SOLID TRASH SEGREGATION SYSTEM USING CONVOLUTIONAL NEURAL NETWORK

MOH MOH THET AUNG

M.C.Tech. **DECEMBER 2022**

SOLID TRASH SEGREGATION SYSTEM USING CONVOLUTIONAL NEURAL NETWORK

BY

MOH MOH THET AUNG

B.C.Tech.

A Dissertation Submitted in Partial Fulfillment of the Requirements for the Degree of

Master of Computer Technology

(M.C.Tech.)

University of Computer Studies, Yangon

DECEMBER 2022

ACKNOWLEDGEMENTS

I would like to give my sincere thanks to people who have assisted and guided me during the Master study.

First and foremost, I especially thank Dr. Mie Mie Khin, Rector, University of Computer Studies, Yangon, for overall supporting my thesis. I greatly appreciate the help. I have received from her.

I owe my special gratitude to Dr. Htar Htar Lwin, Pro-rector, Faculty of Computer Systems and Technologies, University of Computer Studies, Yangon, for giving me a lot of suggestions, encouragement and motivation on my thesis work to reach the goal.

I would like to express my deepest gratitude to my supervisor, Dr. Amy Tun, Professor, the University of Computer Studies, Yangon, for her tremendous support and guidance during my master study.

I would also like to express my respectful gratitude to Daw Mya Hnin Mon, Associated Professor of English Department, for her valuable support and editing the correct usage of language in my thesis.

I would also like to thank all my teachers for mentoring, encouraging, and recommending the dissertation. I am also grateful to all talented and friendly colleagues from Master studies for their friendship during my graduate student life.

Finally, I want to express my sincerest and deepest gratitude to my father, mother and my family for their unconditional love, support, and encouragement. I would not have been where I am today without their endless love and tremendous support. My dissertation is dedicated to them.

ABSTRACT

Environmental protection has long placed a high priority on garbage sorting. Due to the cities' rapid population increase and the massive amount of waste it generates, urban regions are experiencing difficulties with their waste management systems. It is important to have an advanced waste classification system to classify a variety of solid waste materials. One of the most important steps of waste management is the segregation of the waste into the different types of components. In our country, the waste segregation process is normally done manually by hand-picking. To simplify the procedure, a trash segregation system is proposed. Waste material classification system, which is created by using the 50-layer residual network (ResNet-50) Convolutional Neural Network model which is used to categorize the waste into different types such as glass, metal, paper, and plastic, cardboard, trash. The proposed system is tested on the Kaggle Garbage dataset is able to accomplish a high accuracy.

Firstly, the dataset is split into train, valid and test. Secondly, the simple CNN model and CNN based Resnet 50 model are build and use to train the training data. Before training, the image data are needed to resize and process by using data augmentation methods. Thirdly, the trained prediction model is used to classify the test data. Finally, the testing data are used to evaluate accuracy of model performance. The image with its label is coming out as output that are showed. The system is implemented by python language on google colaboratory.

TABLE OF CONTENTS

LIST OF FIGURES

Figure **Page**

LIST OF TABLES

Table **Page**

LIST OF EQUATIONS

CHAPTER 1

INTRODUCTION

This chapter describe the introduction of the system. Moreover, the main objectives of the system are described. The organization of the system are introduced in this chapter.

1.1 Introduction

As the increasing in population, there are a lot of challenges have been taken place in the waste management system. Waste is generated from various of places, such as industries, households, public areas and hospitals. According to 2012 World Bank estimations, solid waste generation was 5,616 tones/day in Myanmar. The waste disposing was expected to reach about 21,012 tones/day by 2025 (World Bank, 2015). However, that estimations show that Myanmar already produced approximately 20,000 tones/day of solid waste as of 2017 (Netherland Enterprise Agency, 2017) [4]. Therefore, waste generated rating increases slightly day by day.

In Myanmar, waste collectors and waste dealers are carrying out recycling activities mostly by the informal sector. These garbage collectors gather recyclables from homes, streets, businesses, and final disposal places, such as cardboard, magazines, aluminum, plastic bottles, tin cans, and glass, and then they sell those commodities to waste dealers. The recycling sector buys them in bulk from waste dealers who clean, sort, store, and sell them for both domestic and international use. Currently, information on recycling volumes, ratios, and the number of recycling factories in Myanmar cities is not available in a precise or trustworthy manner [4]. So, waste is needed to classify with the help of technologies.

To classify correctly the trash, automatic techniques are proposed. In the most recent years, Convolutional Neural Network (CNN) is the most suitable image classification technique wherefrom the segmented objects no handcrafted features are extracted. Convolutional Neural Network (CNN) is the class of deep learning that is the preparation of numerous layers through the computational models to learn the representations of data with the abstraction of numerous layers. Many Convolutional

Neural Network (CNN) based classification and detection models are used mainly for classification and detection.

