# Chapter 1

# Introduction

## 1.1 Deep Learning

The "deep learning" group of machine learning algorithms employs many layers to gradually recover higher-level properties from the input's original data. Lower layers in image processing, for example, identify boundaries, but higher layers distinguish items that are meaningful to humans, such as figures, letters, or faces. Deep learning is used in many modern systems, including computer vision and automatic voice recognition. The performance of a number of large-vocabulary speech recognition tasks, as well as commonly used evaluation sets such as image classification, has steadily increased. In computer vision, convolutional neural networks (CNNs) are more effective. Applying deep learning to invertible difficulties like deconvolution, super-resolution, image restoration, and film colorization has demonstrated effectiveness. Applications like "Shrinkage Fields for Effective Image Restoration," which trains on an image data-set, and "Deep Image Prior," which trains on the image that needs restoration, are examples of learning methods used in these applications.

**Relation among:**

- Artificial Intelligence.
- Deep Learning.
- Machine Learning.
- CNN(Convolutional Neural Network).

Figure 1.1 below depicts AI's connection with deep learning, machine learning, and CNN.

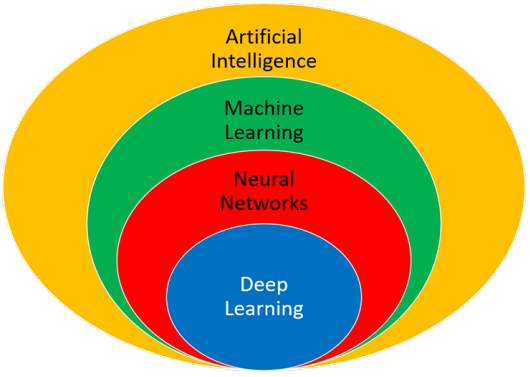


FIGURE 1.1: Deep Learning

**Artificial Intelligence:**

Artificial intelligence (AI) encompasses the emulation of human intelligence within computers and machines, entailing the creation of algorithms and systems capable of executing functions like problem-solving, learning, decision-making, and comprehending natural language. The objective of AI is to fabricate machines that can replicate human cognitive abilities, empowering them to manage intricate tasks and respond to novel circumstances.

**Machine Learning:**

Machine learning constitutes a subset of artificial intelligence, wherein computer systems autonomously acquire knowledge from data and enhance their capabilities without the need for explicit programming. This field relies on algorithms and statistical models to empower computers to make predictions, classify data, and uncover underlying patterns. Machine learning finds extensive application across various domains such as image recognition, natural language processing, recommendation systems, and more, enabling automation of decision-making and the extraction of valuable insights from data.

**Convolutional Neural Networks(CNN):**

A Convolutional Neural Network (CNN) is a specialized form of artificial neural network designed for processing visual data like images and videos. It has significantly influenced areas like computer vision, image recognition, and natural language processing. CNNs are used in various tasks such as image classification, object detection, facial recognition, medical imaging, and autonomous vehicle development due to their feature extraction capabilities in machine learning and computer vision.

## 1.2 Working

Obtaining the best solution necessitates an initial comprehensive grasp of the problem in question. Equally pivotal is the evaluation of whether deep learning is the apt method for tackling it. The ensuing step encompasses pinpointing the relevant data needed for the particular problem. Following this, the third step entails the selection of the fitting deep learning algorithm. In the fourth stage, the dataset is trained using the designated algorithm. Concluding the process, the fifth step involves a thorough final testing of the dataset. The operational flow of Deep Learning is visually represented in Figure 1.2.

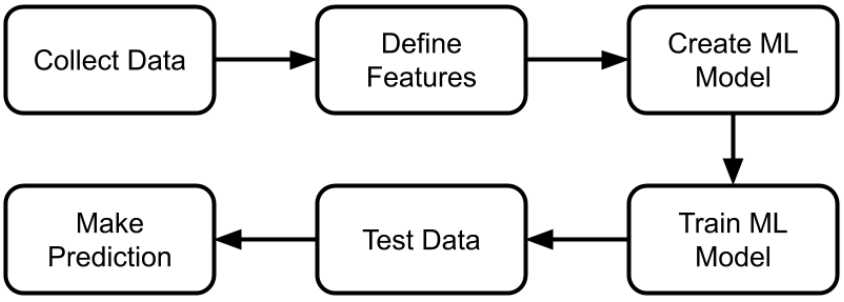


FIGURE 1.2: Working Flow of Deep Learning

**Tool Used:**

- Anaconda.
- Jupyter.

## 1.3 History

Convolutional neural networks, often abbreviated as ConvNets or CNNs, have garnered substantial global recognition since 2012, despite not being entirely novel technologies. Notably, the earliest known commercial use of CNNs dates back to 1998 with the introduction of LeNet-5. Conceived by LeCun and colleagues after extensive research in CNNs, it was initially applied to classify text images, particularly those with dimensions of 32x32 pixels. Since that important milestone, CNNs have solidified their dominance in the field of machine learning, outperforming other methodologies.

CNNs were widely embraced by researchers who recognized their potential, especially when dealing with larger image datasets. The catalyst for the remarkable rise of CNNs is often attributed to the 2012 ImageNet Large Scale Visual Recognition Challenge. Following this pivotal event, CNNs swiftly became a standard tool in various domains, including image processing and more. They have effectively tackled numerous challenges, encompassing tasks such as visual recognition, real-time video monitoring, automated segmentation techniques, facial recognition, text-to-handwriting conversion, natural language processing, and automated translation.

## 1.4 Techniques

### 1.4.1 Convolutional Neural Network

Convolutional Neural Networks (CNNs) replicate the specialized connectivity pattern found in the visual cortex. In the visual cortex, neurons are concentrated in distinct regions and exhibit selective responses to specific visual features. This idea draws inspiration from the observation of neurons becoming active in reaction to specific edge orientations. For instance, some neurons show high activity when presented with vertical edges, while others react to horizontal or diagonal edges. CNNs recreate this biological structure within artificial neural networks, equipping them with robust capabilities for recognizing images and extracting features.

During the forward pass, data input generates layer-by-layer outputs. The model's performance is evaluated using error functions such as cross-entropy or mean squared loss. Then, derivatives are calculated, initiating a backpropagation process designed to minimize the overall loss, thereby enhancing the network's learning and accuracy. This iterative approach continually improves the model's performance. The architecture of a CNN is visually represented in Figure 1.3.

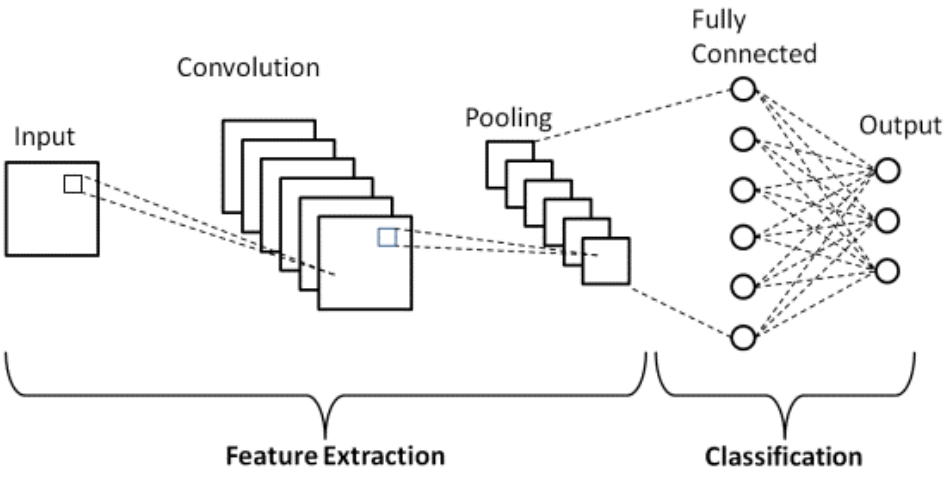


FIGURE 1.3: Architecture of CNN

### 1.4.2 Layers of CNN

- Input Layer.
- Convolutional Layer.
- Relu Layer.

- Pooling Layer.
- Fully connected Layer.
- Softmax Layer.
- Output Layer.

**Input Layer:**

The starting element within a CNN is the Input Layer, which acts as the entry point for receiving unprocessed image data. It's important to note that every pixel in the input image corresponds to an input node within this layer. The dimensions and dimensions of the Input Layer depend on the characteristics of the input image, including attributes like its width, height, and the number of color channels. Essentially, the Input Layer serves as a bridge that transforms the image into a format that the network can efficiently process.

