IBB UNIVERSITY

Faculty of Engineering

Department of Electrical Engineering

Classification of The Current Banknote of Yemeni Currency Using Deep Learning

A project submitted to the Department of Electrical Engineering, Faculty of Engineering, IBB UNIVERSITY, in partial fulfillment of the requirements for the award of the degree of **B.Sc.** in **Computer Engineering and informatics**.

Done By:

Mohammed Taha Mohammed Altwil.

Majed Abdullah Abdulqawi Molhi.

Helal Saleh Hussein Meftah.

Mohammed Ali Mohammed Albatol.

Muhammad Abdullah Yaseen Qasem.

Supervised By:

Dr. Farhan Nashwan

2021

IBB –YEMEN

Department of Electrical Engineering
Faculty of Engineering
IBB UNIVERSITY
IBB –YEMEN

CERTIFICATE

This is to certify that this project entitled **"Classification of The Current Banknote of Yemeni Currency Using Deep Learning"** is being submitted to the Department of Electrical Engineering, Faculty of Engineering, IBB UNIVERSITY, as a partial requirement for the award of **B.Sc**. in **Computer Engineering and informatics**, is a bonfire project carried out under my supervision.

Name: Dr. Farhan Nashwan
(Project Supervisor)
Signature:

Date:

Approved By:

Name: Dr. **Name: Dr.**

Signature: **Signature:**

In view of the available recommendation, I forward this project for debate by the project discussion committee.

Name: Dr. Hajar Al-Suhbani
(Head of the Department)
Signature:

Date:

ACKNOWLEDGEMENTS

First of all, All Praise and thanks to Almighty ALLAH for guidance, blessing, strength, opportunity, and endurance to complete this project. We would like to convey our heartfelt appreciation and gratitude to our supervisor **Dr. Farhan Nashwan** who provide us the opportunity to work on this project, which aided us in conducting extensive study and learning about a bunch of new topics. We are all entirely grateful for him and everything we have learned.

Second, many thanks to our engineering faculty, which was the greatest light in the sky of our studies. Many thanks to all electrical department staff, including all Doctors, lab technicians, and everyone who works in our department.

Finally, we would like to express our gratitude toward our beloved families and friends who encouraged us in our Academic journey. Our experience at IBB University expanded our knowledge and abilities, we are very grateful. Many thanks to everyone who has helped.

Abstract

Deep learning (DL) is a branch of machine learning (ML). DL technology is considered one of the hot topics within machine learning, artificial intelligence as well as data science, and analytics. Due to its learning capabilities from the given data. DL is widely applied in various applications like visual recognition, text analysis, cybersecurity, and more.

Nowadays, automated banking machines and the automatic sale of goods and banking services that can differentiate between Banknote currencies with different designs is a critical task.

This project aims to build a model that has the ability to differentiate between different banknotes using deep learning, and the focus will be on Yemeni banknotes, in particular, using a dataset collected from different sources. Three different pre-trained deep Convolutional Neural networks (CNN) are MobileNet, DenseNet121, and EfficientNetV2 They are trained on the same dataset with constant parameters for each.

The models achieved good promising results in both training and validation, Those Efficient models mentioned above reach a value of 97.55%,99.75%, and 99.70% respectively in terms of testing accuracy.

Table of Contents

List of Figures

Chapter 2: LITERATURE REVIEW

Chapter 3: METHODOLOGY

Chapter 4: RESULT AND DISCUSSION

List of Table

Chapter 3: METHODOLOGY

Chapter 4: RESULT AND DISCUSSION

List of Abbreviations

Abbreviation	Explanation
ADAM	Adaptive Moment Estimation
AI	Artificial Intelligence
ANN	Artificial Neural Network
ATM	Automated Teller Machines
BGD	Batch Gradient Descent
BN	Batch Normalization
CNN	Convolutional Neural Network
CV	Computer Vision
DL	Deep Learning
DNN	Deep Neural Network
ELU	Exponential Linear Unit
FC	Fully Connected
FN	False Negative
FP	False Positive
GN	Group Normalization
GPU	Graphical Processing Unit
GUI	Graphical User Interface
IN	Instance Normalization
JPEG	Joint Photographic Experts Group
LN	Layer Normalization
MAE	Mean Absolute Error
MBGD	Mini-Batch Gradient Descent
ML	Machine Learning
MSE	Mean Square Error
NN	Neural Network
PRELU	Parametric Rectified Linear Unit
RELU	Rectified Linear Unit
RMSPROP	Root Mean Squared Propagation
SGD	Stochastic Gradient Descent
TN	True Negative
TP	True Positive
VGG	Visual Geometry Group
YB	Yemeni Banknote
YBC	Yemeni Banknote Classifier

Chapter 1

INTRODUCTION

1.1 Introduction

Generally speaking, a wide range of various application improvements for machine learning (ML) and artificial intelligence has been widely used in many fields to solve real-world problems. Some of these applications are self-driving cars, visual recognition, text analysis, and automatic banking machine.

Internationally, there are more than 250 various types of banknotes with different designs. Due to recent advances in technical banking services, many automatic methods require banknote recognition. Deep learning (DL) has proven to be an effective tool for improving the accuracy of banknote recognition systems. It has also been applied to other areas of image processing, including object classification, scene segmentation, and face recognition. DL has been successfully used to detect forged banknotes.

Banknote currency recognizers are used in many different places. Some of them are used in automated teller machines (ATMs). The need for automated Banknote currency recognition systems motivates many researchers to develop corresponding accurate and reliable systems for recognizing different Banknote currencies. These days, various types of others currency recognition systems and devices are being used for currency recognition. The existing advances in multilayer neural networks, which are called DL, have led to an effective technique for pattern recognition especially image classification. For this project, deep learning has had tremendous success to improve the accuracy of Yemeni banknotes recognition by enhancing the dataset and employing three pre-trained models; MobileNet, DenseNet121, and EfficientNetV2.

1.2 Motivation

According to developing systems and technology, we came up with ideas to deeply choose this project. After much consideration, we have finally decided to design an efficient system using a DL algorithm to recognize and be able to differentiate between the different categories of the Yemeni Banknote currency (YBC). Another reason for selecting this project was the rare research and scientific papers on the YBC.

1.3 Project Objectives

According to the goals of the project, the project obviously aims to produce an efficient banknote recognition system using DL for the YB currency and this system would be able to recognize and differentiate between the different YB and give you the appropriate class of the Banknote currency.

1.4 Common Difficulties of Currency Systems

In general, banknote detection and classification systems require a big dataset of pictures captured in different conditions. Such as position, light, … etc.

For this project, the big difficulties we have were:

1. There is no available dataset of Yemeni banknote.
2. The not torn banknote currency have been rare (especially for 50RY, and 200RY categories).
3. Limited time allowed when using Graphical Processing Unit (GPU) of Google Colab.

1.5 Scope and Limitations of The Project:

The scope of this work is limited to the following:

1. The system recognizes only YB.
2. The system recognizes only one Banknote currency in a given input image.

1.6 Project Organization

The organized chapters for this project as follow:

Chapter two discusses a general introduction of DL, Convolutional Neural Network (CNN), overview of pretrained model used in this project and some related studies in Currency Papers Recognition. Chapter three includes: a clear explanation for the project models and how to be implemented. Chapter Four contains the results of tested model and experiments on this project. Chapter five clarifies the conclusion for this project and the future work recommendations.

Chapter 2

LITERATURE REVIEW

This chapter offers the theoretical background and topics for the proposed Yemeni banknotes recognition. an introduction to DL and its approaches; additionally, we will discuss CNN and how they work, brief introduction for the pretrained model used in our project; and finally, we will mention related studies.

2.1 Introduction

A ML algorithm learns the underlying relationships in data and can make decisions without explicit programming such that machine will learn from its experience [15].

By using the various techniques of Artificial Intelligence (AI), ML enables computers to complete tasks without the assistance of humans [11].

There are three type of ML methods are supervised learning, which trains algorithms based on human-labeled input and output data, unsupervised learning, which provides the algorithm with unlabeled data so it can find a solution Structure within its input Data, and Reinforcement learning which the algorithm or agent used learns by interacting with its environment and getting positive and negative reward [25].

2.2 Deep Learning

DL is a sub-field of ML, where models inspired by the human brain are expressed mathematically [28]. In contrast to traditional machine learning, deep learning attempts to learn high-level features from mass data [20].

DL allows computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction [17].

In neural network (NN), "deep" usually refers to the number of hidden layers. In contrast to traditional NN, deep networks can have as many as 150 hidden layers A deep learning model is trained using large sets of labeled data and NN that learn features directly from the data [25] Figure 2.1 show the difference between simple NN and Deep Neural Network (DNN).

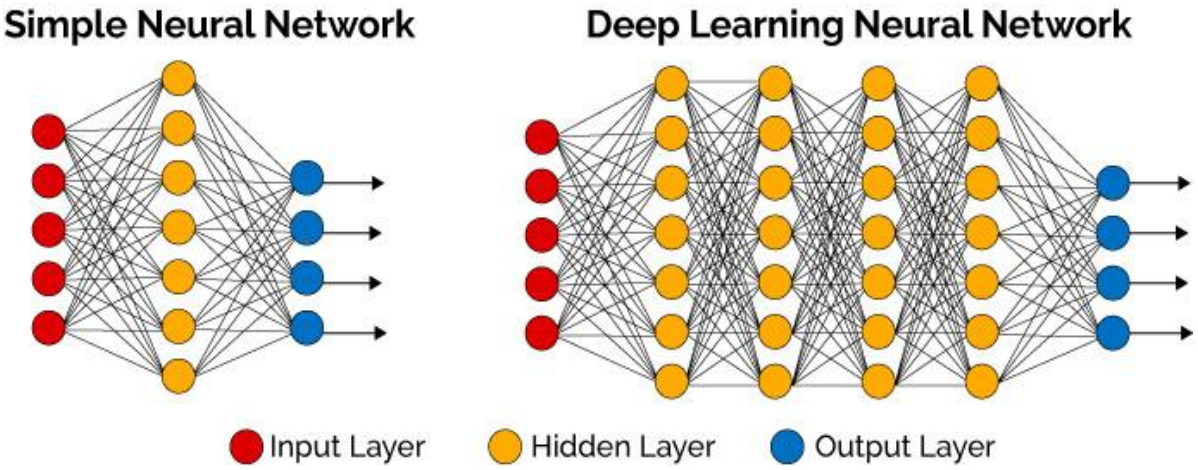


Figure 2. 1 : Difference Between Simple Neural Network and Deep Learning

DL has the advantage of generating features automatically from raw data, with features from higher levels of the hierarchy derived from lower-level features In contrast, traditional ML methods need to design features manually [26] By using complex models in DL, more complex problems can be solved rapidly and effectively, as they allow massive parallelization, thus increasing classification accuracy or reducing regression errors as long as adequate datasets are available [26].