This thesis presents a trash classification system that detects and classify the different types of waste. For the classification purpose, the simple CNN model and Resnet50 has been applied to the system. This trash classification system using Convolutional Neural Network (CNN) based model can improve the performance of waste categorization correctly, saving time and human effort, reducing the trash land-fill in the world.

As a related work, "Classification of Trash for Recyclability Status", Support Vector Machine (SVM) is used to classify waste images, they proved that SVM is better performance than CNN [17]. The dataset was divided into only training 70% and testing 30%. 63% of testing accuracy is achieving in using SVM with scale invariant feature transform. In this experiment, CNN (AlexNet) was achieving accuracy 22% when testing data. According to this result, why CNN is less in accuracy. In this paper, the data is trained only 60 epochs and the learning rates are changing in every 5 epochs.

In another related, namely waste classification using CNN algorithm which is one of the research projects of numerous classifications [18]. In this study, it can be only used for classifying garbage based on their materials, organic and recyclable, thousands of data sets. The dataset comprises two components: a training set and a testing set. In this experiment, ReLU, Sigmoid, and Tanh activation functions are used and compare and quantify if a classification was successful or failed. They trained the data by using 50 epochs and 0.001 learning rate. According to their confusion matrix, the accuracy was achieving about 85.5%. Accuracy is not enough for classification because small dataset only 800 training and 200 testing images are used. Moreover, the classification of items is only two class.

1.2 Objectives

The main objectives of the thesis are as follows:

- To reduce time and effort for classifying types of waste
- To provide recycling opportunities
- To minimize the waste and ensure reduction in landfill space for final disposal
- To realize solid trash segregation through deep learning techniques

1.3 Organization

There are five chapters in this thesis. The introduction to the garbage segregation system is presented in Chapter 1. Additionally, the goals and structure of the thesis are described along with any connected publications. The background theory is discussed in Chapter 2. Artificial Neural Network and its workflow are also described. And all necessary technologies related to image preprocessing and classification are thoroughly examined. Convolutional neural network (CNN) overview and contemporary CNN-based models are thoroughly presented. The design and implementation of the thesis work are Described in Chapter 3. This chapter explains the overall system design and step-by-step implementations. Results of the experimental results are displayed and analyzed in Chapter 4. This thesis is concluded in Chapter 5, which also discusses the advantages, limitations, and future extensions of the suggested system.

CHAPTER 2

BACKGROUND THEORY

In this chapter, the technological theories are described that are needed for this thesis. General knowledge of artificial neural network and how the working flow of Artificial Neural Network (ANN) are also explained in detail. Moreover, the learning methods and training algorithms are also described. And then, this chapter consists of the various layers of convolutional neural network and modern architectures of CNN.

2.1 Biological and Artificial Concept

Nowadays, developers are trying to use Artificial Neural Network in many places of business, such as information technologies, robotics, medical, industrial and so on. Artificial intelligent is represented as the computer system which is the replication of human intelligence functions by machines. Artificial intelligent technology is the most useful system in building AI systems. Because, it is associated with the related endeavor of utilizing computers to analyze human intelligence. On the other hand, ANN is the transformation of human intelligent into mathematical model that are used in computations system [15].

2.1.1 Biological Neuron

The brain's nerve cells, or neurons, are referred to by the term neural, which is derived from the (animal) human nervous system. Four sections are made up the nerve cells in the human brain:

- Dendrites: Signals from other neurons are received there.
- Cell body: It’s the total of all incoming signals to produce the input.
- Synapses: These are the points at which neurons connect with one another.
- Axon: When the total of a neuron's signals hits a particular threshold, it transmitting the signal to neighboring neurons through the axon.

A Neural Network (NN) is a broad term for a network was made up of billions of neurons and many connections.

2.1.2 Artificial Neuron

Artificial neurons are computer simulations of biological neural networks that take one or more inputs and combine them using an activation function to produce an output. A synthetic version of neurons was found in the human brain. When determining the weighted sum of the inputs, it was provided the value of the activation function (x). Figure 2.1 shows an artificial neuron's basic mechanism of an example of a particular class of machine learning approaches is the artificial neural network (ANN). The word is derived from the biological relationships between neurons in the human brain. ANNs are data-driven algorithms that attempt to uncover latent functional relationships despite the absence of explicit physics by learning from a dataset of examples. Although they have many distinct topologies which have been built on the same fundamental building component, the neuron. Processing components called neurons are grouped together with various connections. Every neuron's input is accompanied with a weight and a bias; through an activation function, data is transmitted to the next level of the structure. Numerous neurons in an artificial neural network are interconnected in intricate ways to solve linear or non-linear problems. The topology of a feedforward neural network was chosen to forecast hematic parameters. Neurons are stacked in layers in a feedforward network, with the first layer receiving inputs. And then, one or more middle layers known as hidden layers because they are isolated from the outside environment, and the last layer producing outputs [15]. In the Figure, there are no loops between neurons and the data; it always flows in one direction.