**Convolutional Layer:**

A fundamental component within the CNN architecture is the convolution layer, responsible for extracting features. It combines both linear and nonlinear processes, involving convolution and activation functions. Convolution entails applying a small set of numbers, known as a kernel, to the input data, referred to as a tensor. This process generates a feature map by performing a pointwise multiplication between the kernel and the data vector at different locations. Multiple kernels are employed to create several feature maps, each emphasizing distinct input characteristics, effectively functioning as separate feature detectors. The size and quantity of these kernels are crucial hyperparameters that dictate the convolution operation, typically being 3x3 but occasionally 5x5 or 7x7. These parameters also govern the depth of the resulting feature maps, which can vary. This layer described in figure 1.4.

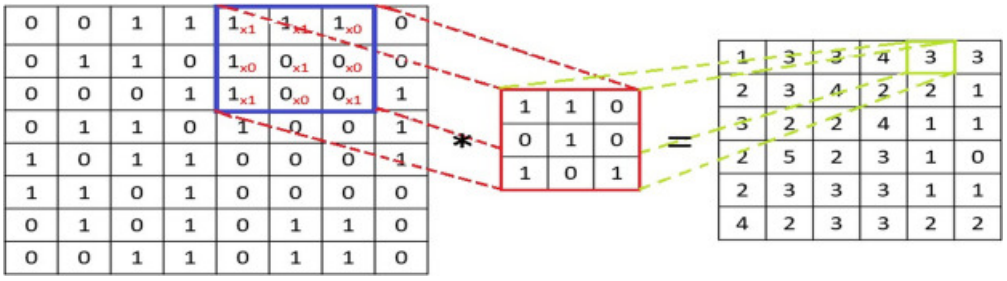

FIGURE 1.4: Convolution Layer

**Relu Layer:**

To effectively train deep neural networks using stochastic gradient descent and defect-correction training algorithms, it is necessary to employ an activation function that exhibits linear-like behavior while actually introducing nonlinearity to facilitate the learning of complex connections in the data. Furthermore, this function should be resilient in handling a substantial amount of information without causing undue fatigue. An example of an activation function that fulfills these criteria is the Rectified Linear Unit, commonly abbreviated as ReLU. Networks that utilize the rectifier function in their hidden layers are referred to as rectified networks. This layer described in figure 1.5.

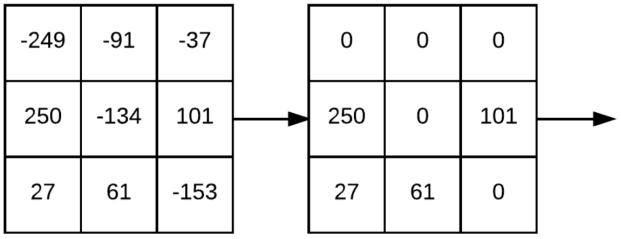


FIGURE 1.5: Relu Layer

**Pooling Layer:**

The Pooling Layer typically follows a Convolutional Layer and serves to reduce the size of the convolved feature map, reducing computational costs by reducing the number of connections and processing each feature map individually. Pooling methods, categorized by their techniques, condense information from the convolution layer. Max Pooling selects the highest value in a segment, Average Pooling calculates the segment's average, and Sum Pooling computes the sum of segment elements. This layer often connects the Convolutional Layer with the Fully Connected (FC) Layer, enabling the network to discern features independently while reducing computational load. This layer described in figure 1.6.

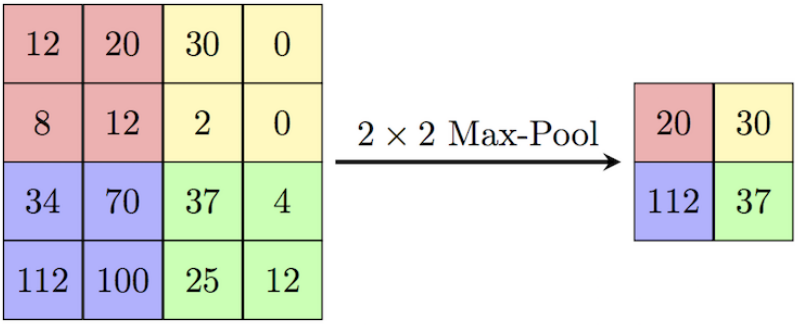


FIGURE 1.6: Pooling Layer

**Fully Connected Layer:**

The Fully Connected Layer functions in a manner similar to conventional neural network layers. In these layers, each neuron is linked to every neuron in both the preceding and succeeding layers. Typically, these layers are positioned near the end of the network, often for tasks like classification or regression. They take flattened feature maps as input, where the spatial information has been transformed into a one-dimensional vector. The operation of the Fully Connected Layer in a CNN model is depicted in Figure 1.7.

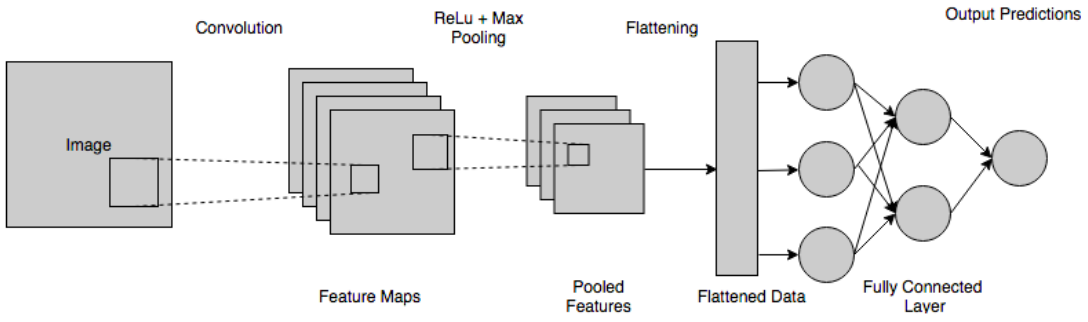


FIGURE 1.7: Fully Connected Layer

**Softmax Layer:**

The Fully Connected Layer functions in a manner akin to conventional neural network layers. Within these layers, every neuron establishes connections with each neuron in the layers that come before and follow it. Typically, these layers find application towards the conclusion of the network, particularly for tasks such as classification or regression. They take as input flattened feature maps, wherein the spatial information has been transformed into a one-dimensional vector.

**Output Layer:**

The Output Layer serves as the ultimate phase in the CNN, carrying out the crucial task of generating the network's predictions or classifications. The quantity of neurons in this layer is tailored to the specific task being addressed. For instance, binary classification might employ a single neuron with a sigmoid activation, whereas multi-class classification could involve multiple neurons with Softmax activation functions.

## 1.5 Motivation

The project, "Enhanced Counterfeit Currency Detection through Multi-Fusion Techniques," addresses the pressing need for robust and accurate counterfeit currency detection systems in today's financial landscape. Traditional methods face challenges when dealing with intricate currency patterns, varying lighting conditions, and complex backgrounds. By harnessing the capabilities of Convolutional Neural Networks (CNNs) and multi-fusion techniques for feature extraction and analysis, the project aims to improve accuracy, speed, and adaptability in counterfeit currency detection. This initiative carries practical significance as it contributes to strengthening security in financial transactions, streamlining currency sorting processes, and enhancing counterfeit detection. These advancements align with the current financial sector's requirements for swift and reliable currency recognition.

## 1.6 Existing System

Current currency recognition systems predominantly depend on traditional techniques, which encompass a mixture of rule-based algorithms and manually crafted features. These systems have played a fundamental role in various financial applications such as Automated Teller Machines (ATMs), currency sorting devices, and fraud detection systems. Nevertheless, they manifest various constraints and difficulties, thus prompting the requirement for more sophisticated and versatile methodologies.

**Rule Based Algorithms:**

Numerous conventional currency recognition systems function by adhering to predetermined regulations and guidelines. These regulations are usually established using human-defined characteristics and practical rules. For instance, a system might depend on certain patterns, colors, or dimensions to distinguish various banknote denominations. While these rule-based methods may be effective in well-regulated settings, they frequently encounter difficulties in real-world situations where currencies can be torn, soiled, or partially concealed.

**Limited Adaptability:**

Conventional systems struggle to accommodate alterations in currency designs or the introduction of new denominations. When a new banknote is introduced or an existing one is altered, these systems necessitate manual modifications to their regulations and attributes. This procedure can be both time-consuming and expensive.