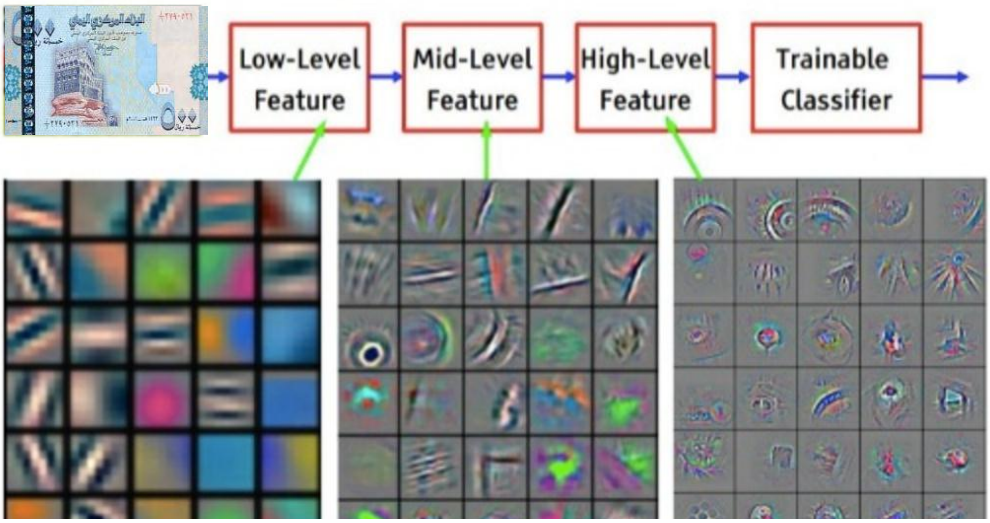


Figure 2. 2:Deep Learning Representations are hierarchical and trained

2.2.1 Computer Vision

Computer vision (CV) uses computers to simulate human visual science, aiming to make decisions and judgments based on image analysis and understanding. CV is used in image classification applications [9].

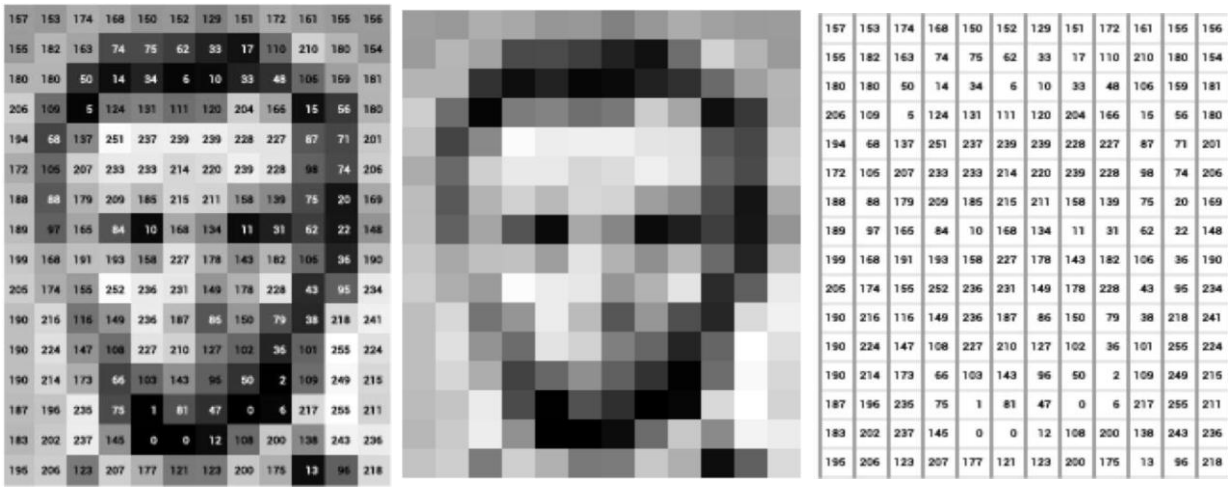


Figure 2. 3:Images represented as numbers

2.3 Convolutional Neural Network

In ML, CNN are algorithms that are specialized in detecting patterns and make sense of them; this ability makes CNN useful for image analysis. Every layer of a CNN performs a unique function on the data it receives [21] Nowadays, CNN is considered one of the most widely used ML techniques, especially in vision-related applications [15].

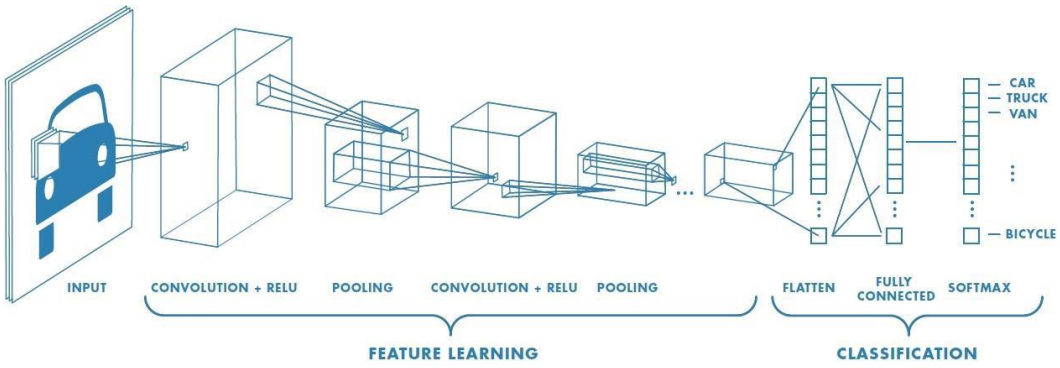


Figure 2. 4:Typical CNN architecture

2.3.1 CNN Architecture

CNN's can be divided into the following layers:

1. The input layer contains the image's pixels.

2. The convolutional layer determines the output of neurons connected to nearby regions of the input by using the scalar product between their weights and the input volume. This layer applies an activation function to each element of the convolution output

3. In the pooling layer, the input will be down-sampled along with its spatial dimensionality, further reducing the number of parameters.

4. Fully Connected (FC) layers perform the same tasks as standard ANNs and are then used for classification by generating class scores based on the activations [14].

2.3.2 Convolutional Layer

The first layer of the convolution algorithm is used to extract features from an input image. As a result of convolution, the relationship between pixels is preserved by learning image features from small squares of input data. There are two inputs to the convolution operation, such as an image matrix and a filter or kernel [25].

Convolutional layers are the key to DL, especially for image recognition. It is the first and the most important layer. It generates various feature maps by conveying the whole image as well as the intermediate feature maps with different filters. Feature maps consist of mappings from input to output, from input layers to hidden layers [10].

A CNN depends heavily on a convolutional layer, as its name implies. There are a number of parameters that define the use of learnable kernels in the layer. The kernels usually have a small spatial dimension, but spread across the entire depth. When the data hits a convolutional layer, the layer convolves each filter across the spatial dimensionality of the input to produce a 2D activation

map. the convolutional layer to work, each kernel will have its own activation map, which will be stacked along the depth dimension [15].

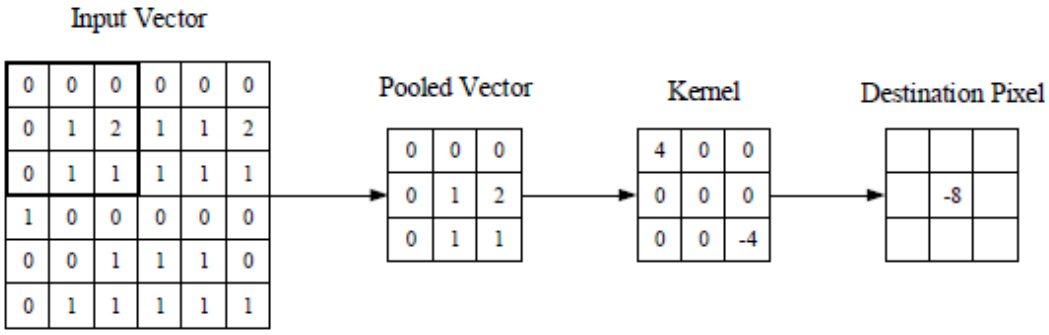


Figure 2. 5:visual representation of convolutional layer

2.3.2.1 Advantages of The Convolution Operation

1. the weight sharing mechanism in the same feature map reduces the number of parameters.

2. local connectivity learns correlations among neighboring pixels.

3. invariance to the location of the object [20].

Through the optimization of their output, convolutional layers can also significantly reduce the complexity of the model. This is accomplished by optimizing three hyperparameters, the depth, the stride, and setting zero padding.

A convolutional layer's depth can be manually set by adjusting the number of neurons within that layer to the same input region. As with other ANNs, all neurons in the hidden layer are connected directly beforehand. Increasing this hyperparameter can reduce the number of neurons, but it can also reduce the model's ability to recognize patterns [15].

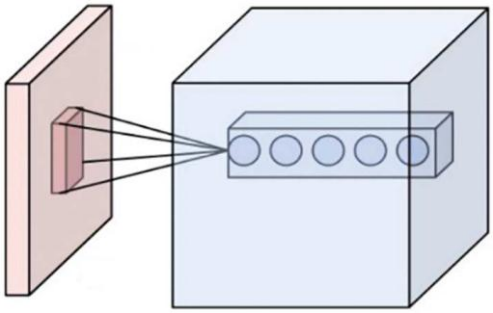


Figure 2. 6:Depth process

we can also define the stride in which we set the depth around the input. For example, if we were to set a stride as 1, then we would have a heavily overlapped receptive field producing extremely large activations. Alternately, increasing the stride will reduce overlapping and produce a lower spatial dimension output [15].

As shown in Figure 2.7, the stride determines how depth columns are allocated around the spatial dimensions (width and height). By setting stride to 1, the filters are moved one pixel at a time. This leads to heavily overlapping receptive fields between the columns, and also to large output volumes. A stride of 2 results in a jump of 2 pixels in the filters at a time as we slide them around. The receptive fields will overlap less and the resulting output volume will have smaller dimensions spatially [10].

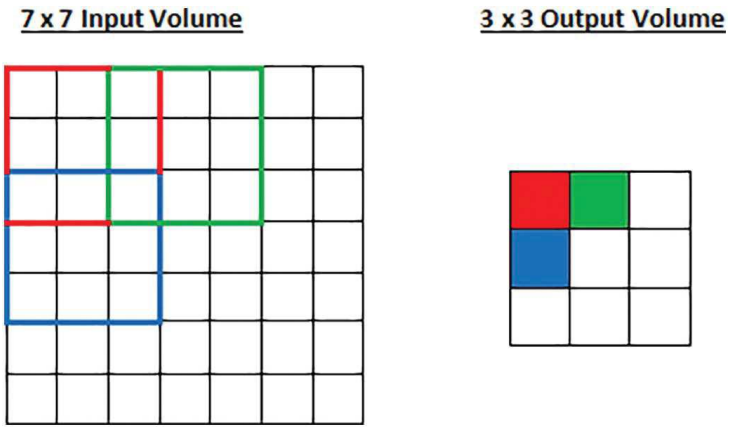


Figure 2. 7: Stride process

The zero-padding method simply pads the border of the input and allows for further control of the output volume dimension. Figure 2.8 illustrates an input volume of 32x32x3. By padding two borders of zeros around the volume, we can a volume of 36x36x3 is obtained. We will also get a 32x32x3 output volume when we apply the convolution layer with our 5x5x3 filters [10].

The parameter sharing approach is based on the assumption that if a feature is useful for one region, then it probably will also be useful in another region. If we constrain each individual activation map within the output volume to the same weights and bias, then we will see a massive reduction in the number of parameters being produced by the convolutional layer [15].