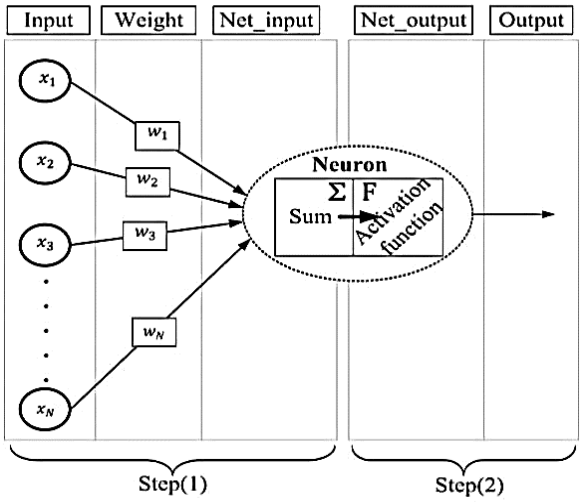


Figure 2.1 Working Flow of an Artificial Neuron

2.1.3 Activation Functions

A neuron's activation status is determined by an activation function. According to this, it will perform some straightforward computation to decide whether the input from the neuron to the network is important or not for the prediction process. The activation function's function is to enable non-linearity in an artificial neural network and produce output from a set of input values provided to a layer. An activation function in a neural network describes how the output from one or more nodes in a layer is produced by using the weighted sum of the input. The term "transfer function" is sometimes used to describe the activation function. If the output range of the activation function is constrained, it is referred to as a "squashing function." The term "nonlinearity" describes how many activation functions in a layer or network design.

Different activation functions might be applied to different parts of the model, and they could significantly affect the neural network's capabilities and efficacy. Although networks are designed to use the same activating function for every node in a layer, the activating process is applied inside or after each network node's internal processing.

Three different types of layers make up a network: input layers, which accept unprocessed domain data; hidden layers, which take inputs from one layer and relay outputs to another; and output layers, which offer predictions. Typically, all buried layers share the same activation function. Depending on the type of prediction required by the model, the activation function of the output layer is frequently different from that of the hidden layers. The first-order derivative may usually be determined for a specific input value since activation functions are typically differentiable. Popular types of activation functions are Linear, Binary Step, Tanh, ReLU, Parameterised ReLU, Swish, Softmax, Sigmoid.

2.2 Artificial Neural Networks

Artificial neural networks (ANNs) and simulated neural networks (SNNs), which are a subset of machine learning, are the neural network types used in deep learning approaches. The human brain was served as the model for both their name and structure. Artificial neural networks algorithms are built by using a model of a human neuron (ANN). The human brain contains millions of neurons. To send and process signals, electrical and chemical impulses are used. The particular structures that

structure that link these neurons together are called synapses. Synapses enable neurons to transmit additional signals. Numerous simulated neurons are combined to construct neural networks.

An example of a way for processing data is an artificial neural network. It's functions similarly to how the human brain interprets data. A network of connected processing units called an ANN works together to process data. Additionally, they get useful outcomes from it. A neural network is not just useful for categorization. Regression of continuous target attributes is another application for it. Almost all commercial applications and businesses employ these technologies. Finding answers to complex problems in a range of industries, including pattern recognition, image recognition, time series forecasting, modeling, audio transcription, data analysis, check processing, education and health care, weather forecasting, and signal processing, is their main goal.

2.2.1 Structure of Artificial Neural Network

The usual structure of an artificial neural network consists of layers. The "activation function" is one of many interconnected "nodes" that was made up a layer. The three layers of a neural network are the input layer, output layer, and hidden layer or layers. There must be connections between the input layer nodes, the hidden layer nodes, each hidden layer node, and the output layer nodes [15]. This structure can be seen in the Figure 2.2.

Input Layer: The artificial neurons that receive information from the external environment are found in the input layers. Raw data can be provided to the network through input unit activity. The number of input nodes and the number of explanatory variables is equal in the input layers. The patterns are supplied to the network at the "input layer," which communicates with one or more "hidden layers." Because the input layer's nodes are passived, they have no effect on the results. They duplicate a single input value from their inputs to each of their outputs. It transmits copies of each value from the input layer to every hidden node.

Hidden Layers: A cluster of neurons in the hidden layer performs all computations on the input layer. Any number of hidden layers are allowed in a neural net. There is only one hidden layer in the most basic network. The real processing is done at the hidden layer using a series of weighted connections. A neural net can have any number of hidden layers. There is only one hidden layer in some of the most basic

network. Using a number of connection weights, the hidden layer performs the actual processing. The weighted inputs are then summed to create a single number. The Hidden Layer controls how each hidden unit behaves. The weights and actions of the input units are determined by the connections between the input and hidden units.