**Difficulty with Complex Features:**

Currency notes incorporate elaborate security elements such as holograms, watermarks, and microprinting. Identifying and authenticating these intricate features can pose difficulties, potentially leading to incorrect positive or negative identifications when employing rule-based techniques.

**Lighting and Background Variations:**

Fluctuations in lighting environments and backgrounds can have a notable influence on the efficacy of standard systems. Shadows, reflections, and alterations in lighting can result in inaccuracies in the recognition of currency.

**Inefficient Handling of Variations:**

Banknotes can display variances as a result of wear and tear, folding, or creasing. Standard systems may encounter challenges in managing these discrepancies, which can result in misclassifications or rejections during the currency processing.

Enhanced Counterfeit Currency Detection through Multi-Fusion Techniques marks a departure from these customary systems. It aims to overcome the constraints of existing systems and make precise predictions regarding counterfeit currency notes.

**Drawbacks:**

- Reliance on Rule-Based Methods: Many traditional systems depend on rule-based algorithms specifically crafted to identify counterfeit currency using predefined criteria. These rules often center around distinct visual attributes and patterns, rendering them susceptible to counterfeiters who can replicate these characteristics with increasing sophistication.
- Limited Flexibility: Traditional systems may encounter difficulties when adapting to novel counterfeit techniques and the changing designs of counterfeit currency. They often necessitate manual updates and adjustments, which can be time-consuming and costly.
- Challenges with Advanced Counterfeits: Contemporary counterfeit currency can be remarkably convincing, incorporating advanced printing techniques and security features that closely resemble genuine banknotes. Traditional systems may falter in detecting such counterfeit notes, leading to erroneous approvals.
- Insufficient Texture and Pattern Analysis: Traditional systems frequently lack the capability for in-depth texture and pattern analysis, which is critical for identifying counterfeit attributes that might not be readily apparent to the human eye.

## 1.7 Proposed System

The suggested system, known as Enhanced Counterfeit Currency Detection through Multi-Fusion Techniques, signifies a substantial leap forward in currency recognition technology. It tackles the constraints of current systems and seeks to offer a more precise and versatile solution. This system harnesses state-of-the-art methods and technologies to improve the retrieval and categorization of currency images, aligning with the requirements of contemporary financial environments.

**Multi-fusion techniques:**

Multi-fusion techniques combine data from various sensors, algorithms, and models, creating a more comprehensive and reliable assessment of currency authenticity. By integrating multiple data sources and analysis approaches, it becomes possible to enhance the detection of counterfeit banknotes while reducing the chances of false positives or negatives. These techniques allow the system to adapt to different counterfeit strategies and evolving designs, ultimately providing a more effective and adaptable solution for counterfeit currency detection.

**Positives**

- Improved Accuracy: Multi-fusion techniques combine various data sources and methods, enhancing the accuracy of counterfeit detection.
- Reduced False Alarms: By integrating multiple assessment approaches, the system can minimize the occurrence of false positives and negatives.
- Adaptability: These techniques enable the system to adapt to new counterfeit strategies and changing currency designs.
- Robustness: Multi-fusion methods provide a more robust and reliable solution for counterfeit currency detection.
- Comprehensive Assessment: They allow for a comprehensive assessment of currency authenticity, considering various aspects and features.

# Chapter 2

# LITERATURE REVIEW

## 2.1 Multi Fusion Techniques

Multi-fusion techniques in the context of fake currency detection refer to the integration of multiple methods or sources of information to enhance the accuracy and reliability of the detection process. When detecting counterfeit currency, it's crucial to consider various aspects and features of banknotes that counterfeiters may attempt to replicate. Multi-fusion techniques combine these aspects to create a more robust and comprehensive approach to counterfeit detection.

Fake currency detection using multi-fusion techniques like RESNET-50 and VGG16 utilizes deep learning models to enhance accuracy and reliability. These models excel at extracting detailed features from banknotes, like patterns, watermarks, and security elements, aiding in the discrimination of genuine currency from counterfeits. By combining RESNET-50 and VGG16, this multi-fusion method integrates their strengths, creating a more robust system for detecting fake currency and improving overall effectiveness.

## 2.2 ResNet-50

RESNET-50 is a convolutional neural network (CNN) architecture that is part of the RESNET (Residual Networks) family of models. It was developed by Kaiming He and his colleagues in 2015 and has been widely used in various computer vision tasks, including image classification and object detection. The "50" in RESNET-50 indicates that it consists of 50 layers, making it a relatively deep neural network.

The key innovation in RESNET is the use of residual blocks, which enable the training of very deep networks while mitigating the vanishing gradient problem. In a residual block, the input to a layer is combined with the output of that layer, allowing for the network to learn residual functions. This skip-connection or shortcut helps to propagate gradients effectively, making it easier to train deep networks.

**Evolution of ResNet-50:**

RESNET-50, part of the RESNET CNN family, was introduced in 2015 with the innovative concept of residual blocks, addressing gradient vanishing to enable deep network training. It's one of many RESNET variants, with models of increasing depth for improved performance. These models are widely used for pre-training in computer vision, serving as feature extractors or for transfer learning. Ongoing research focuses on making them more efficient through techniques like network pruning and customization for specific applications, highlighting their pivotal role in computer vision's continuous evolution.

**Applications and Performance:**

RESNET-50 excels in image classification, object detection, and image segmentation with top-tier accuracy. Its pre-trained model is a popular choice for transfer learning, while it also makes significant contributions to medical imaging, scene understanding, and efficiency-optimized variants for resource-constrained environments. Its adaptability, accuracy, and robustness have solidified its position as a cornerstone in computer vision, driving advances in healthcare, autonomous systems, and content analysis.

**Challenges, Future Directions, and Ethical Considerations:**

The use of RESNET-50 and similar deep learning models in computer vision presents challenges in computational resources, overfitting, and model interpretability. Future directions include efficient architecture development, improved transfer learning, and interdisciplinary applications. Ethical considerations encompass bias mitigation, privacy protection, accountability, and addressing job displacement, highlighting the need for responsible and equitable AI development as these models continue to shape AI and computer vision's future.

## 2.3 VGG16

VGG16 is a widely used convolutional neural network (CNN) model developed by the Visual Geometry Group at the University of Oxford. It features 16 layers, primarily composed of 3x3 convolutional filters, making it effective at capturing fine-grained image features. VGG16 is commonly employed in image classification and computer vision tasks and is often used as a pre-trained model for transfer learning or feature extraction.

**Evolution of VGG16:**

VGG16 is a well-established convolutional neural network (CNN) model, and its architecture has remained relatively consistent since its inception. There hasn't been a significant evolution or version updates for VGG16 like there have been for some other models like ResNet, which introduced deeper variants (e.g., ResNet-50, ResNet-101, etc.). VGG16's architecture, with 16 weight layers, has continued to be used in its original form for various computer vision tasks. However, researchers and practitioners have explored modifications and extensions to adapt the VGG architecture to specific applications, but the core VGG16 model itself has not significantly evolved beyond its original design.

**Applications and Performance:**

VGG16, a widely used convolutional neural network model, excels in image classification, transfer learning, and object detection. Its versatility extends to medical imaging, content-based image retrieval, art recognition, and more. While computationally intensive, it offers competitive results in various domains. Researchers continue to explore its applications and refine its efficiency, making it a valuable asset in computer vision tasks.

**Challenges, Future Directions, and Ethical Considerations:**

The use of VGG16 and similar deep learning models in computer vision presents challenges in terms of computational resources, model size, and overfitting, while future directions include efficiency improvements, customization for specific tasks, and interdisciplinary applications. Ethical considerations involve addressing bias, privacy protection, accountability, and the potential impact on employment, highlighting the need for responsible and fair AI development and deployment.

## 2.4 Tasks

**Data Collection:**

The first step in any machine learning or image processing project is the creation of a dataset, with a particular emphasis on selecting relevant photos. This dataset can encompass images of various categories, such as faces, objects, or textures, that pertain to your project. To ensure optimal performance of your model, it's essential to curate a diverse and high-quality dataset.

**Gray Scale Image Conversion:**

Converting color photos to grayscale is often beneficial for simplifying data and reducing complexity. In grayscale images, pixel intensity is represented using shades of gray. Techniques for conversion include averaging color channels and employing specific algorithms.

**Edge Detection Image Conversion:**

Edge detection proves valuable for emphasizing image outlines and object contours, though it is not mandatory. This is achieved through techniques such as employing the Sobel operator or applying the Canny edge detector.