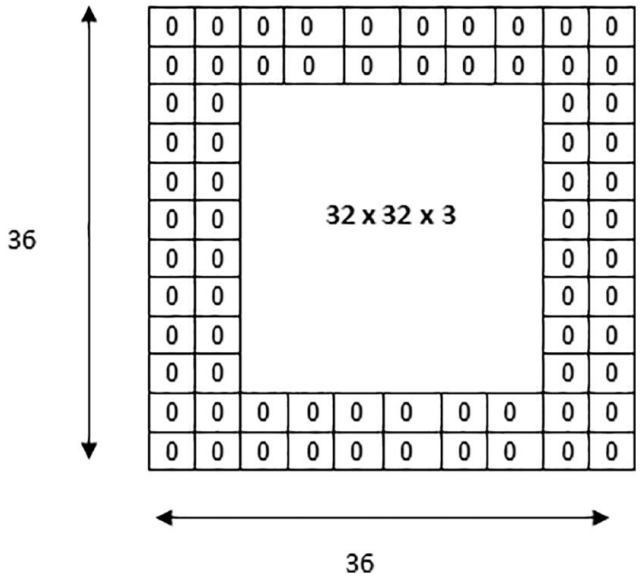


Figure 2. 8: Zero-padding process

Figure 2.9 illustrates an example of Learned features from a Convolutional Layers such that, first layers will learn edges and lines, second layers will learn some feature like eye, and the last layers will learn the important feature that we look for.

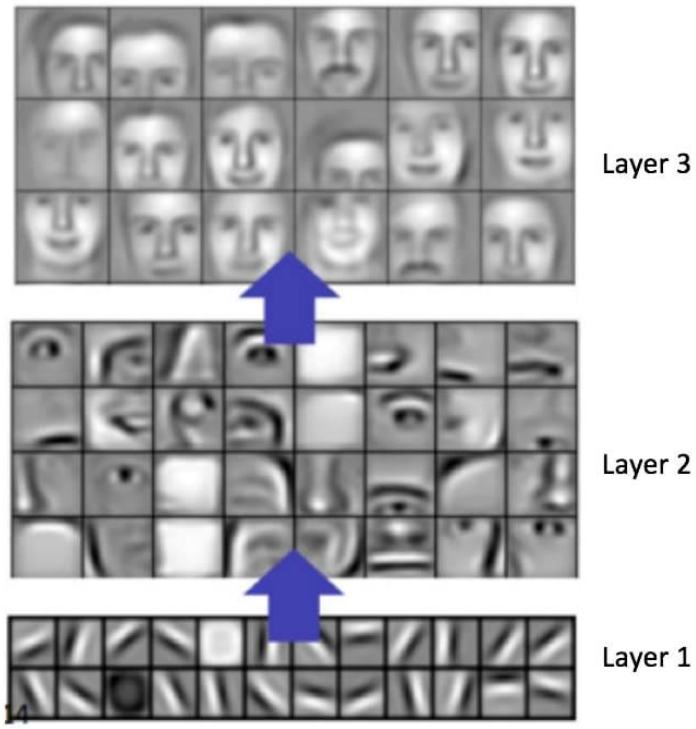


Figure 2. 9: Learned features from a Convolutional Layers

2.3.3 Pooling Layers

As a result of pooling layers, the dimensions of the layers will be reduced. As a result, the number of parameters to learn and the amount of computation in the network is reduced. A pooling layer summarizes the characteristics of an area of the feature map created by a convolution layer [11].

A pooling layer reduces the dimensionality of the representation, thus reducing the number of parameters and the computational complexity of the model [14].

Reduction in the size of the feature-map to invariant feature set not only regulates the complexity of the network but also helps in increasing the generalization by reducing overfitting. The various types of pooling formulations, such as max, average, L2, overlapping, and spatial pyramids, are available. They are used on CNNs [15].

2.3.3.1. Average Pooling

The concept of average or mean was introduced and first used in the first convolution-based deep neural network. as shown in Figure 2.10, an average

pooling layer performs down-sampling by dividing the input into rectangular pooling regions and computing the average values of each region [28].

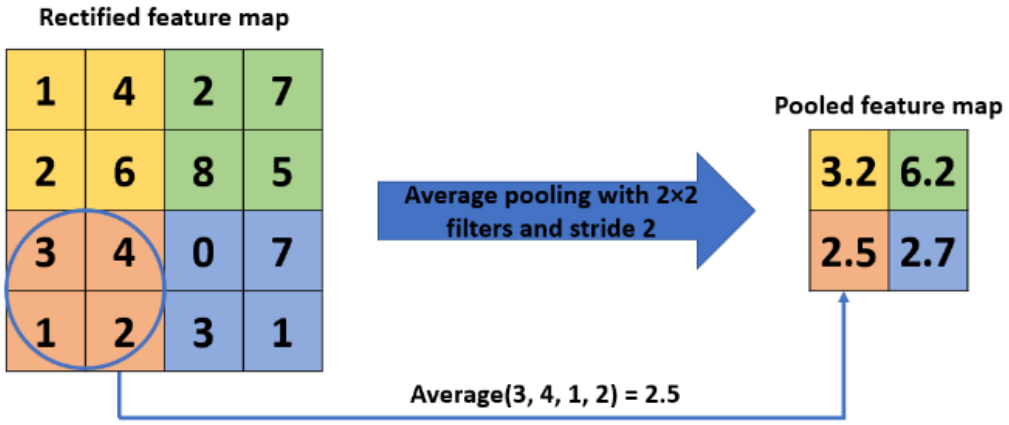


Figure 2. 10: Example of Average Pooling operation.

2.3.3.2 Max-Pooling

The convolutional output bands can be down-sampled, which lowers variability, using the max-pooling operation. The maximum value inside a set of R activations is sent forward by the max-pooling operator [28]. An example of the Max-Pooling operation is shown in Figure 2.11:

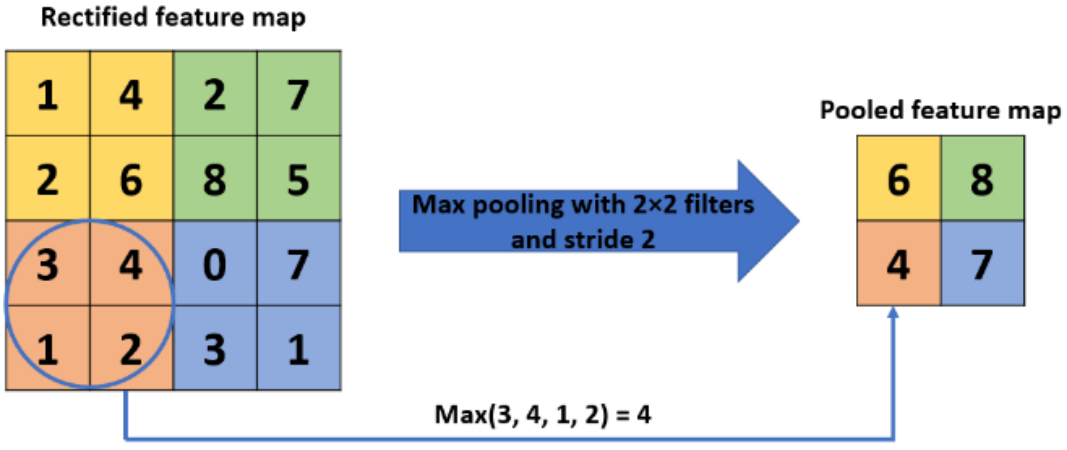


Figure 2. 11: Example of Max Pooling operation.

2.3.3.3 Mixed Pooling

Because average pooling down-weighs the activation by combining the non-maximal activation, max pooling only extracts the greatest activation. A hybrid strategy that combines average pooling and maximum pooling was offered as a solution to this issue [28].

2.3.4 Activation Function

The activation function acts as a decision-making function and aids in the recognition of complex patterns. The learning process can be sped up by choosing the right activation function. The equation defines the activation function for a convolved feature map, Different activation functions are used to instill non-linear feature combinations, including sigmoid, tanh, max out, SWISH, Rectified Linear Unit (RELU), and versions of RELU, such as leaky RELU, Exponential Linear Unit (ELU), and Parametric Rectified Linear Unit (PRELU). However, because they assist in solving the problem, RELU and its variants are recommended [8].

2.3.4.1 Rectified Linear Unit

RELU produces x if x is positive; else, it produces zero. It can be expressed formally as:

$$f(x)=\max(0, x) \qquad (2.1)$$

The function value of x is 0 when it is less than 0 and 1 when it is greater than or equal to 0. x itself serves as the function value. RELU function has a big benefit over sigmoid and tanh functions in that it helps accelerate learning. Tanh and Sigmoid participate in the derivative of RELU are a constant, unlike other exponential operations that divide while computing derivatives. Additionally, the gradient of the sigmoid and tanh functions is quite tiny if the value of x is too high or too little, which might make the function converge slowly. However, RELU's derivative is 0 when x is less than 0 and 1 when x is greater than 0, allowing for an ideal convergence effect [9].

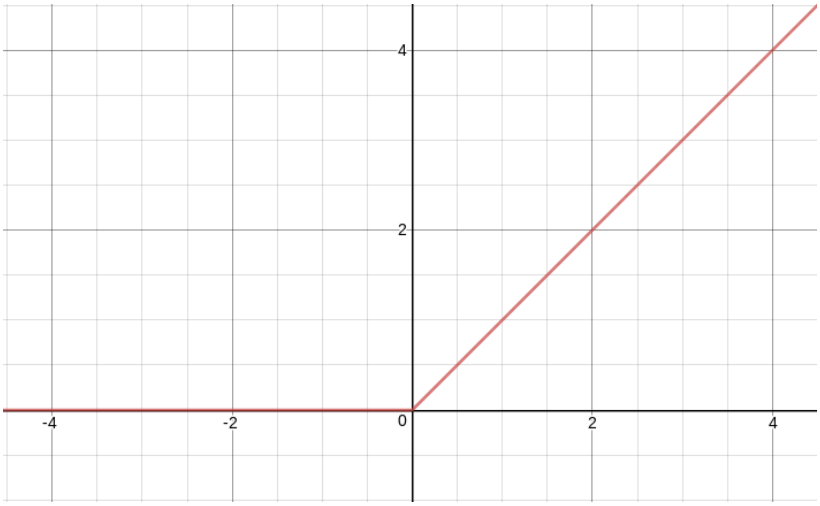
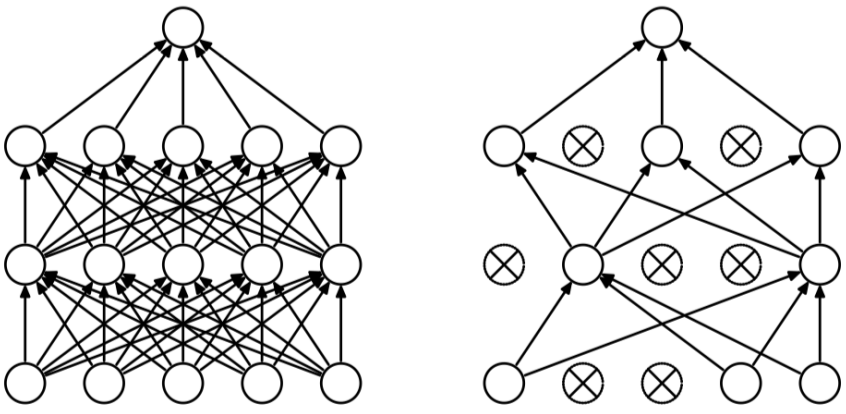


Figure 2. 12: The Rectified Linear Unit (RELU) activation function produces 0 as an output when $x < 0$, and then produces a linear with slope of 1 when $x > 0$.

2.3.5 Dropout

Dropout is a technique used to improve over-fit on neural networks [25]. by randomly skipping some units or connections with a specific probability, dropout creates regularization inside the network, which eventually enhances generalization.

Overfitting occurs in NNs when many connections that learn a non-linear relation cooperate [15]. Instead of relying on the prediction abilities of a limited fraction of network neurons, this constraint drives the network to learn more resilient features [22].



(a) Standard Neural Net (b) After applying dropout.