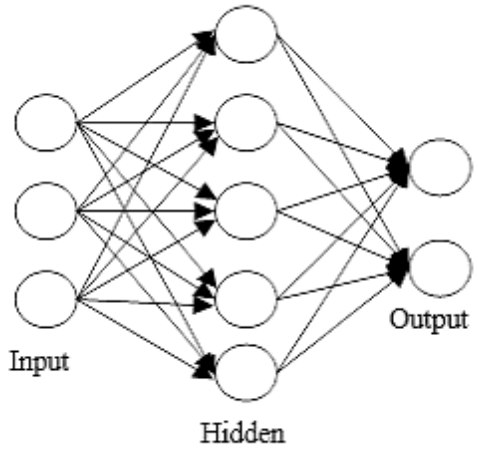


Figure 2.2 Architecture of ANN

Output Layer: The "output layer" is then connected to the concealed layers. The output layer takes connections from input layer or hidden layers. The output values are produced with the prediction made for the response variable. Commonly, categorization issues have a single output node. In order to produce the output values, the active nodes of the output layer aggregate and change the data. Correct weight selection is necessary for the neural network to perform useful data manipulation. This differs from traditional information processing. The weights between the hidden layer and the output units and the activity of the hidden neurons both have an impact on how the output units behave.

2.2.2 Work Flows of Artificial Neural Networks

The ideal way to conceptualize artificial neural networks is as weighted directed graphs, where the artificial neurons serve as the nodes, and the directed edges have weights that described the link between neuron inputs and outputs. The Artificial Neural Network takes input signals from the outside environment in the shape of patterns and pictures in the form of vectors. The notation y(n) is then used for each n inputs in order to numerically express these inputs. The weights that have been assigned to each input are then multiplied by each input; these weights are the specifics that the artificial neural networks use to handle a particular issue. These weights typically describe the degree

of connectivity between the neurons in the ANN. Within the computing unit, all of the weighted inputs are pooled (yet another artificial neuron).

If the weighted sum is equal to zero, the output is biased to create a non-zero value; otherwise, the system's reaction is scaled up. The constant input value of bias is one, and its weight is one. The range of the weighted input total in this example is 0 to positive infinity. A particular threshold value is standardized in order to keep the response within the constraints of the intended output. The activation function receives the sum of the weighted inputs after that. The activation function is a collection of transfer functions that is typically employed to generate the desired output. This explanation can be seen in Figure 2.3. Although there are numerous other kinds of activation functions, linear or non-linear sets of functions are by far the most common. The most well-liked sets of activation functions include Tan hyperbolic Sigmoidal, Binary, and Sigmoidal (linear) (non-linear).

The input node converts the data into a numerical format. Each node receives a number that corresponds to an activation value. The number increases as the activation becomes stronger. Following node receives the activation value based on weights and the activation function. The weighted sum is calculated and updated by each node using the transfer function (activation function). The action is subsequently performed via the activation function. It is the unique function of this neuron. The neuron must then decide whether to send the signal or not. The weights are modified by the ANN, which affects the signal extension.

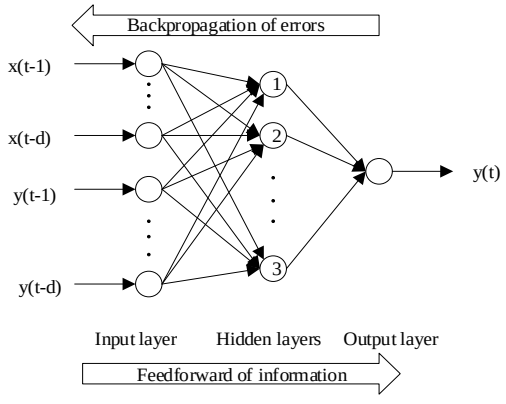


Figure 2.3 Work Flows of Artificial Neural Network

Up until it reaches the destination node, activation travels throughout the network. It makes natural that the data would be shared by the output layer. The network analyzes the output and intended outputs using the cost function. The cost function is

the difference in values between actual and projected values. Cost function's size determines how closely the outcome resembles the anticipated outcome.

2.2.3 Types of Artificial Neural Network

Feedforward Neural Network: Information only goes in one direction in a feedforward neural network. To put it another way, information travels from the input layer to the hidden layer, then to the output layer.

There are no loops for feedback. In these networks, the output of one layer does not influence that of another layer. Simple neural networks that directly connect inputs and outputs are called feed-forward neural networks [12]. This network is shown in the Figure 2.4. They have fixed inputs and outputs. The most prevalent uses include pattern production, pattern recognition, and classification.