**Feature Extraction:**

Identifying and extracting vital components from images is of paramount importance. These elements encompass attributes such as patterns, colors, textures, and shapes. Methods like shape detection, texture analysis, and histogram examination are employed to capture significant attributes, such as local binary patterns.

**Model Testing and Training:**

After preprocessing and feature extraction, the next step is to develop and fine-tune a machine learning or deep learning model. The dataset is divided into training and testing sets; the former is used for model training, and the latter is for evaluating its performance. Various techniques, like Convolutional Neural Networks (CNNs) for image-related tasks, can be employed. The model is trained to make predictions based on the extracted features, and the testing phase assesses the model's accuracy and other metrics using fresh data.

# Chapter 3

# METHODS

## 3.1 Average Method

Grayscale images convey pixel intensity using a single value, ranging from 0 for black to 255 for white, whereas color images employ three channels: red, green, and blue (RGB). The "Average method" determines grayscale by taking the average of the RGB values (Grayscale = (R + G + B) / 3) for each pixel. This method is a speedy way to convert an image to grayscale but may not preserve intricate details, particularly in colorful photos. For maintaining a higher level of color accuracy, it's advisable to explore more advanced techniques. However, if your primary concern is swift grayscale conversion without meticulous color preservation, the "Average" method is a suitable choice.

$$\text{Gray scale value} = \frac{\text{RED} + \text{BLUE} + \text{GREEN}}{3}$$

**Example-1:**

$$\begin{aligned} \text{Red Pixel value} &= 50 \\ \text{Blue Pixel value} &= 30 \\ \text{Green Pixel value} &= 70 \\ \text{Gray scale value} &= \frac{50+30+70}{3} = \frac{150}{3} = 50 \\ \text{GRAY SCALE VALUE} &= 50 \end{aligned}$$

**Example-2:**
Original Image Matrix

| | | |
|---|---|---|
| (R1, G1, B1) | (R2, G2, B2) | (R3, G3, B3) |
| (R4, G4, B4) | (R5, G5, B5) | (R6, G6, B6) |
| (R7, G7, B7) | (R8, G8, B8) | (R9, G9, B9) |

Gray Scale Image Matrix

| | | |
|---|---|---|
| Avg1 | Avg2 | Avg3 |
| Avg4 | Avg5 | Avg6 |
| Avg7 | Avg8 | Avg9 |

Before applying the grayscale conversion method, the image is shown in Figure 3.1. After applying the gray scale conversion method to Figure 3.1, it becomes Figure 3.2.


FIGURE 3.1: Original Currency Image


FIGURE 3.2: Gray scale Image

## 3.2 Canny Edge Detection Method

The Canny edge detector is a widely employed technique in image processing, known for its precision in locating well-defined edges, even in noisy or low-contrast images. It finds applications in diverse fields such as fake currency detection, object recognition, and image segmentation. Compared to simpler methods like the Sobel or Prewitt operators, the Canny edge detector offers superior edge localization and noise suppression, making it indispensable for situations requiring precise edge detection.

Canny edge detection offers several advantages. It effectively identifies edges in images, providing precise boundary information. Additionally, it reduces noise, enhancing edge detection accuracy. The algorithm is versatile, allowing users to fine-tune its parameters for specific applications. Canny edge detection is a widely used technique in computer vision and image processing due to its robust performance and adaptability. The Canny edge detection algorithm involves several steps, which are described below.

**1. Gaussian Smoothing:**

In the Canny edge detection algorithm, the initial step is Gaussian Smoothing, achieved by convolving the image with a Gaussian filter. This process reduces noise and yields a smoother image by assigning weighted values to neighboring pixels based on their distance from the central pixel. Gaussian smoothing is essential for noise reduction and improves the overall effectiveness of edge detection.

**Gaussian Fileter:**

| | | |
|---|---|---|
| 1 | 2 | 1 |
| 2 | 4 | 2 |
| 1 | 2 | 1 |

**2. Gradient Calculation:**

The Canny edge detection process involves computing the gradient of the smoothed image, which is essential for assessing edge strength and direction. This is achieved by calculating the gradient magnitude and direction for each pixel, typically employing derivatives in both the x and y directions, often implemented using Sobel operators.

**Example: Step 1:** 6 X 6 Matrix

| | | | | | |
|---|---|---|---|---|---|
| 150 | 150 | 150 | 255 | 255 | 100 |
| 150 | 150 | 255 | 255 | 100 | 100 |
| 150 | 255 | 255 | 100 | 167 | 143 |
| 255 | 255 | 1 | 145 | 1 | 157 |
| 255 | 123 | 186 | 174 | 114 | 223 |
| 200 | 189 | 147 | 289 | 120 | 176 |

**Step 2:** X Directioanl Kernal

| | | |
|---|---|---|
| -1 | 0 | 1 |
| -2 | 0 | 2 |
| -1 | 0 | 1 |

Y Directioanl Kernal

| | | |
|---|---|---|
| -1 | -2 | -1 |
| 0 | 0 | 0 |
| 1 | 2 | 1 |

**Step 3:** Apply X -Directional kernel on 6X6 Matrix.

| | | |
|---|---|---|
| 150 * (-1) | 150 * 0 | 150 * 1 |
| 150 * (-2) | 150 * 0 | 255 * 2 |
| 150 * (-1) | 255 * 0 | 255 * 1 |

| | | |
|---|---|---|
| -150 | 0 | 150 |
| -300 | 0 | 510 |
| -150 | 0 | 255 |

If we add all the values of the above matrix, we will obtain the sum.

$$G(x) = 315$$

**Step 4:** imilarly, just as in step 3, we can apply the Y-directional kernel to the 6x6 matrix, resulting in the value G(Y).

$$G(Y) = 315$$

**Step 5:** By using Euclidean distance formula

$$\begin{aligned} \text{Gradient Magnitude (G)} &= \sqrt{(G(x))^2 + (G(y))^2} \\ \text{Magnitude (G)} &= \sqrt{G(x)^2 + G(y)^2} \\ G &= \sqrt{315^2 + 315^2} \\ \text{Magnitude (G)} &= 445 \end{aligned}$$

Pixels with the highest magnitude(G) are likely part of an edges

**Step 6:** Compute gradient direction: The gradient direction is determined using the arctangent function, which provides the angle or orientation of the edges at each pixel.

$$\begin{gathered} \text{Gradient direction} = \arctan\left(\frac{G(Y)}{G(X)}\right) \\ \text{Gradient direction} = \arctan\left(\frac{445}{445}\right) \\ \text{Gradient direction} = 45^\circ \end{gathered}$$

resulting in a gradient direction of 45 degrees.

**3. Non-Maximum Suppression:** In the Non-Maximum Suppression step of edge detection, after calculating gradient magnitudes and directions for each pixel, the process examines each pixel's gradient direction and its neighboring pixels in that direction. It then compares the current pixel's magnitude with its neighbors, preserving only local maxima. This step thins out detected edges, keeping potential edges as local maxima.

**4. Double Thresholding:** In double thresholding, remaining edges are categorized as strong, weak, or non-edges. Two thresholds, a high and a low threshold, are defined. If the gradient magnitude surpasses the high threshold, the pixel is labeled as a strong edge. If the magnitude falls between the low and high thresholds, it's categorized as a weak edge. Pixels with magnitudes below the low threshold are classified as non-edges and discarded.

**5. Edge Tracking by Hysteresis:** In edge tracking, weak edges that likely belong to the same object or region are linked. The algorithm initiates from strong edges and recursively follows connected weak edges. If a weak edge is linked to a strong edge, it becomes a strong edge as well. This process persists until no more weak edges remain connected to the strong ones.

**6. Final Edge Map:** The algorithm concludes by generating a binary edge map, using white pixels to denote strong edges and black pixels for non-edges. This edge map effectively emphasizes the image's object and region boundaries.
After applying the Canny edge detection method to the currency note, Figure 3.3 is obtained.


FIGURE 3.3: Canny Edge Detection Image

## 3.3 VGG16 Model

VGG-16, developed by Karen Simonyan and Andrew Zisserman from the University of Oxford in their 2014 publication "VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION," marked a significant milestone in computer vision. This convolutional neural network (CNN) excelled in the 2014 ImageNet Large Scale Visual Recognition Challenge (ILSVRC), securing first and second place in object localization and picture classification. Achieving an impressive 92.7% top-5 test accuracy on ImageNet, which contains 14 million images grouped into 1000 classes, VGG-16 analyzes RGB images at a fixed size of 224x224 to produce a 1000-value vector, demonstrating its pivotal role in advancing the field.