Figure 2. 13: Dropout is a simple way to prevent neural networks from overfitting

2.3.6 Normalizations for Different Architectures

Techniques of normalization. It is commonly known that normalizing the feature maps speeds up and stabilizes training. For general tasks or specific applications, a variety of normalization methods, such as Batch Normalization (BN), Layer Normalization (LN), Instance Normalization (IN), Group Normalization (GN), are created to normalize the activations [16]. all kinds of NNs can always select the one that suits their applied tasks the most. For CNN, BN became the most widely used normalization layer soon after it was proposed in 2015. Most breakthroughs of CNNs, such as Inception V2/V3, ResNet, DenseNet, MobileNet, and Efficient- Net choose BN as the default normalization method [30].

Although there do exist some unique normalization methods for a particular group of scenarios, such as GN for the small batch-size regime and IN for style transfer, they could hardly match the comprehensive performance of BN or provide similar regularization ability to BN.

The normalization technique has drawn a lot of attention in the literature as one of the most crucial elements of deep neural networks. BN, one of those techniques, is crucial for CNN vision tasks (CNNs). It primarily centers and scales the input to normalize it, demonstrating excellent regularization ability [30].

2.3.6.1 Batch Normalization

BN is another regularization technique that normalizes the set of activations in a layer. Normalization works by removing the batch mean from each activation and dividing it by the batch standard deviation. This normalization approach, along with standardization, is a standard technique in the preprocessing of pixel values [22]. BN is a frequently used approach for training deep neural networks quicker and more consistently (DNNs). Despite its widespread use, the precise reasons for Batch Norm's success are still unknown.

BN is a technique for improving NN training by stabilizing layer input distributions, this is achieved by introducing additional network layers that control the first two moments (mean and variance) of these distributions [29].

2.3.7 Fully Connected Layer

The fully linked layer is usually utilized for classification at the network's end. It is a global operation, as opposed to pooling and convolution. It accepts feature extraction input and analyzes the output of all preceding layers globally. As a result, it creates a non-linear combination of selected features for data classification [15].

This layer merely takes the output of the previous pooling layer as input and outputs an N-dimensional vector, where N is the number of classes from which the computer can choose. It enables us to feed the NN forward into a vector with a preset length. We might feed the vector forward into specific number categories for image classification. The result is a vector of numbers as well [10].

FC layers behave similarly to standard neural networks and comprise roughly 90% of the parameters in a CNN. It allows us to feed the NN forward into a vector with a preset length. We could either feed the vector forward into specific number categories for picture classification or use it as a feature vector for subsequent processing [20].

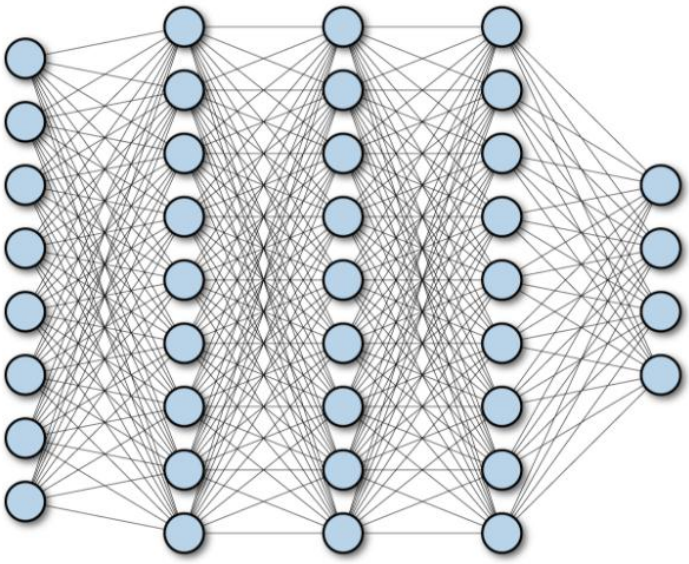


Figure 2. 14: Example of A fully connected layer in a deep network.

2.3.8 Softmax

Softmax functions are squashing functions. Squashing functions restrict the function's output to the 0 to 1 range. This allows the output to be simply understood as a probability. Similarly, Softmax functions are multi-class sigmoid, which means they are used to calculate the probability of numerous classes at the same time. Because the outputs of a Softmax function can be understood as a probability (i.e. they must add to 1), a Softmax layer is commonly employed as the final layer in neural network functions. It is important to realize that a Softmax layer must have the same number of nodes as the output layer [25].

The Softmax function is typically used in the calculation of FC layers to calculate the probability of the data after dimension reduction. CNN's final result is the image with the highest probability [23].

Softmax classifier benefits include efficient classification of fundamental issues that are linearly separable using the soft-max algorithm, as well as its simple model structure, which makes it straightforward to train and predict. This classifier has the disadvantage of only working on linearly separable data and does not enable null rejection [21].

The Softmax function can be expressed as follows:

$$F(x_i) = \frac{e^{x_i}}{\sum_{k=1}^{n} e^{x_j}} \quad (i=1, 2, \ldots, n) \qquad (2.2)$$

where $x_1, x_2, \ldots, x_N$ are the input values of the Softmax layer and the output value $f(x_i)$ represents the probability that the sample belongs to the i-th category. Mathematical transformation of Softmax involving exponential and logarithmic operations is adopted to perform the hardware implementation, which is shown in (2) [31].

$$F(x_i) = \exp\left(\ln\left(\frac{e^{x_i}}{\sum_{k=1}^{n} e^{x_j}}\right)\right) = \exp\left(x_i - ln\left(\sum_{k=1}^{n} e^{x_j}\right)\right)(i = 1,2, \ldots, n) \qquad (2.3)$$

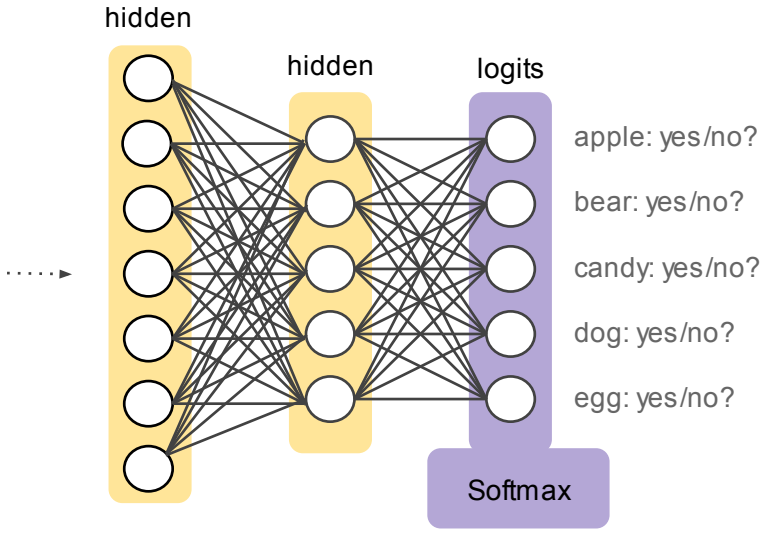


Figure 2. 15: A Softmax layer within a neural network.

2.3.9 Backpropagation

Backpropagation, short for "backward error propagation," is a supervised learning approach for artificial NNs that uses gradient descent. The approach computes the gradient of the error function with respect to the NN's weights given an artificial NN and an error function, backpropagation was one of the first approaches to show that artificial NN could learn good internal representations, i.e. their hidden layers could learn nontrivial features [25].

Backprop computes how slightly changing each synapse strength would change the network's error using the chain rule of calculus. Moreover, it does this computation for all the synapse strengths at the same time and it only requires the same amount of computation as is needed for a forward propagation pass through the Network [32].

Experts studying multilayer feedforward networks trained with backpropagation discovered that many nodes learnt properties comparable to those established by human experts and discovered by neuroscientists studying biological neural networks in mammalian brains (e.g. certain nodes learned to detect edges, while others computed Gabor filters) [25].

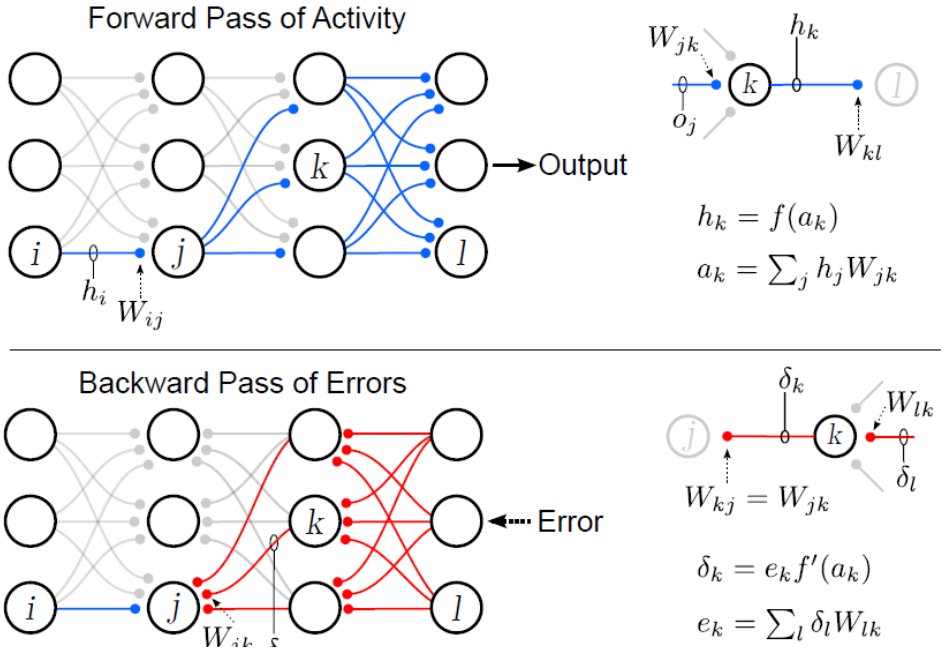


Figure 2. 16: The Backpropagation algorithm [32]

2.3.10 Optimizer

Non-convex functions are frequently optimized in CNNs. Because mathematical approaches need a large amount of computer power, optimizers are employed in the training process to minimize the loss function in order to obtain ideal network parameters in a reasonable amount of time. Momentum, RMSprop, An Adaptive Moment Estimation (Adam), and other popular optimization techniques [9].

2.3.10.1 Loss Function

The loss function or cost function is used to calculate the difference between the expected and actual values. The loss function is commonly used as an optimization problem learning criterion. Loss function can be utilized with CNN to solve regression and classification issues, with the goal of minimizing loss function. Mean Absolute Error (MAE), Mean Square Error (MSE), Cross Entropy, and other commonly used loss functions [9].

2.3.10.2 Gradient Descent

To train our CNN models, we can use one of three gradient descent methods: batch gradient descent (BGD), stochastic gradient descent (SGD), or mini-batch gradient descent (MBGD) (MBGD).The BGD implies that a large batch of data must be calculated to provide a gradient for each update, ensuring convergence to the global optimum of the convex plane and the local optimum of the non-convex plane [9].

For each update, SGD only uses one sample. Because just one sample's gradient is needed to calculate, SGD takes significantly less time per update than BGD. In this scenario, SGD is appropriate for online learning. Based on BGD and SGD, MBGD was proposed, which combines the benefits of both. MBGD uses a small batch of data for each update, allowing it to not only execute a more efficient gradient descent than BGD but also lower variance, resulting in more consistent convergence [9].

2.3.10.3 Adam Optimization

Optimizers are techniques or strategies for modifying the features of NNs, such as weights and learning rate, in order to reduce losses. Gradient Descent, Momentum, Adagrad, RMSProp, and more sorts of optimizers are available [11].