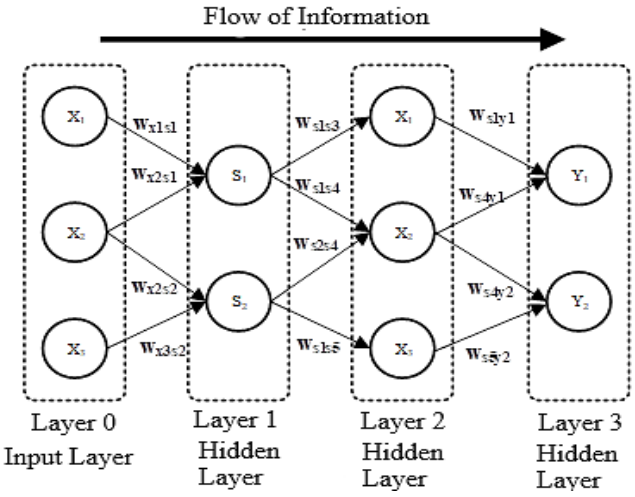


Figure 2.4 Feedforward Network

A feature of feedback neural networks is feedback loops. For instance, memory retrieval commonly makes use of recurrent neural networks. These networks operate best with sequential or time-dependent data. The feedback loops of recurrent neural networks (RNNs) define them. In a content-addressable memory, they are employed. The feedback loop of the network is presented by the Figure 2.5. Various fields of feedback and feedforward network are presented in Figure 2.6.

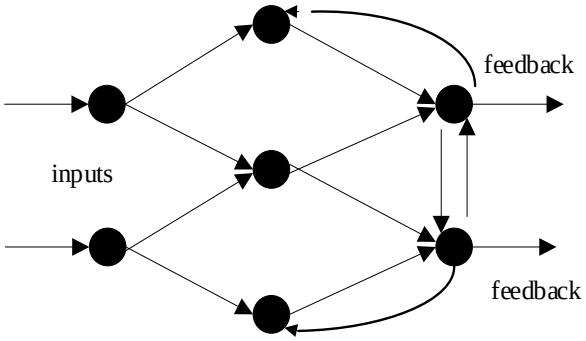


Figure 2.5 Architecture of Feedback Network

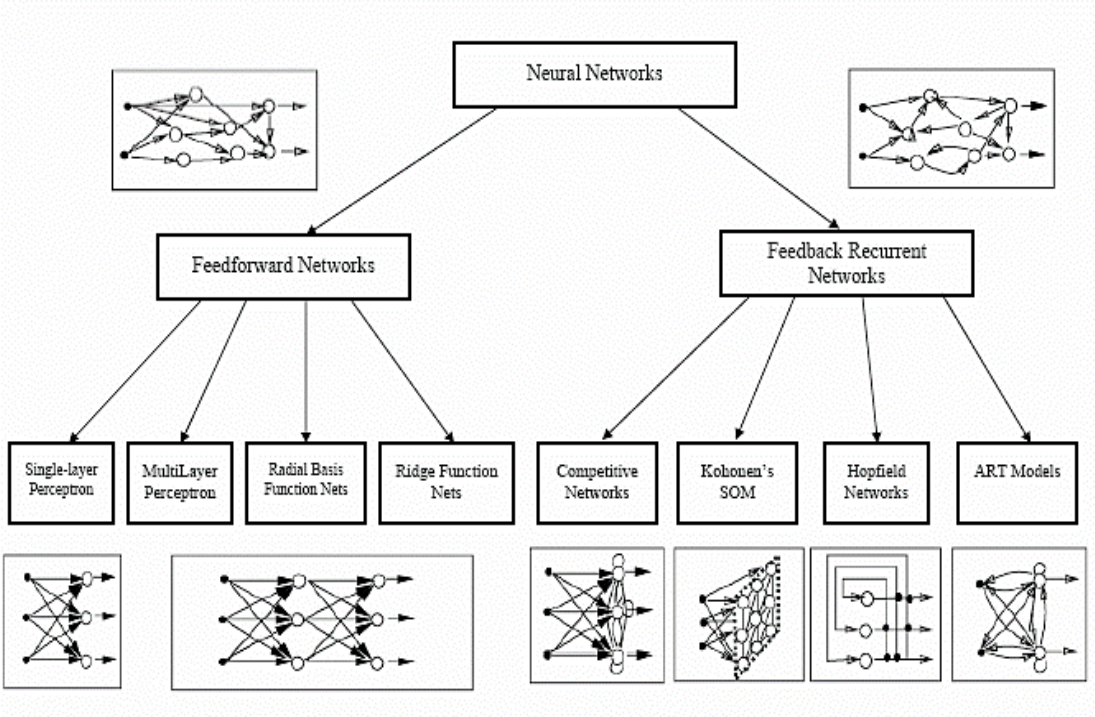


Figure 2.6 Sub-Fields of ANN

2.3 Deep Artificial Neural Network

The number of hidden layers is more than two and the neurons of these layers are so many, this kind of ANN is called Deep ANN. The name "Deep" is the definition of the including many hidden layers and neurons. Therefore, the main difference between simple ANN and deep ANN is depending on including of many hidden layers.