VGG16 achieves an impressive 92.7% top-5 test accuracy on the ImageNet dataset, which contains 14 million photos categorized into 1000 classes. The input consists of 224x224 pixel images with RGB channels, resulting in a 224x224x3 tensor. VGG16's architecture features two consecutive 3x3 convolution layers with 64 filters, maintaining spatial resolution with 1-pixel padding and stride. After this, spatial max pooling is applied using a 2x2 pixel window and a 2-pixel stride, reducing activation sizes to 112x112x64. VGG16 stands out as the top-performing model on ImageNet, highlighting its effectiveness in image recognition tasks.

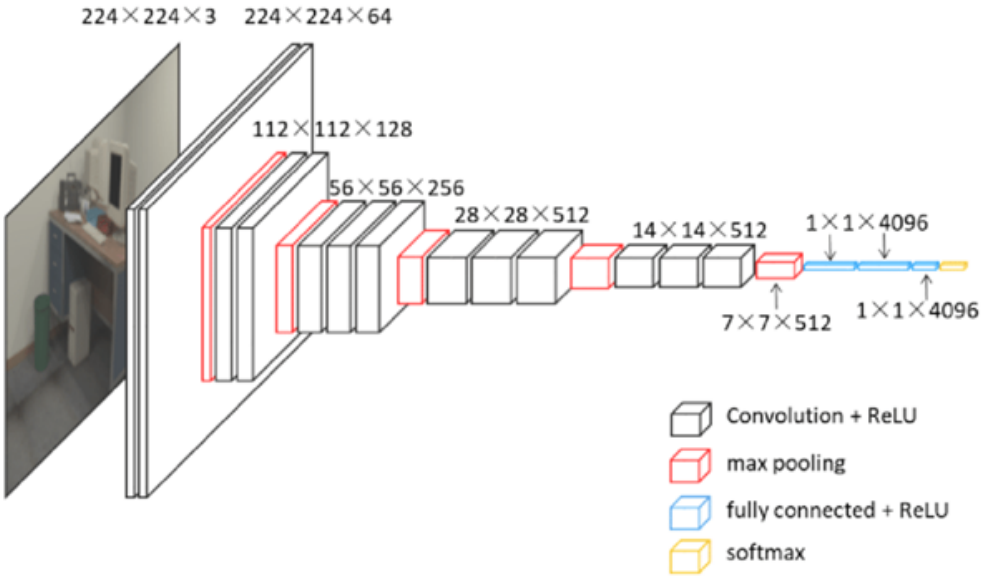


FIGURE 3.4: Sequential Model

## 3.4 Layers of VGG16

VGGNet-16 is notable for its architecture featuring 16 convolutional layers with consistent 3x3 convolutions, similar to AlexNet. It requires two to three weeks for training on four GPUs, making it a preferred choice for feature extraction in the computer vision community. The weight configuration for VGGNet is openly accessible and serves as a standard feature extractor in various applications and challenges. However, managing VGGNet's 138 million parameters can be challenging. It is commonly used with transfer learning, where parameters are fine-tuned to enhance accuracy, and the pretrained model's parameter values can be leveraged for subsequent tasks.

**16 Layers of VGG16:**

1. Convolution (64 filters)
2. Convolution (64 filters) + Max Pooling
3. Convolution (128 filters)
4. Convolution (128 filters) + Max Pooling
5. Convolution (256 filters)
6. Convolution (256 filters)
7. Convolution (256 filters) + Max Pooling
8. Convolution (512 filters)
9. Convolution (512 filters)
10. Convolution (512 filters) + Max Pooling
11. Convolution (512 filters)
12. Convolution (512 filters)
13. Convolution (512 filters) + Max Pooling
14. Fully Connected (4096 nodes)
15. Fully Connected (4096 nodes)
16. Output Layer (Softmax Activation, 1000 nodes)

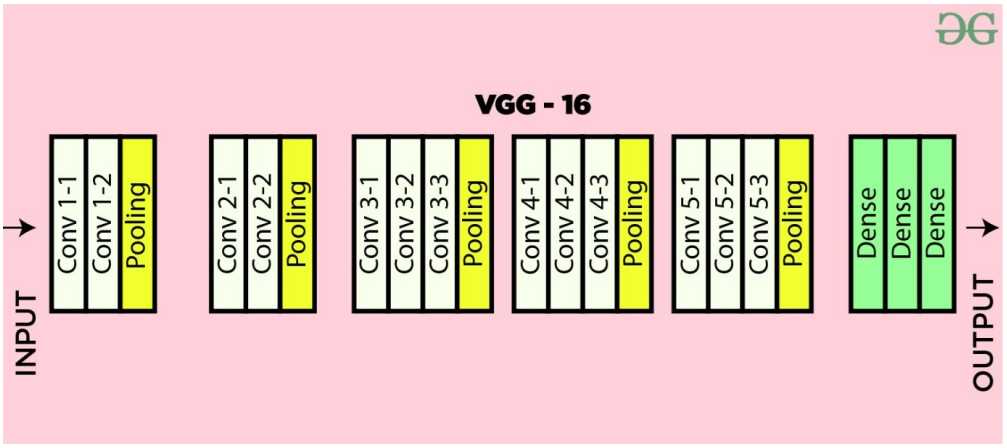


FIGURE 3.5: VGG16 Layers

## 3.5 Complexity and Challenges of VGG16

The VGG16 CNN model, while highly effective, presents several challenges. Its deep architecture with 16 layers demands significant computational power for both training and inference, making it less suitable for resource-constrained environments. Additionally, VGG16 requires substantial memory capacity due to its numerous parameters, which can be problematic for hardware with limited memory, particularly in real-time applications. Overfitting is a common risk with deep networks like VGG16, necessitating the use of techniques like dropout or regularization to ensure better generalization. Training VGG16 from scratch is time-consuming, often taking weeks, which may not be feasible for time-sensitive projects.

Transfer learning is a popular approach with VGG16, but it involves fine-tuning and domain-specific data, which can be complex and time-consuming. Furthermore, the model's size, with 138 million parameters, can be challenging to deploy on storage-limited devices. Lastly, VGG16's use of 3x3 convolutional filters exclusively limits its receptive field, potentially affecting its ability to capture very large or fine-grained features compared to models with different filter sizes.

while VGG16 has significantly contributed to computer vision, it's vital to consider its complexities and challenges when using it in practical applications, accounting for computational resources, memory, overfitting, training time, transfer learning, model size, and receptive field limitations.

## 3.6 RESNET 50

ResNet-50, a deep residual convolutional neural network (CNN) introduced by Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun in 2015, has emerged as one of the most popular and successful CNN architectures. It has proven its mettle by achieving state-of-the-art results in various computer vision tasks, spanning image classification, object detection, and semantic segmentation. ResNet-50 is characterized by its 50 convolutional layers, organized into five blocks, with the hallmark feature being its incorporation of residual blocks. These residual blocks enable the network to learn intricate representations of input data by adding the output of each block to the input, preserving crucial information from earlier layers, a key element in training deep neural networks.

ResNet-50's architecture entails an input layer that takes in a 3-channel image of size 224x224 pixels. The convolutional layers are structured into four blocks, with each block comprising three convolutional layers, followed by batch normalization and rectified linear unit (ReLU) activation. The residual blocks within each block consist of two convolutional layers, with the first having a stride of 1 and the second a stride of 2, effectively downsampling feature maps by a factor of 2. These residual blocks are enriched with shortcut connections, allowing complex data representations to be learned by adding the output of each block to the input. The final block culminates with a single convolutional layer followed by an average pooling layer, which reduces the feature map size to 1x1 pixels. The output layer comprises a fully connected layer with 1000 neurons, used for classifying the input image into one of 1000 classes.

ResNet-50's prowess stems from training on the extensive ImageNet dataset, containing over 14 million images labeled across 1000 categories. This training equips the network to recognize a wide spectrum of objects, spanning animals, plants, vehicles, and everyday items. The versatility of ResNet-50 is showcased in its ability to tackle a range of computer vision tasks, including image classification, object detection, and image segmentation. Furthermore, it frequently serves as a feature extractor for diverse machine learning applications, from natural language processing to speech recognition.

The model boasts several advantages, such as its accuracy in delivering state-of-the-art performance across computer vision tasks, efficiency in being trainable and deployable on various devices, and its versatility in handling diverse vision challenges. However, its complexity and large size can pose challenges in terms of training, interpretation, and resource requirements.