ADAM is one of the fundamental methodologies of optimization algorithms that compute individual adaptive learning rates based on the gradient's first and second moments [27].

Adam is a first-order gradient-based optimization technique for stochastic objective functions that is based on adaptive estimations of lower-order moments. The method is simple to develop, computationally efficient, requires little memory, is insensitive to gradient rescaling, and is well suited for issues with huge amounts of data and/or parameters. Adam is a method for efficient stochastic optimization that just requires first-order gradients and consumes little memory [27].

The approach calculates distinct adaptive learning rates for different parameters using estimations of the gradient's first and second moments; the name Adam is derived from adaptive moment estimation [27].

2.4 Transfer Learning

Transfer Learning is another fascinating paradigm for avoiding overfitting that works by first training a network on a large dataset like ImageNet and then using the weights from that training as the beginning weights in a new classification assignment. The weights of convolutional layers are typically replicated rather than the complete network including FC layers [22].

Transfer Learning is a ML technique in which a model is trained and produced for one task and then re-used on another. It refers to the situation in which what is learned in one environment is used to increase optimization in another [15]. Transfer Learning is typically utilized when a new dataset exists that is smaller than the original dataset used to train the pre-trained model [24].

Deep transfer learning is classified into four types based on the methodologies used: instances-based deep transfer learning, mapping-based deep transfer learning, network-based deep transfer learning, and adversarial-based deep transfer learning [18].

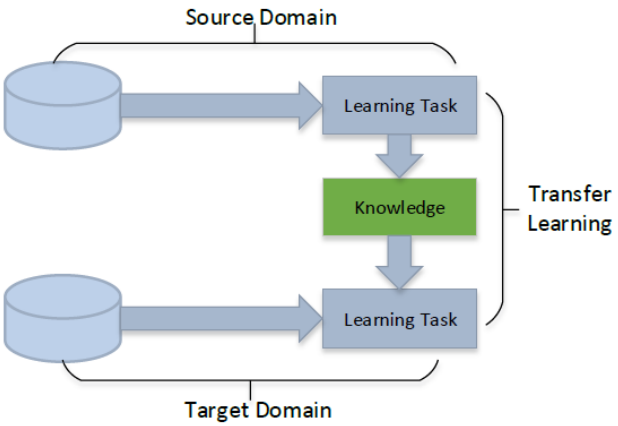


Figure 2. 17:Learning process of transfer learning[18]

2.5 Data Augmentation

CNN's perform better with larger datasets. The working database, on the other hand, is not that large. A typical tendency in DL algorithm training is to use data augmentation techniques to transform a very small dataset into a large one. Data augmentation has been shown to increase the classification accuracy of DL algorithms. Deep learning models' performance be increased by enhancing old data rather than collecting new data [12].

Existing image augmentation approaches can be divided into two broad categories: classic, white-box methods and deep neural network-based black-box methods. In this section, we will quickly describe groupings of methods that have had the most influence in the field of picture synthesis and enhancement. The most common contemporary approach for data augmentation is to combine affine picture alterations with color manipulation. We define rotation, reflection, scaling (zoom in/out), and shearing as affine transformations [19]. Figure 2.18 show Different image generated from original image using Data Augmentation methods.

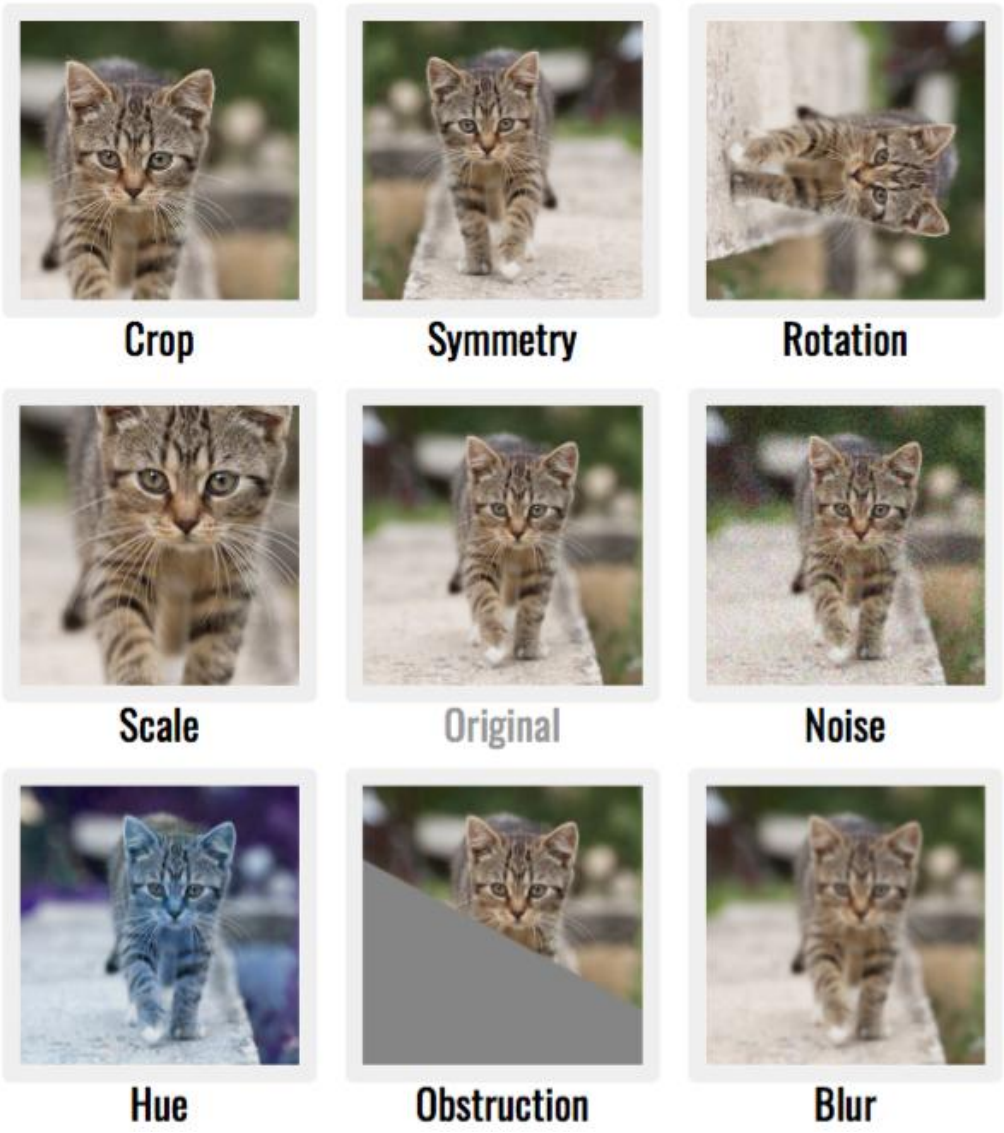


Figure 2. 18: Different image generated from original image using Data Augmentation

Geometric distortions or deformations are often employed to increase the number of samples for training deep neural models, balance dataset sizes [9], and improve their efficiency. Although affine transformations are commonly employed for data augmentation, it is still a study topic. Histogram equalization, boosting contrast or brightness, white balancing, sharpening, and blurring are the most frequent approaches [29].

The approach involves randomly painting a noise-filled rectangle in an image, which changes the original pixel values. As the authors explain, adding photos with varying degrees of occlusion to the dataset decreases the danger of overfitting and strengthens the model. Another strategy was introduced in response to the requirement to develop a robust CNN capable of defending itself against adversarial attacks. The picture dataset is supplemented by adding stochastic additive noise to the images, making them more resistant to noise adversarial attacks [19].

Image augmentation approaches that are now being developed are not restricted to classical and CNN-based methods. One intriguing option is a random erasing strategy, which is quick and simple to build while producing good results in CNN generalization ability [19].

2.6 Network Architectures

CNNs have three dimensions: width, height, and depth. the neurons in one layer of CNNs do not connect to all the neurons in the next layer but only to a small region of it also, the final output will be reduced to a single vector of probability scores, organized along the depth dimension .In this section we will A brief introduction for three network architectures will be introduced.

2.6.1 MobileNet

MobileNet CNN is designed for mobile and embedded vision applications. They are built to produce lightweight deep neural networks in mobile and embedded devices with low latency using depthwise separable convolutions [6]. Figure 2.19 illustrates MobileNet architecture. There are 28 layers in a MobileNet. A standard MobileNet has 4.2 million parameters that can be further reduced by adjusting the width multiplier hyperparameter. 224 * 224 * 3 is the size of the input image in MobileNet.

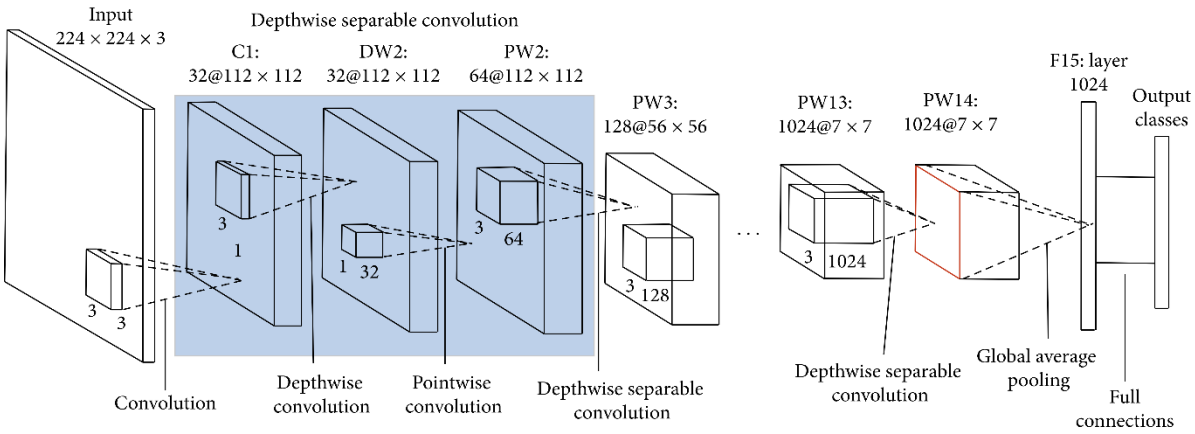


Figure 2. 19: Architecture of MobileNet (Source: ResearchGate)

2.6.2 DenseNet121

With DenseNet architecture, each layer is connected to another, hence the name Densely Connected Convolutional Network. A dense CNN uses fewer parameters because redundant feature maps are not learned. There are four different versions of DenseNet: DenseNet121, DenseNet169, DenseNet201, and DenseNet264. For Yemeni currency classification, we have used DenseNet121[7]. DenseNet's architecture is illustrated in Figure 2.20.

