is directly connected to the input of the next layer. However, as the network becomes deeper, the gradients can become very small, making it difficult to train the network effectively.

In ResNet, residual blocks are introduced to address this problem. Each residual block consists of two or more convolutional layers and a skip connection that adds the input to the output of the block. This allows the network to learn residual functions, which can be thought of as the difference between the input and the output of the block. The residual function is then added to the input of the block, allowing the network to learn the identity function when it is optimal to do so. By using residual blocks, the gradients can propagate more easily through the network, allowing deeper networks to be trained effectively.

The idea behind transfer learning with ResNet is that the lower layers of the pre-trained ResNet model, which learn to extract low-level features such as edges and textures, can be re-used for the new task. These pre-trained layers can be frozen during the fine-tuning process, so that they are not re-trained, and only the higher-level layers of the ResNet model are trained on the new dataset.

**VGG16**

VGG16 is a deep convolutional neural network architecture that was developed by researchers at the Visual Geometry Group (VGG) at the University of Oxford. It is a widely used architecture for image classification tasks and has achieved state-of-the-art performance on several computer vision benchmarks.

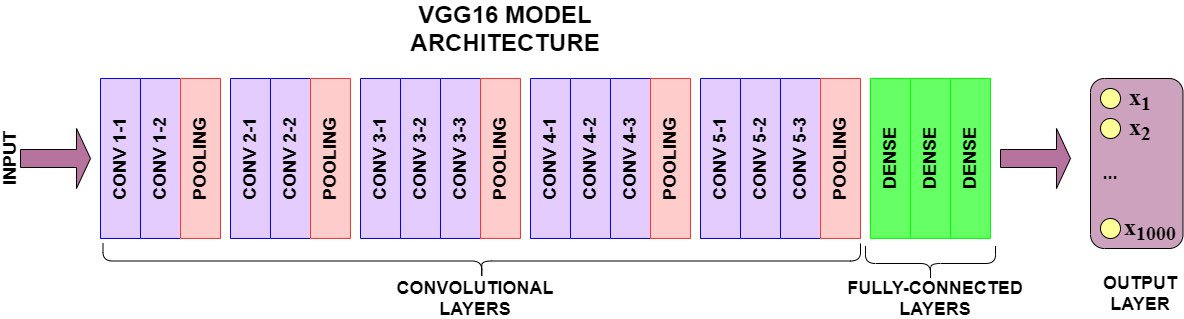
The basic idea behind VGG16 is to use a deep neural network with many convolutional layers to extract increasingly complex and abstract features from an input image. The network is composed of 16 layers, including 13 convolutional layers and 3 fully connected layers.

The input to the network is an RGB image of size 224x224 pixels. The image is first passed through a series of convolutional layers, each with a small filter size (3x3) and a stride of 1 pixel. The output of each convolutional layer is then passed through a rectified linear unit (ReLU) activation function, which introduces non-linearity into the model.

After several convolutional layers, the size of the feature maps is gradually reduced by max pooling layers, which downsample the feature maps by taking the maximum value in each pool. The output of the final pooling layer is a set of high-level features that capture increasingly abstract information about the input image.

During training, the weights of the VGG16 model are learned by minimizing a loss function, such as cross-entropy, between the predicted class probabilities and the true class labels. The weights are updated using backpropagation, which computes the gradient of the loss with respect to the model parameters.

A pre-trained VGG16 is implemented using transfer learning methodology. The VGG16 is trained using the image-net dataset. This could be used directly for classifying currency notes. By using the pre-trained VGG16 model as a feature extractor, we can leverage the powerful feature extraction capabilities of the VGG16 model and apply it to the Indian currency classification task, which can lead to improved accuracy and faster training times compared to training a new model from scratch.



Fig[ ] : VGG16 architecture

**CNN+VGG16 Ensemble Model**

The CNN+VGG16 ensemble model is a machine learning model that combines the strengths of two different models, a convolutional neural network (CNN) and the VGG16 pre-trained model, to improve the accuracy of image classification tasks.

The basic idea behind the ensemble model is to train two different models on the same dataset, one using a CNN architecture and the other using the VGG16 pre-trained model. Each model is trained on the same set of input images, but with different architectures and weights. The outputs of the two models are then combined, usually through averaging or weighted averaging, to produce the final classification output.

The advantage of the CNN+VGG16 ensemble model is to improve the accuracy of image classification tasks. The CNN model is good at capturing low-level features in images, while the VGG16 pre-trained model is good at capturing high-level features. By combining these two models, we can improve the overall accuracy of the classification task.

1. Results and Discussion

Both VGG16 and ResNet have been shown to be very effective for image classification tasks. However, VGG16 performed better than ResNet for classification of Indian currency which might be because of following reasons:

1. The dataset used is small,as VGG16 has fewer parameters than ResNet, it is easier to train in case of VGG16.
2. VGG16 is better at learning simple features such as lines, edges, and shapes.
3. VGG16 does not have skip connections like ResNet, which can sometimes make it easier to interpret and debug the model. In cases where interpretability is important.

ResNet has been shown to outperform VGG16 on larger and more complex datasets, but VGG16 may perform better in certain specific cases as described above.

1. Conclusion
2. References