Deep ANN is used for classification or recognition of unstructured data or pattern shown in Figure 2.7. In this processing, it extracts high level features from these unstructured data. In this feature extracting, hierarchical feature extracting is used.

Hierarchical feature extraction is the extraction of features by using appearance of hierarchical to get features that used for final layer of classification.

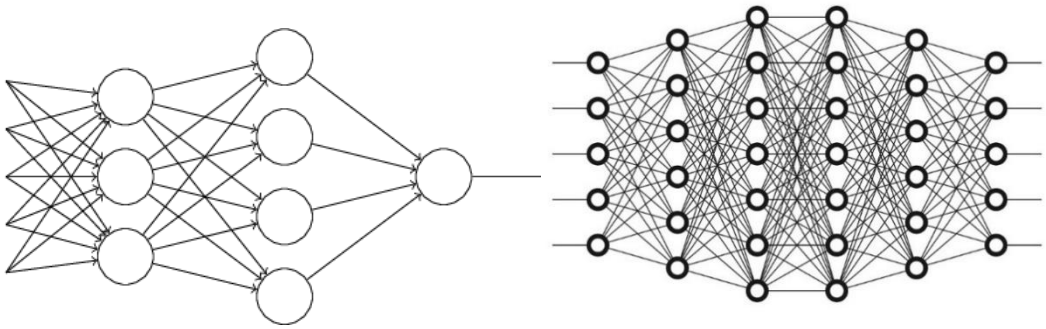


Figure 2.7 Architecture Simple ANN (Left) and Deep ANN (Right)

2.4 Types of Learning

For training artificial neural networks, there are many different methods. Each methods have their own advantages and disadvantages. Artificial neural networks learn when the weights of the network are modified using a learning algorithm. Three categories are used to group a neural network's learning processes. These three types of learning are supervised, aided, and reinforced [15]. Figure 2.8 shows the types of learning methods.

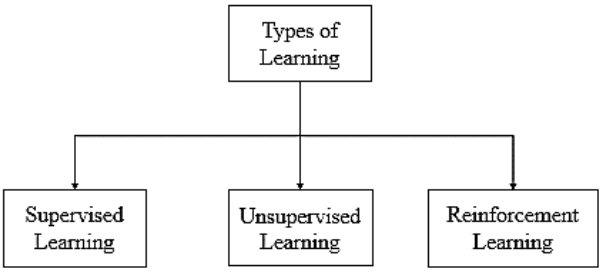


Figure 2.8 Types of Learning

2.4.1 Supervised Learning

Under the direction of a teacher, this type of instruction is provided. This learning curriculum is reliant. During supervised learning, the input vector is provided to the network, and after training, an ANN generates an output vector. The preferred output vector is contrasted with this one. If the actual output value differs from the

expected output vector, an error signal is generated. The weights are changed in accordance with this error signal until the desired output is obtained in Figure 2.9.

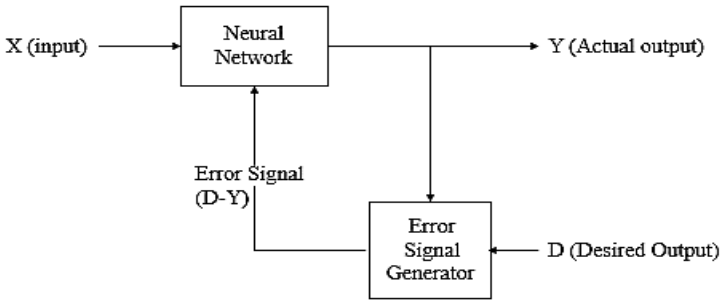


Figure 2.9 Flow of Supervised Process

2.4.2 Unsupervised Learning

Without the supervision or guidance, this type of training takes place. It is a separate learning procedure. During ANN training, input vectors of like types are merged to generate clusters in unsupervised learning shown in Figure 2.10. When a new input pattern is applied, the neural network responds with an output response that represents the class of the new input pattern.

There is no information in the environment about what the desired outcome should still be or whether it is correct. In order to perform this type of learning, the network must identify the connections between the input and output data as the patterns and attributes from the input data.