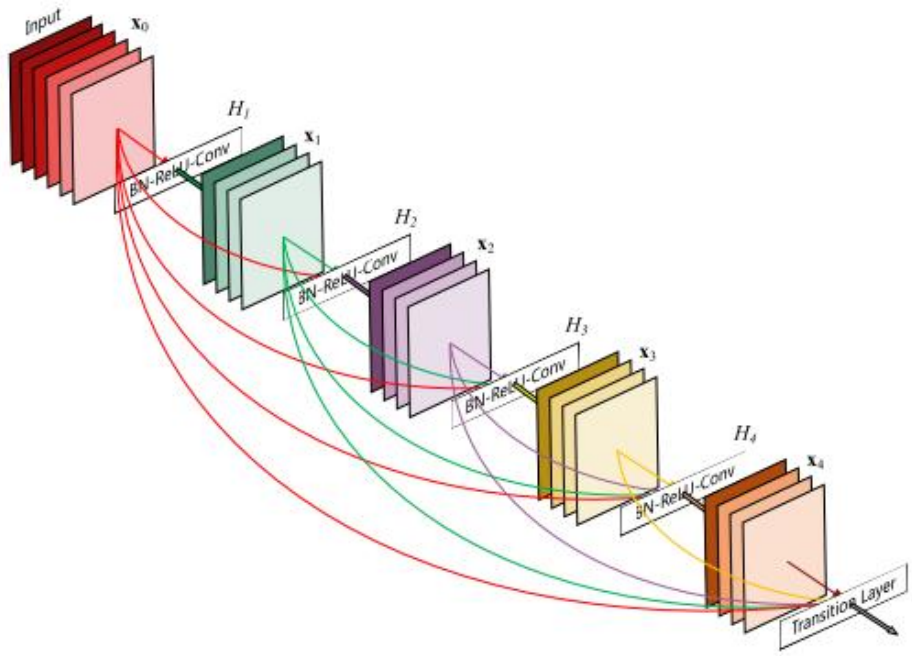


Figure 2. 20: DenseNet Architecture[7]

2.6.3 EfficientNetV2

The EfficientNetV2 family of convolutional networks has a faster training speed and better parameter efficiency than previous models. This method improves both the size of the image and the regularization of the image while training [8]. Figure 2.21 illustrates the EfficientNet architecture.

Figure 2. 21: EfficientNet Architecture (Source: ResearchGate)

2.7 Performance Matrix for Classification

The performance of the different networks for the testing dataset is evaluated after the completion of the training phase and was compared using the following Four performances metrics: accuracy, sensitivity or recall, precision, and F1 score.

$$\text{Accuracy} = \frac{(TP+TN)}{(TP+FN)+(FP+TN)} \qquad (2.4)$$

$$\text{Sensitivity (Recall)} = \frac{(TP)}{(TP+FN)} \qquad (2.5)$$

$$\text{precision} = \frac{(TP)}{(TP+FP)} \qquad (2.6)$$

$$\text{F1 score} = \frac{(2*TP)}{(2*TP+FN+FP)} \qquad (2.7)$$

Classification accuracy is the percentage of correct predictions where the top class with the highest probability matches the target label. A second performance precision is determined by dividing the true positives by the total number of relevant results. Third, the recall is the percentage of TPs and false negatives to the total number of false positives. Lastly, the harmonic mean of precision and recall is represented by the F1 Score. The performance matrix above is averaged across all classes in multiclass classification problems [26].

2.7 Related Studies

Many researchers have proposed different techniques for recognition of paper currencies based on image processing techniques, NNs, and DL. Some of these techniques are discussed in this section.

First, [1] proposed a currency recognition technique for Indian paper currency. This paper uses the process of currency recognition to extract visible and hidden characteristics of a currency. A simple and quite efficient algorithm was implemented for image processing, edge detection, segmentation, feature extraction, and image matching.

Using two classifiers, [2] proposes a new method for identifying Bahraini paper currency. One classification technique used to classify the minimum distances was the weighted Euclidean distances, and another was the Neural Network with back propagation. Different image processing techniques were used for preprocessing before the classifier, including the use of edge detection masks with Sobel, Prewitt, and Canny operators

The paper [3] examined how transfer learning was applied to banknote recognition. Three excellent models, Alexnet, Googlenet, and Vgg16 were trained and tested on Bosnian money papers containing 11 classes. The best accuracy was achieved by Vgg16 with 100% compared to Alexnet with 95.24% and Googlenet with 88.65%.

In [4], a system was proposed for the recognition of Bangla currency. For feature extraction, they used MobileNet, which is a smaller and more efficient model than VGG, Inception, and ResNet. For each category of the banknote,

the classification part consists of fully connected dense layers and one softmax layer. They perform different data augmentations to avoid overfitting. In the testing phase, they achieved a 99.8% accuracy rate.

Using pre-trained models, this paper [5] presented a method of recognizing Yemeni paper currencies using DL is used. Three pre-trained models are implemented which are AlexNet, DenseNet121, and ResNet50. The obtained results of the proposed method reach a value of 96.99%, 96.83%, and 99.04% in terms of validation accuracy using ResNet-50, DenseNet121, and AlexNet respectively.

Chapter 3

METHODOLOGY

In this section, the method has been used for the proposed model illustrated. This model can be used to classify various Yemeni Banknote currencies using CNN.

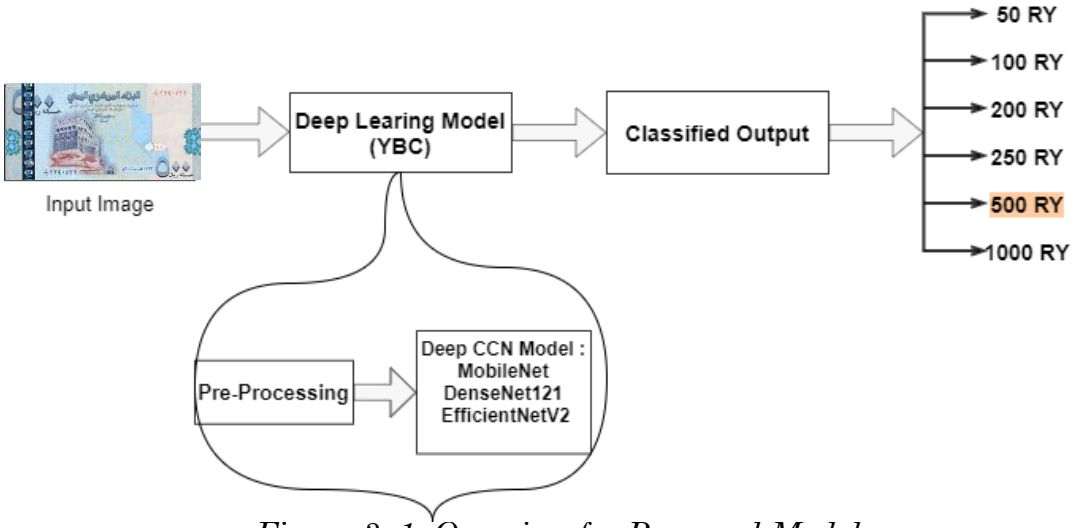


Figure 3. 1: Overview for Proposed Model

3.1 Introduction

In our project, we have been working on two different stages: the first with Dataset contains 930 Images divided into six classes, secondly, using data Augmentation of 6 and there have been some problems; to solve them in the second stage with Dataset of 1680 images divided into six classes and also trying another way by dividing Dataset into 12 classes. In the second stage, we have used different types of augmentation no-augmentation, 5 augmentations, and 10 augmentations, finally, we found the 5 augmentations are the better to use in our project and we choose the 6 classes approach because it give us better results.

3.2 Language and Tool Used

We have used Python programming language in the Google Colab environment. Google Colab is a research tool for ML education and research. Colab offers a free GPU cloud service hosted by Google; importantly it supports to development of DL applications using popular libraries such as Keras, and TensorFlow.

Since Colab is working on Google Drive, we uploaded the dataset to the Google Drive account, to be shared with the Google Colab environment, to be called by Python programming language, and used for training and testing the model.

Keras's deep learning framework was used for building the CNN. Necessary libraries were imported from Keras to train the model.

3.3 Dataset

3.3.1 Data Collection

A dataset of YB has been collected from scratch. Building this dataset starts with taking pictures of six classes of YBC in various environments. The six classes of YB are shown in Figure 3.2

Figure 3. 2shows the classes of Yemeni banknote currency.

For more details about the classes of YBC, table 3.1 provide a detailed description of the number of images that have been included in each class. Each class includes front and back pictures.

Table 3. 1: Collected dataset in each class

Categories	***No of Front images***	***No of Back images***	***Total images***
50 Rials	188	113	301
100 Rials	174	119	293
200 Rials	164	115	279
250 Rials	168	130	298
500 Rials	160	136	296
1000 Rials	176	110	286

The total number of these collected images reached 1753 for all classes, depending on the class 1000 Rials which contains the least images shown in Table 3.1 we chose 280 random images from each class including 170 front images and 110 back images, on which the data was divided into:

60% of images were selected randomly to be used during the training process.

20% of images were selected randomly to be used during the validation processes

20% of images were selected randomly to be used during the testing process as shown in Table 3.2.

Table 3. 2show split dataset into train, valid and test.

Categories	***No of training images***	***No of validation images***	***No of testing images***
50 Rials	168	56	56
100 Rials	168	56	56
200 Rials	168	56	56
250 Rials	168	56	56
500 Rials	168	56	56
1000 Rials	168	56	56

3.3.2 Image Format

Dataset was collected from different cameras with different sizes and (JPEG) format.

3.3.3 Data Augmentation

we improved the performance of the model by augmenting the data without collecting new data. we utilized five augmentation strategies to generate new training sets, (Rotation, width shift, height shift, brightness, and zoom). Rotation augmentations are done by rotating the image right or left on an axis of 45°, shifting and zooming images are done by a range of 0.2, and increasing and decreasing brightness work by 0.6 to 1.3. we set the five augmentation strategies to work together to produce images from the original image are shown in Figure 3.3

Figure 3. 3: sample of how augmentation work

After doing augmentation on the whole dataset, the dataset became 10080 images is the original dataset plus new data 5 images per one image.

3.4 Preprocessing

3.4.1 Image Resize

One of the important things in the data preprocessing was to resize the images as the images were of various sizes, the images were resized to 224 by 224 Pixels, and All images were normalized to ImageNet standards to use on all architecture that takes input image 224 by 224 Pixels.

3.4.2 Normalization

All images were normalized according to the pre-trained model standards the input pixel values are scaled between 0 and 1.

3.5 CNN Models

We have used different Architectures to do some experiments and testing:

1. ResNet
2. NasaNetLarge
3. NASNetMobile
4. InceptionResNetV2
5. Xception
6. MobileNet
7. DenseNet121
8. EfficientNetV2

We found three versions (MobileNet, DenseNet121, and EfficientNetV2) that are the best and most suitable.

3.5.1 Fine Tuning

All layers of the pre-trained CNNs are retained as a fixed feature extractor for our dataset, just the last layer in the pre-trained CNN is replaced with the output layer that can classify our targeted 6 classes. All layers have RELU activations except the output layer uses Softmax activation.

3.5.2 Training the Models

We training all models with 20 epochs, using a batch size of 10, Learning Rate of 0.0001, Adam optimizer and dropout rate 0.001 during the training, the model will iterate over batches of the training set. For each batch, gradients will be computed and updates will be made to the weights of the network automatically. Training is usually run until the loss converges to a constant.

This is something that needs to be mentioned. In this training module, enhanced images were used as inputs by data augmentation; the training wasn't based on the original dataset images. We used the following output classes as coding: [0: 100RY, 1: 1000RY, 2: 200RY, 3: 250RY, 4: 50RY, 5: 500RY].