In the real world, ResNet-50 is put to practical use in image classification, object detection, and semantic segmentation tasks across applications like social media, search engines, e-commerce, self-driving cars, security systems, robotics, medical imaging, satellite imagery, and autonomous driving. It serves as a robust and efficient solution for researchers and practitioners seeking high-performing CNN models.

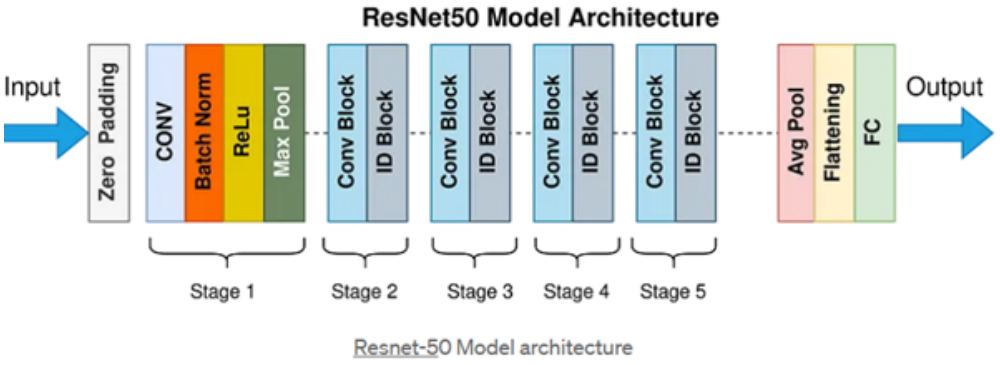


FIGURE 3.6: ResNet 50 Architecture

## 3.7 Complexity and Challenges of ResNet50

**Complex Architecture:** ResNet50 stands as a deep neural network, boasting a staggering 50 layers in its architecture. While this depth enhances its capabilities, managing such complexity during training and inference demands a firm grasp of deep learning principles and techniques, making it a challenge for those without a solid understanding of these concepts.

**Training Resources:** Training ResNet50 from the ground up can be resource-intensive. It necessitates a substantial amount of labeled data and significant computational power. This resource demand can render ResNet50 less accessible for smaller-scale projects or organizations with limited resources, impacting its adoption in certain contexts.

**Overfitting Risk:** Much like many other deep models, ResNet50 is vulnerable to overfitting. This implies that the model may perform exceptionally well on the training data but falter on unseen data, a phenomenon that is common with complex models. To address this issue, techniques such as dropout and regularization are often required, adding another layer of complexity to the training process.

**Model Size:** ResNet50's size is relatively substantial, characterized by a multitude of parameters. This largeness can result in storage and memory challenges during both training and deployment. These challenges become particularly pronounced when deploying the model on devices with limited resources, underscoring the need for careful resource management.

**Complexity in Fine-Tuning:** Complexity in Fine-Tuning: Fine-tuning a pre-trained ResNet50 for specific tasks can be a complex undertaking. It necessitates expertise in adjusting model parameters and collecting domain-specific data, making it a potentially time-consuming process that demands specialized knowledge and resources.

## 3.8 Training

The sequential model's training procedure is crucial to the "Fake Currency Detection using Multi-Fusion Techniques" project, which uses ResNet-15 and VGG-16. In order for neural networks to learn from data and generate precise predictions, this is a crucial step. The training process includes a number of important components. A labeled dataset is first split into the training set and the validation set, two separate subsets. The former is used to train the model, and the latter is employed to assess the model's performance as it is being trained.

The goal of training is to minimize a preset loss or cost function by iteratively adjusting the model's parameters, which include weights and biases. The difference between the model's predictions and the target values found in the training data is quantified by this function. Optimization methods like gradient descent are used to maximize these parameters and minimize the loss. The project's Figure 3.8 provides a visual depiction of this critical stage by showing model's training procedure.

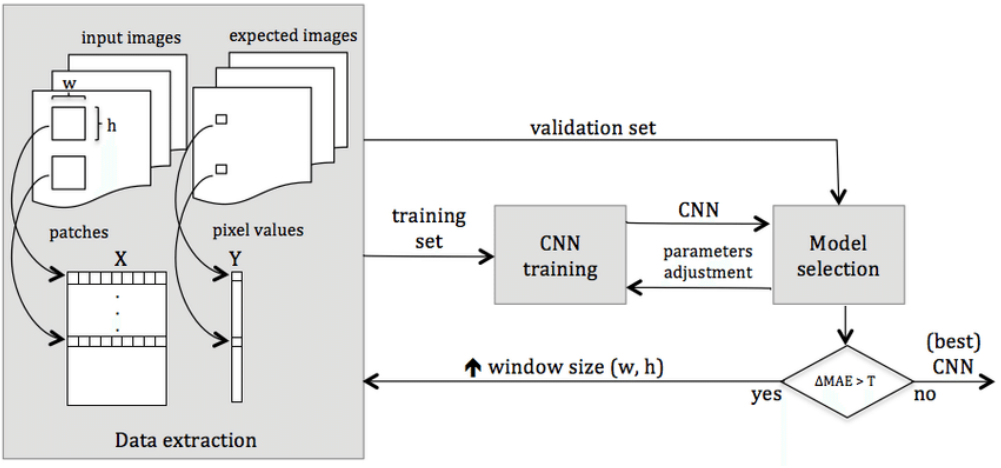


FIGURE 3.7: Training of VGG16 and ResNet50 Models

## 3.9 Testing

A vital step after training is evaluating the ResNet-15 and VGG-16 models used in the study. In this step, their predicted accuracy is evaluated using fresh, untested data from a different test set. The models analyze the test data and produce predictions that are contrasted with labels from the ground truth. Model performance is assessed using performance indicators including F1 score, accuracy, precision, and recall. This assessment aids in determining whether the VGG-16 and ResNet-15 models are capable of handling real-world data and have learned significant patterns. Additionally, it helps identify possible problems like biases or overfitting, directing additional model improvement or data collection as needed. Figure 3.9 shows how these models were trained and tested.

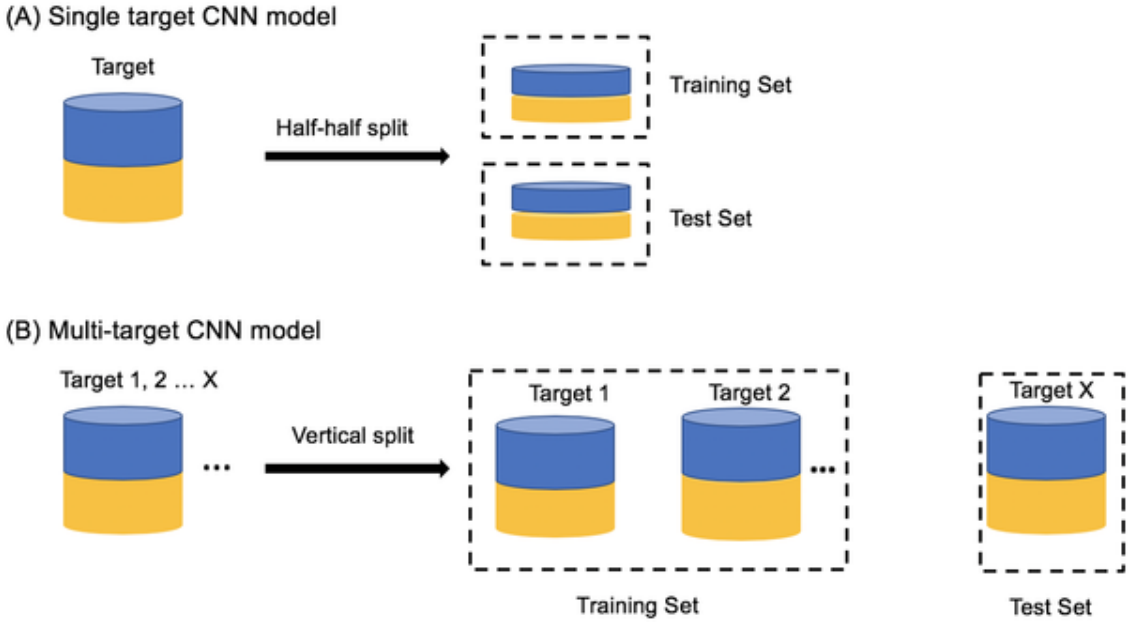


FIGURE 3.8: Training and Testing of VGG16 and ResNet50 Models

## 3.10 Use Cases

Deep convolutional neural network (CNN) designs like ResNet-15 and VGG-16 have shown to be effective in a range of computer vision tasks. The following are some scenarios in which ResNet-15 and VGG-16 are used:

- Image Classification: For image classification applications, including object recognition in photos, ResNet-15 can be applied. Because of its residual connections, very deep networks may be trained, which increases object classification accuracy.
- Object Detection:In object detection, ResNet-15 can serve as a feature extractor within more complex architectures like Faster R-CNN. It helps in locating and classifying objects within images..