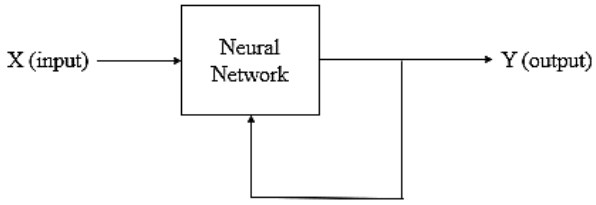


Figure 2.10 Flow of Unsupervised Process

2.4.3 Reinforcement Learning

This type of learning maybe much less input. This learning is comparable to supervised process. The result will give the network some feedback while it is being trained using reinforcement learning. The feedback, meanwhile, is more evaluative than informative, suggesting that there is not a teacher present, as in supervised learning.

Finally, this network is updating the weights to get future information when receiving input which is described in Figure 2.11.

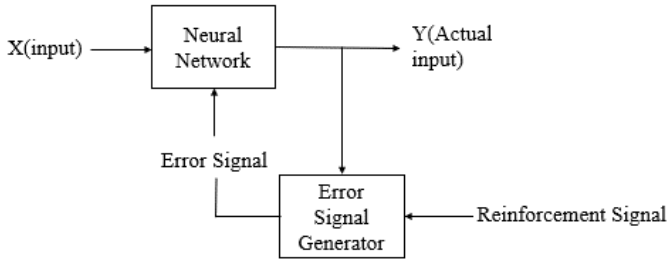


Figure 2.11 Flow of Reinforcement Process

2.5 Training Methods

By finding a decision function during training, the network's weights are updated. There are numerous variations of the training algorithms. Trying to predict which training algorithm will yield the greatest outcomes is challenging (Sundar et al., 2012). The algorithms alter the network weights and biases to properly transfer arbitrary inputs to outputs. Table 2.1 lists the several of applications used in each learning.

Table 2.1 The Training Algorithms

Learning Method	Application	Algorithms
Supervised Learning	Classification	-Naïve Bayes Classifier - Decision Trees - Support Vector Machines - K-Nearest Neighbors - Convolutional Neural Network
	Regression	- Linear Regression - Neural Network Regression - Support Vector Regression - Decision Tree Regression - Lasso Regression
Unsupervised Learning	Clustering	- K-mean clustering - Mean-shift clustering - Gaussian Mixture
Reinforcement Learning	Decision Making	- Q learning - R learning

2.5.1 Convolutional Neural Network

The classification and detection network architecture are also explained in the following. Convolutional neural networks (CNNs), a specific type of neural network, have won various competitions in computer vision applications. Speech recognition, image recognition, video analysis, language processing, image classification, and segmentation are just a few of CNN's intriguing application areas.

2.5.1.1 Layers of Convolutional Neural Network (CNN)

The biological neural networks that have numerous layers and direct connections between the neurons in each layer are the inspiration for convolutional neural network (CNN). Deep artificial neural networks include convolutional neural networks (CNN) that is suitable for classification of color images.

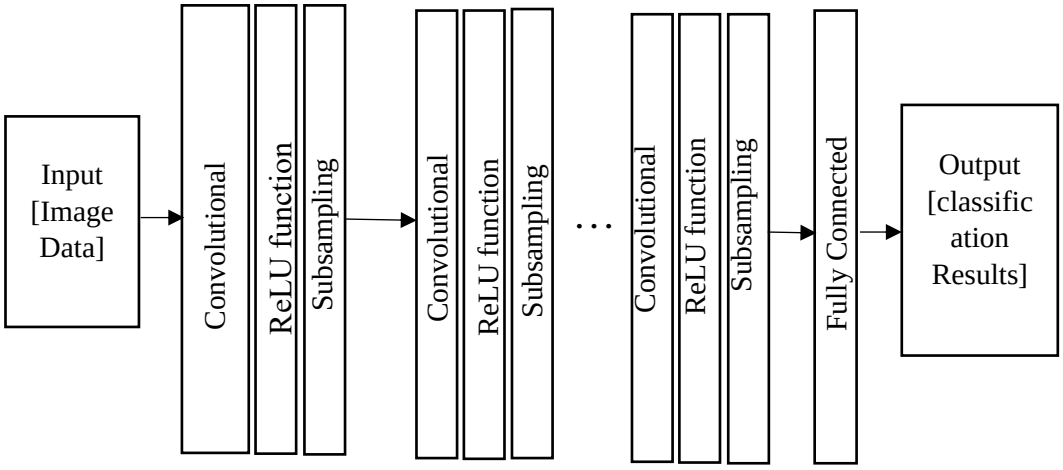


Figure 2.12 Structure of Convolutional Neural Network

There are many different layers in CNN which is presented by Figure 2.12. They are the following layers.