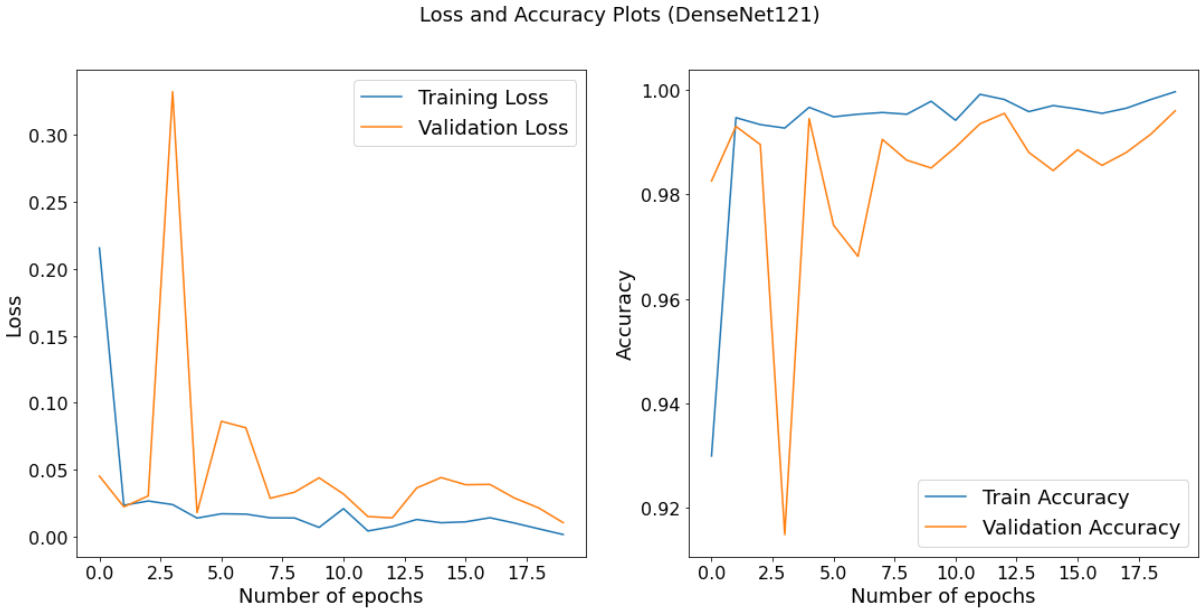
Based on our experiments, we choose three of the best pre-trained architectures. A description of the training behavior will be provided for each one. The number of parameters and number of layers for all architecture are shown on Table 3.3.

Table 3. 3: Number of parameters and layers.

Model	***Total parameters***	***Non-Trainable parameters***	***Number of Layers***
MobileNet	4,237,982	21,888	89
DenseNet121	7,043,654	83,648	429
EfficinetNetV2	12,939,844	109,216	385


Loss and Accuracy Plots (MobileNet)
Training Loss
Validation Loss
Loss
Number of epochs
Train Accuracy
Validation Accuracy
Accuracy
Number of epochs

(a)


Loss and Accuracy Plots (DenseNet121)
Training Loss
Validation Loss
Loss
Number of epochs
Train Accuracy
Validation Accuracy
Accuracy
Number of epochs

(b)

(c)

Figure 3. 4: Accuracy and loss for Training and validation for (a) MobileNet, (b) DenseNet121 and (c) EfficinetNetV2

3.5.3 Testing the Models

After training and evaluating the model on the validation, the models are tested using 2016 images, after doing Data Augmentation on the original as shown above in table 3.2.

For Testing the models, we use two different ways:

1. using Model.evaluate() function

2. using confusion matrix and another performance matrix such as Precision, Recall, Accuracy, and F1 Score.

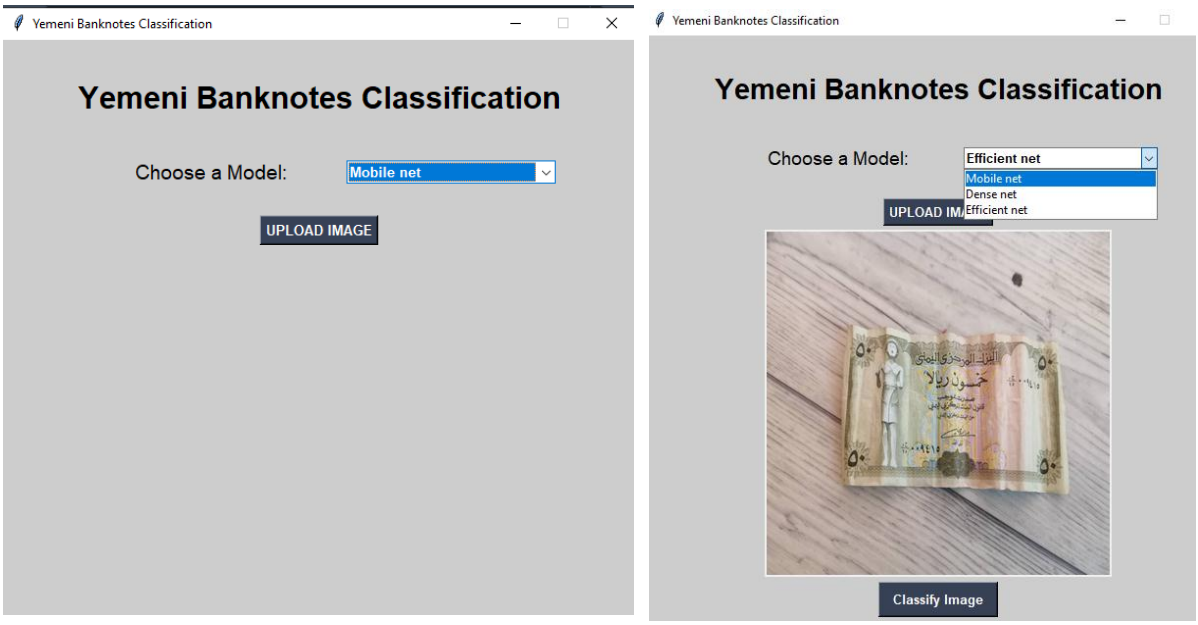
3.7 User Interface

The graphical user interface for the developed Yemeni Banknote Classifier (YBC) system build using a python programming language with Tkinter Library, the GUI is shown in Figure 3.5.

The system starts, when an input image is selected using the Upload Image button, the system allows choosing the model from the different

models using a drop-down list that contains 3 different models MobileNet, DenseNet, and EfficientNet.

after the image has been uploaded and the model has been chosen then to predict the class of the input image, the button Classify image is clicked. After that Another interface shows the output result that includes a name for the selected model, a table for probability for each class, and the class with the highest probability among all classes.



(a) (b)

Figure 3. 5: Graphical User interface of YBC System (a) before Upload Image (b) after upload the desired image and select Model

Chapter 4

RESULTS AND DISCUSSION

4.1 Introduction

This chapter will provide details about training and testing the three models that used in our project, all results achieved by the proposed YCC system during training and testing will be discussed, and for all trained Models different performance matrices will be used to evaluate the models

4.2 Experiment

we worked on different models using two different approaches, the first with 6 classes and the second with 12 classes in the second approach the back and front of currency paper were considered as different classes, also we work on a dataset of 1680 images that have been collected and build from scratch. as a result, we got a better result when the 6 classes approach is used so we implement it as a final version of the YCC proposed System.

4.3 Results

4.3.1 Training Results

We trained our custom models on the training and validation dataset using 6048 images for training and 2016 for validation for a total of 20 epochs, the results were as shown in table 4.1:

Table 4. 1: Training loss and validation loss and accuracy for all models

Model		***Training accuracy***	***training loss***	***Validation accuracy***	***Validation loss***
DenseNet121	**First epoch**	0.9566	0.1387	0.9897	0.0320
	Last epoch	1.000	0.00024	0.9976	0.0074
EfficientNetV2	**First epoch**	0.8945	0.3226	0.9905	0.0353
	Last epoch	0.9938	0.0088	0.9950	0.012
MobileNet	**First epoch**	0.8223	0.6500	0.9169	0.2898
	Last epoch	0.9996	0.0278	0.9836	0.0714

The training behavior for the three models on 20 epochs are shown in the Figure 4.1

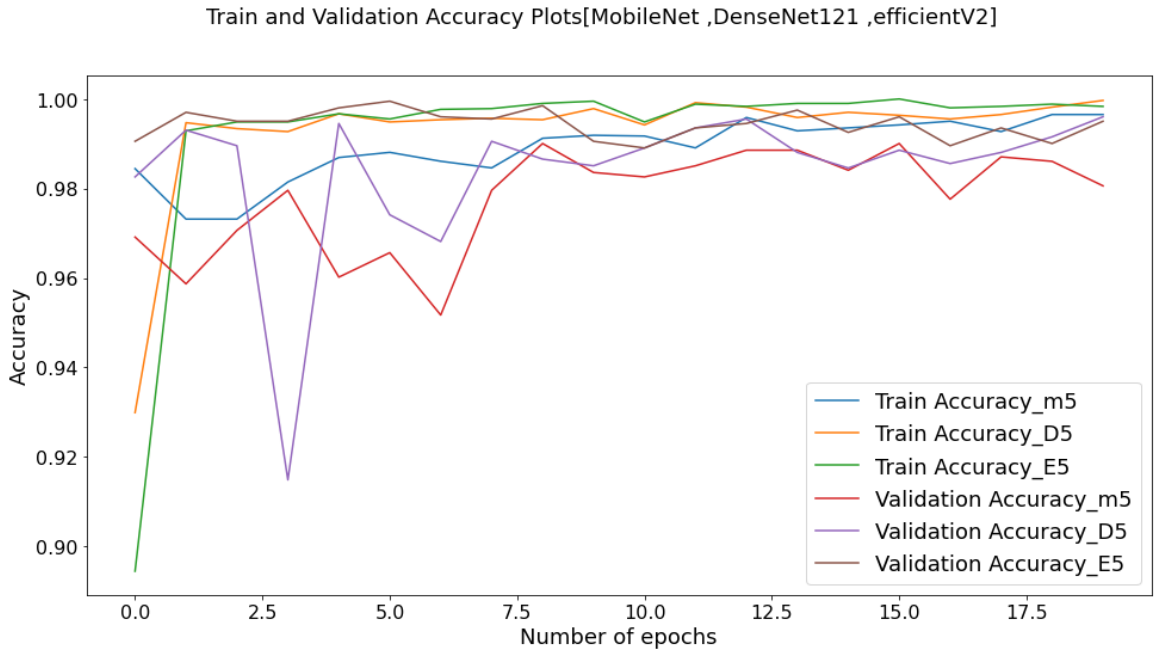
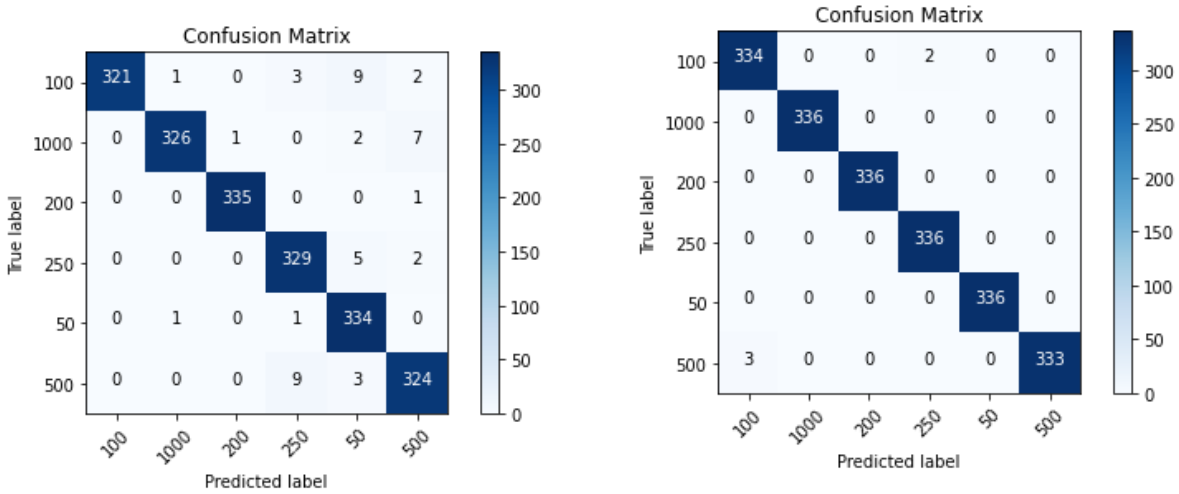


Figure 4. 1: Validation Accuracy and Training Accuracy for DenseNet121, MobileNet and EfficeintNetV2

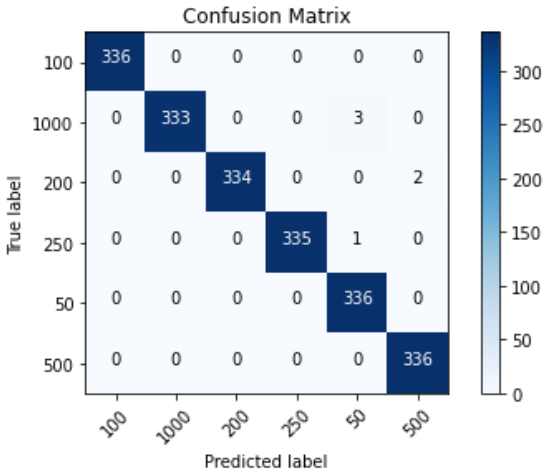
Based on the previous results, we got the best result from DenseNet121 that archived training accuracy of 100% and validation accuracy of 99.76%.