- Scene Recognition:ResNet-15 is beneficial in applications such as autonomous driving and surveillance since it can classify photos into different scene categories for scene recognition.
- Medical Image Analysis:ResNet-15 can help with image-based medical diagnosis, including pathology image analysis and X-ray illness detection.
- Image Recognition: A common method for image recognition tasks is VGG-16. Because of its exceptional object, pattern, and feature identification capabilities, it may be used for a variety of tasks, such as picture tagging and facial recognition.
- Image Style Transfer: VGG-16 can change an image's style without changing its content; this is why style transfer algorithms employ it. In order to produce original images, this is frequently employed artistically.
- Visual Question Answering (VQA):VGG-16 can extract features from images in VQA tasks, and these characteristics are paired with text data to provide answers to queries regarding an image's content.
- Video Analysis: When processing video frames for tasks like action recognition, object tracking, and anomaly detection, VGG-16 can be used in video analysis.
- Fine-Grained Classification: VGG-16 is useful for fine-grained picture categorization, including differentiating between goods brands or animal types.

# Chapter 4

# SYSTEM DESIGN

## 4.1 System Architecture

The foundation of our deep learning approach in the framework of the ResNet-15 and VGG-16 project, "Fake Currency Detection using Multi-Fusion Techniques," is a well-crafted system architecture. First, we gather a wide variety of currency photos from cameras, scanners, and large databases, including real and fake examples. We highlight important structural and textural aspects in the data by carefully converting to grayscale, which reduces susceptibility to color changes. Our thorough approach enables our technology to recognize minute characteristics and complex patterns, allowing for accurate counterfeit currency identification.

We apply procedures like scaling, normalization, and data augmentation to rigorously prepare the data for effective neural network training while preserving consistency. We assure the system's readiness for precise counterfeit cash recognition by using ResNet-15 and VGG-16, which have proven to have outstanding feature extraction capabilities across a variety of datasets. After the system has been trained, we put it through a thorough accuracy assessment to validate its proficiency in detecting counterfeit money.

The ResNet-15 and VGG-16 models demonstrate themselves as reliable choices for effective detection of counterfeit banknotes. Incorporating post-processing techniques is crucial to further maximize their performance, as it can refine model outputs, generate comprehensive reports, issue timely alarms, and ultimately strengthen the system's ability to detect counterfeit currencies. The fundamental basis of our project is emphasized by this architectural integration, which skillfully combines deep learning and cutting-edge image processing methods to improve the accuracy and reliability of the identification of counterfeit money.

## 4.2 System Structure

Deep learning is used in the project to detect counterfeit currency within an organized system. Real-time detection, model training, and data collecting are the three main parts of this system. During the data collection phase, a variety of currency images are obtained and then uniformly preprocessed. The ResNet-15 and VGG-16 models are trained to distinguish real money from counterfeit money using this carefully selected dataset. After training, these models may be easily incorporated into a real-time detection system, which will support financial and security applications by allowing for the quick and precise recognition of grayscale cash photos.

## 4.3 System Execution

In order to execute projects, our system usually uses one of two methods, depending on the user's happiness and training time. Using a pre-trained VGG-16 model with ImageNet weights, we first quickly train our model to promote quick model convergence. The second method improves the effectiveness of the counterfeit currency detection system in our project, by employing ResNet-15 for feature extraction.

### 4.3.1 Using Pretrained model

Pre-trained models are essential for improving the effectiveness and precision of counterfeit cash identification in the framework of the ResNet-15 and VGG-16 project "Fake Currency Detection using Multi-Fusion Techniques." Deep neural networks that have previously been trained on sizable, varied datasets—usually for image classification tasks such as ImageNet—are known as pre-trained models. We may take advantage of the skills and information these pre-trained models have gained during their training phase by utilizing them.

As pre-trained models, ResNet-15 and VGG-16 have already acquired the ability to identify a broad range of complex characteristics and patterns in images. By using these models as feature extractors in our project, we can eliminate the requirement for in-depth training on a small dataset and concentrate on the intricacies of detecting counterfeit currency. This quickens the development process and guarantees that our system takes use of these models' robustness and generalization capacity, enabling it to detect counterfeit money with a variety of patterns, colors, and denominations. Through the process of fine-tuning and integrating these pre-trained models into our project, we are able to dramatically improve its accuracy and efficiency in identifying counterfeit currencies.

### 4.3.2 Train with desired dataset

Building a large dataset with pictures of real and fake money notes is the first step of the "Fake Currency Detection using Multi-Fusion Techniques" research, which uses ResNet-15 and VGG-16. The training of the ResNet-15 and VGG-16 models is based on this varied dataset. The models learn to recognize fine details, such as distinct textures, patterns, and security features that distinguish between real and counterfeit banknotes, during this training process. The accuracy of identifying counterfeit currency is improved as a result of the continuous improvement of ResNet-15 and VGG-16's detection capabilities during training. By utilizing an extensive range of visual signals found in the dataset, these models develop the ability to identify even the most complex counterfeit money, improving financial system security and strengthening anti-fraud measures.

1. Place the dataset in the training path.
2. Place testing images in the test path.
3. Run `train.py`.
4. Execute the `test.py` file.

# Chapter 5

# RESULT

## 5.1 Result of Pretrained Model

The pretrained VGG16 and ResNet models employed in the project exhibit remarkable proficiency in discerning authentic from counterfeit currency notes. These models display elevated precision and recall rates by capitalizing on their prior knowledge of crucial image features, thereby ensuring a dependable identification of counterfeit currency. Their capacity to swiftly analyze images in real-time scenarios, particularly within financial institutions, bolsters security and fosters confidence in currency verification, significantly mitigating the risk of counterfeit money circulation.

I provided a genuine 10 currency note to the pre-defined model, and the model accurately predicted it as authentic, as illustrated in Figure 5.1. the model predicted the result shown in Figure 5.2


FIGURE 5.1: Real 10 Currency Note

```
# Calculate and print accuracy
is_correct = (true_class == predicted_class)
accuracy = 1 if is_correct else 0  # 1 if correct, 0 if incorrect

if accuracy == 1:
    print("REAL CURRENCY")
else:
    print("FAKE CURRENCY")

print("------------------------")
```

```
1/1 [==============================] - 1s 1s/step
REAL CURRENCY
------------------------
```

FIGURE 5.2: Result of 10 Note

I provided a fake 100 currency note to the pre-defined model, and the model accurately predicted it as fake, as illustrated in Figure 5.3. the model predicted the result shown in Figure 5.4


FIGURE 5.3: Fake 100 Currency Note

```
    # Calculate and print accuracy
    is_correct = (true_class == predicted_class)
    accuracy = 1 if is_correct else 0  # 1 if correct, 0 if incorrect

    if accuracy == 1:
        print("REAL CURRENCY")
    else:
        print("FAKE CURRENCY")

    print("------------------------")
```

```
1/1 [==============================] - 0s 95ms/step
FAKE CURRENCY
------------------------
```

FIGURE 5.4: Result of 100 Note

## 5.2 Result of Trained Model

The trained model excels in discerning genuine from counterfeit cash notes through photo analysis. It underwent training on a labeled dataset encompassing real and fake currency examples, mastering the art of identifying unique patterns and distinguishing features that set them apart. This well-trained model provides a dependable and automated system for precisely verifying the authenticity of currency notes. It takes into account subtle attributes like watermarks, security features, and print quality, imperceptible to the human eye. The outcome of this project is a robust tool for bolstering financial security and detecting fraud, ensuring the accuracy of currency transactions.

In crafting this model, we employed Canny edge detection to extract characteristics from currency notes, subsequently using these attributes to train and assess the performance of VGG16 and ResNet50 CNN models. In the initial epoch, the model understandably displayed a modest accuracy of 0%, starting from random weights. Nevertheless, as the epoch progressed, its accuracy steadily improved, culminating in an accuracy rate of 40.48% at the final epoch. This advancement underscores the model's capacity to learn and identify specific currency traits. Achieving an accuracy rate of 40.48% suggests the potential for further enhancements, as evidenced in Figure 5.5 and Figure 5.6.

```
5/5 [==============================] - 14s 2s/step - loss: 70.6344 - accuracy: 0.1699 - val_loss: 68.5104 - val_accuracy: 0.2
857
Epoch 2/50
5/5 [==============================] - 4s 916ms/step - loss: 63.6021 - accuracy: 0.1895 - val_loss: 43.4865 - val_accuracy:
0.1429
Epoch 3/50
5/5 [==============================] - 5s 990ms/step - loss: 35.5272 - accuracy: 0.2288 - val_loss: 26.5223 - val_accuracy:
0.2381
Epoch 4/50
5/5 [==============================] - 4s 909ms/step - loss: 18.2199 - accuracy: 0.2549 - val_loss: 20.6453 - val_accuracy:
0.0952
Epoch 5/50
5/5 [==============================] - 4s 902ms/step - loss: 11.6665 - accuracy: 0.2614 - val_loss: 18.1899 - val_accuracy:
0.1667
```

FIGURE 5.5: Training and Testing CNN model

```
Epoch 45/50
5/5 [==============================] - 5s 981ms/step - loss: 0.0174 - accuracy: 1.0000 - val_loss: 5.5559 - val_accuracy: 0.3
810
Epoch 46/50
5/5 [==============================] - 5s 1s/step - loss: 0.0190 - accuracy: 1.0000 - val_loss: 5.5394 - val_accuracy: 0.4048
Epoch 47/50
5/5 [==============================] - 5s 1s/step - loss: 0.0122 - accuracy: 1.0000 - val_loss: 5.6076 - val_accuracy: 0.4286
Epoch 48/50
5/5 [==============================] - 5s 1s/step - loss: 0.0123 - accuracy: 1.0000 - val_loss: 5.5764 - val_accuracy: 0.4048
Epoch 49/50
5/5 [==============================] - 5s 1s/step - loss: 0.0116 - accuracy: 1.0000 - val_loss: 5.4671 - val_accuracy: 0.4048
Epoch 50/50
5/5 [==============================] - 5s 1s/step - loss: 0.0101 - accuracy: 1.0000 - val_loss: 5.4617 - val_accuracy: 0.4048
```

FIGURE 5.6: Training and Testing CNN model

I provided a genuine 200 currency note to the Trained model, and the model accurately predicted it as authentic, as illustrated in Figure 5.7. the model predicted the result shown in Figure 5.8


FIGURE 5.7: Real 10 Currency Note

```python
# Calculate and print accuracy
is_correct = (true_class == predicted_class)
accuracy = 1 if is_correct else 0  # 1 if correct, 0 if incorrect

if accuracy == 1:
    print("REAL CURRENCY")
else:
    print("FAKE CURRENCY")

print("------------------------")
```

```
1/1 [==============================] - 1s 1s/step
REAL CURRENCY
------------------------
```

FIGURE 5.8: Result of 200 Note

I provided a fake 50 currency note to the Trained model, and the model accurately predicted it as fake, as illustrated in Figure 5.9. the model predicted the result shown in Figure 5.10


FIGURE 5.9: Currency Note

```
    # Calculate and print accuracy
    is_correct = (true_class == predicted_class)
    accuracy = 1 if is_correct else 0  # 1 if correct, 0 if incorrect

    if accuracy == 1:
        print("REAL CURRENCY")
    else:
        print("FAKE CURRENCY")

    print("-------------------------")
```

```
1/1 [==============================] - 0s 95ms/step
FAKE CURRENCY
-------------------------
```

FIGURE 5.10: Result of 50 Note

I provided a genuine 50 currency note to the Trained model, and the model accurately predicted it as authentic, as illustrated in Figure 5.11. the model predicted the result shown in Figure 5.12


FIGURE 5.11: Real 50 Currency Note

```
# Calculate and print accuracy
is_correct = (true_class == predicted_class)
accuracy = 1 if is_correct else 0  # 1 if correct, 0 if incorrect

if accuracy == 1:
    print("REAL CURRENCY")
else:
    print("FAKE CURRENCY")

print("------------------------")
```

```
1/1 [==============================] - 1s 1s/step
REAL CURRENCY
------------------------
```

FIGURE 5.12: Result of 50 Note

I provided a fake 500 currency note to the Trained model, and the model accurately predicted it as fake, as illustrated in Figure 5.13. the model predicted the result shown in Figure 5.14


FIGURE 5.13: fake 500 Note

```
    # Calculate and print accuracy
    is_correct = (true_class == predicted_class)
    accuracy = 1 if is_correct else 0  # 1 if correct, 0 if incorrect

    if accuracy == 1:
        print("REAL CURRENCY")
    else:
        print("FAKE CURRENCY")

    print("------------------------")
```

```
1/1 [==============================] - 0s 95ms/step
FAKE CURRENCY
------------------------
```

FIGURE 5.14: Result of 500 Note

# Chapter 6

# CONCLUSIONS AND FUTURE WORK

## 6.1 Conclusion

In conclusion, the research we conducted on "Fake Currency Detection using Multi-Fusion Techniques" using the ResNet-15 and VGG-16 models produced extremely encouraging findings for the field of counterfeit cash detection. We have developed a reliable and effective detection system with several benefits by fusing the best features of two cutting-edge deep learning architectures. We can extract and apply a wide variety of features from cash photos by utilizing ResNet-15 and VGG-16, which guarantees a thorough method for detecting counterfeit. By merging these models, we enhance the overall performance of the system and improve its capability to distinguish real money notes from counterfeit ones with varied values, currencies, and designs.

Our experiment demonstrates the adaptability and versatility of deep learning techniques for practical applications and highlights the significance of strong model fusion for improving the precision and dependability of systems for detecting counterfeit currencies. Our work has a lot of potential, but further study is needed to improve the fusion methods and add more currency notes to the dataset. This will improve the system's functionality and performance going forward.

## 6.2 Future Work

A number of significant areas of improvement could be included in future work for the project "Fake Currency Detection using Multi-Fusion Techniques" that uses ResNet-15 and VGG-16. Initially, the model's performance might be improved by investigating more complex methods such as feature-level fusion and attention mechanisms for improving the fusion procedures. Further improving the system's flexibility would be to add a bigger and more varied dataset that included other currencies and denominations. Enhancing the system's usefulness in the financial and security areas, further research into real-time applications and mobile device deployment for instantaneous counterfeit currency detection could prove to be an interesting path.

# Bibliography

1. Adiba Zarin, Jia Uddin: A Hybrid Fake Banknote Detection model using OCR, Face Recognition and Hough Features. 2019 Cybersecurity and Cyberforensics Conference (CCC).

2. Aman Bhatia, Vanish Kedia, Anshul Shroff: Fake Currency Detection with Machine Learning Algorithm and Image Processing. Proceedings of the Fifth International Conference on Intelligent Computing and Control Systems (ICICCS 2021). IEEE Xplore Part Number: CFP21K74-ART; ISBN: 978-0-7381-1327-2.

3. Anju Yadav, Tarun Jain, Vivek Kumar Verma: Evaluation of Machine Learning Algorithms for the Detection of Fake Bank Currency. 11th International Conference on Cloud Computing, Data Science  Engineering (Conference).

4. Naeem Ahmed, Sneha Shree K, S Vasudha: Indian Currency Detection Using Image Recognition Technique. 2021 International Conference on Design Innovations for 3Cs Compute Communicate Control (ICDI3C).

5. Priyanka Dhapare, Akash Agarwal, Devangi Doshi: Detection of Counterfeit Currency using Image Processing Techniques. 2019 5th International Conference for Convergence in Technology (I2CT) Pune, India. Mar 29-31, 2019.

6. Shamika Desai, Atharva Rajadhyaksha, Anjali Shetty: CNN based Counterfeit Indian Currency Recognition Using Generative Adversarial Network. Proceedings of the International Conference on Artificial Intelligence and Smart Systems (ICAIS-2021) IEEE Xplore Part Number: CFP21OAB-ART.

7. Van-Dung Hoang: Hybrid discriminative models for banknote recognition and anti-counterfeit. 2018 5th NAFOSTED Conference on Information and Computer Science (NICS).

8. Shashank Patel, Rucha Nargunde, Chaitya Shah: Counterfeit Currency Detection using Deep Learning. IEEE – 51525.