4.3.2 Testing Results

In order to provide more details about the performance of the trained models, we provide the confusion matrices on testing data of each model. As shown in Figure 4.2



(a) (b)



(c)

Figure 4. 2: Confusion Matrix on testing data of (a)MobileNet (b)DenseNet121 (c)EfficintNetV2

From Confusion Matrix, the performance of classifiers is analyzed with 2016 images, which consist of 336 images for each class by using the performance of the classifiers in terms of precision, recall, f1-score, and Accuracy of classification as shown in Table 4.2

Table 4. 2: Performance analysis of all models

Model	***Accuracy***	***Recall***	***Precision***	***F1_Score***
DenseNet121	99.751%	99.751%	99.753%	0.9975
EfficientNetV2	99.702%	99.702%	99.705%	0.9970
MobileNet	97.668%	99.668%	97.722%	0.9767

According to Table 4.2 and Figure 4.2, DenseNet121 classified the image with the highest metrics. The confusion matrix also clearly shows the perfectness of the DenseNet121 in classifying the testing dataset. Similarly, by using evaluate () method to evaluate models we got that DenseNet121 achieved 99.751% accuracy which is the best among all models as shown in Figure 4.3

```
MobileNet Model

202/202 [==============================] - 21s 98ms/step - loss: 0.0811 - accuracy: 0.9767
Accuracy is  97.66865372657776
loss is  8.109357208013535

EfficientNet Model

202/202 [==============================] - 21s 103ms/step - loss: 0.0084 - accuracy: 0.9970
Accuracy is  99.70238208770752
loss is  0.8419349789619446

DenseNet Model

202/202 [==============================] - 21s 105ms/step - loss: 0.0161 - accuracy: 0.9975
Accuracy is  99.7519850730896
loss is  1.6137193888425827
```

Figure 4. 3: Comparison of MobileNet, DenseNet121, and EfficintNetV2 testing Accuracy using the method evaluate ()

To make sure which is the best model we test all models on a new dataset of 241 images,30 images for each class and the result obtained was shown in Table 4.3

Table 4. 3: Performance analysis of all models

Model	***No. of image classified Successfully***	***No. of image classified wrong***
DenseNet121	241	0
EfficientNetV2	241	0
MobileNet	234	7

Results from the experiments clearly prove the viability of the approach proposed. To demonstrate the system's success, a picture of the Yemen paper currency was taken and classified using the GUI of the YCC proposed System and the result was shown In Figure 4.3

Figure 4. 4: The result for testing an image of a 50 RY using DenseNet121.

Chapter 5

CONCLUSION AND FUTURE WORK

5.1 Conclusion

In this project, we have built a system that is capable of Recognizing the type of YB using DL. Firstly, we have collected a large number of various pictures of Yemeni banknotes. Secondly, to solve overfitting problem, we used data augmentation to increase the data set.

Third, for training Tow different Approaches used, 12 class and 6 class we got a better result when using 6 class. Different DL model have been used among them we choose the best Three: MobileNet, DenseNet121, and EffiecientNetV2, and achieved the following accuracies in the testing phase: 97.55%,99.75%, and 99.70% respectively.

We concluded that DL was the best high tech which gave us enormous results to identify the YB.

5.2 Recommendations and Future Work

To develop the system and improve its performance the following future work are suggested:

1. Expand the project capabilities to be able to identify Yemeni and foreign Banknotes.
2. Identifying more than one Banknote simultaneously in one image.
3. Applying this system in the real world as hardware at many places such as banks, Automatic Teller Machines, and as assistive devices for the visually impaired.
4. Expanding the system to be capable of identifying fake Banknotes and the real ones.

References

[1] P. Neelambika, K. Bhagyashree, K. Keerthana, and B. Anusha,"Currency Recognition System." International Journal of Research in Engineering, Science and Management, 2020.

[2] Althafiri, E., Sarfraz, M.I., & Alfarras, M. (2012). Bahraini Paper Currency Recognition. *Journal of Advanced Computer Science and Technology, 2*.

[3] Almisreb, Ali & Saleh, Mohd.A.. (2019). Transfer Learning Utilization for Banknote Recognition: A Comparative Study Based on Bosnian Currency. Southeast Europe Journal of Soft Computing. 8. 10.21533/scjournal. v8i1.172.

[4] Murad, H., Tripto, N. I., & Ali, M. E. (2019, January). Developing a bangla currency recognizer for visually impaired people. In Proceedings of the Tenth International Conference on Information and Communication Technologies and Development (pp. 1-5).

[5] Saeed, A., Ba Alawi, A. and Hassan, A., (2021). Yemeni Banknote Recognition Model based on Convolution Neural Networks. 2021 1st International Conference on Emerging Smart Technologies and Applications (eSmarTA),

[6] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., ... & Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861.

[7] Huang, G., Liu, Z., van der Maaten, L., & Weinberger, K. Q. (2018). Densely connected convolutional networks. arXiv 2016. arXiv preprint arXiv:1608.06993, 1608.

[8] Tan, Mingxing & Le, Quoc. (2021). EfficientNetV2: Smaller Models and Faster Training.

[9] Li, Z., Liu, F., Yang, W., Peng, S., & Zhou, J. (2021). A survey of convolutional neural networks: analysis, applications, and prospects. IEEE transactions on neural networks and learning systems.

[10] Traore, B., Kamsu-Foguem, B. and Tangara, F., 2018. Deep convolution neural network for image recognition. Ecological Informatics, 48, pp.257-268.

[11] Tripathi, M. (2021). Analysis of Convolutional Neural Network based Image Classification Techniques. Journal of Innovative Image Processing, 3(2), 100–117. https://doi.org/10.36548/jiip.2021.2.003

[12] Rahman, Tawsifur & Chowdhury, Muhammad & Khandakar, Amith & Islam, Khandakar & Islam, Khandaker & Mahbub, Zaid & Kadir, Muhammad & Kashem, Saad. (2020). Transfer Learning with Deep Convolutional Neural Network (CNN) for Pneumonia Detection Using Chest X-Ray. Applied Sciences. 10. 3233. 10.3390/app10093233.

[13] Ray, Susmita. (2019). A Quick Review of Machine Learning Algorithms. 35-39. 10.1109/COMITCon.2019.8862451

[14] O'Shea, K., & Nash, R. (2015). An Introduction to Convolutional Neural Networks. arXiv e-prints, page. arXiv preprint arXiv:1511.08458.

[15] Khan, Asifullah & Sohail, Anabia & Zahoora, Umme & Saeed, Aqsa. (2020). A Survey of the Recent Architectures of Deep Convolutional Neural Networks. Artificial Intelligence Review. 53.

[16] Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., ... & He, Q. (2020). A comprehensive survey on transfer learning. Proceedings of the IEEE, 109(1), 43-76.

[17] LeCun, Yann & Bengio, Y. & Hinton, Geoffrey. (2015). Deep Learning. Nature. 521. 436-44. 10.1038/nature14539.

[18] Tan, Chuanqi & Sun, Fuchun & Kong, Tao & Zhang, Wenchang & Yang, Chao & Liu, Chunfang. (2018). A Survey on Deep Transfer Learning.

[19] Mikołajczyk, Agnieszka & Grochowski, Michał. (2018). Data augmentation for improving deep learning in image classification problem. 117-122. 10.1109/IIPHDW.2018.8388338.

[20] Guo, Yanming & Liu, Yu & Oerlemans, Ard & Lao, Song-Yang & Wu, Song & Lew, Michael. (2015). Deep learning for visual understanding: A review. Neurocomputing. 187. 10.1016/j.neucom.2015.09.116.

[21] Jogin, Manjunath & Mohana, Mohana & Madhulika, M & Divya, G & Meghana, R & Apoorva, S. (2018). Feature Extraction using Convolution Neural Networks (CNN) and Deep Learning. 2319-2323. 10.1109/RTEICT42901.2018.9012507.

[22] Shorten, C. and Khoshgoftaar, T., 2019. A survey on Image Data Augmentation for Deep Learning. Journal of Big Data, 6(1).

[23] Chu, Hongyang & Liao, Xinwei & Dong, Peng & Zhao, Xiaoliang & Zou, Jiandong. (2019). An Automatic Classification Method of Well Testing Plot Based on Convolutional Neural Network (CNN). Energies. 12. 2846. 10.3390/en12152846.

[24] Hussain, Mahbub & Bird, Jordan & Faria, Diego. (2018). A Study on CNN Transfer Learning for Image Classification.

[25] Barhoom, Alaa M. A. & Abu-Naser, Samy S. (2022). Diagnosis of Pneumonia Using Deep Learning. International Journal of Academic Engineering Research (IJAER) 6 (2):48-68.

[26] Kamilaris, Andreas & Prenafeta Boldú, Francesc. (2018). Deep Learning in Agriculture: A Survey. Computers and Electronics in Agriculture. 147. 10.1016/j.compag.2018.02.01

[27] Kingma, D. P. (2015). Adam: A Method for Stochastic Optimization/Diederik P. Kingma, Jimmy Ba, URL: https://arxiv. org/abs/1412.6980.

[28] Gholamalinezhad, H., & Khosravi, H. (2020). Pooling Methods in Deep Neural Networks, a Review. *ArXiv, abs/2009.07485*.

[29] Madry, Aleksander & Santurkar, Shibani & Tsipras, Dimitris & Ilyas, Andrew. (2018). How does batch normalization help optimization?

[30] PATRO, S GOPAL & Sahu, Dr-Kishore Kumar. (2015). Normalization: A Preprocessing Stage. IARJSET. 10.17148/IARJSET.2015.2305.

[31] Wang, Meiqi & Lu, Siyuan & Danyang, Zhu & Lin, Jun & Wang, Zhongfeng. (2018). A High-Speed and Low-Complexity Architecture for Softmax Function in Deep Learning. 223-226. 10.1109/APCCAS.2018.8605654.

[32] Lillicrap, T. P., Santoro, A., Marris, L., Akerman, C. J., & Hinton, G. (2020). Backpropagation and the brain. Nature Reviews Neuroscience, 21(6), 335-346.