- Input Layer
- Convolutional Layer
- Non-Linear Activation Function (ReLU)
- Sub-Sampling (Pooling) Layer
- Fully Connected Layer

2.5.1.2 Input Layer

Input layer in CNN should include image data. This data is represented by three-dimensional matrix. It generally represents the image's pixel matrix in neural networks for image processing.

2.5.1.3 Convolutional Layer

The primary component of CNN is the convolutional layer. The main functions of Convolutional layer are convolution and padding. Convolution is the mathematical operation of overlapping of two functions. Convolutional process for two dimensional images is following.

$$g_{ij} = \sum_{x=-m}^{m} \sum_{y=-m}^{m} f_{xy} x_{(i-x)(j-y)} \quad \textbf{2.1}$$

whereas, g = image obtained from convolutional process

f = filter kernel used in convolutional process

h = receptive fields of input image that are overlapped

(x, y), (i, j) = coordinate values of feature map and filter map

According to the above equation, convolutional process in CNN is the mathematical operation to obtain the element-wise manipulating result by overlapping the receptive fields with filters. The following Figure 2.13 shows how to convolute the image with filter also called kernel in the convolutional layer.

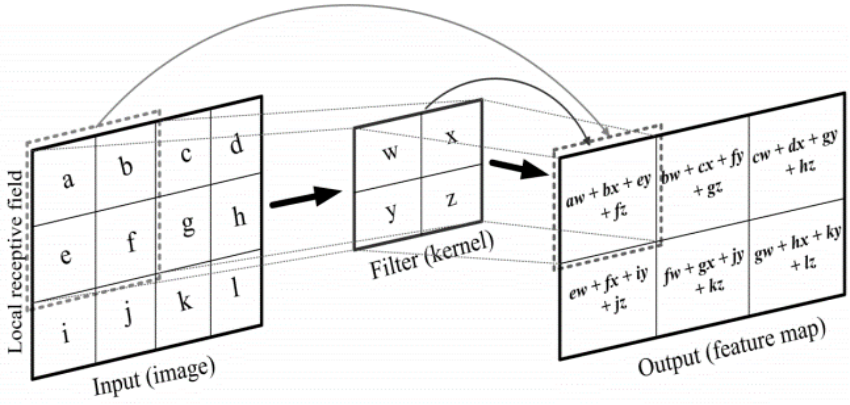


Figure 2.13 Convolutional Processing of Two-Dimensional Image

In the convolutional process, the color channel (depth) of the input image needs to be same with the color channel of the filter (kernel). If the input image is (300×300×3) RGB color image, the filter(kernel) size should be (3×3×3). A feature map is getting out by convoluting the input image with a filter. Therefore, if the input image is convoluted by the number of n^{th} filters, the number of n^{th} feature map will be getting out. Before convoluting the input image, the stride number is defined. Stride number is the number of pixels that the filter kernel moved when overlapping input image. The following Figure 2.14 shows the feature map that is smaller than the input image because the input image is convoluted by stride number 2. Therefore, the more the number of strides, smaller the feature map. To adjust size the feature map, padding the input image is needed.

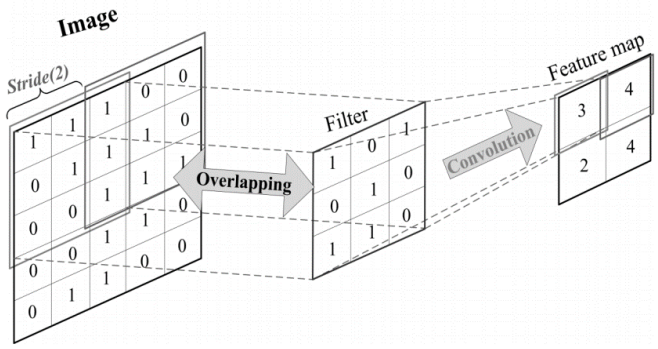


Figure 2.14 Example of Convolutional Processing Using Stride Number 2

Padding is doing across the images or feature maps with a defined value especially "zero". This way of padding is called "zero padding". In CNN, there are three types of zero padding. They are full padding in Figure 2.15, valid padding (also called non-zero padding) in Figure 2.16, same padding (also called half padding) in Figure 2.17 are demonstrated.

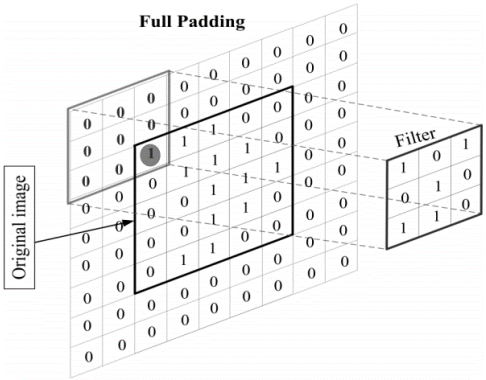


Figure 2.15 Full Padding

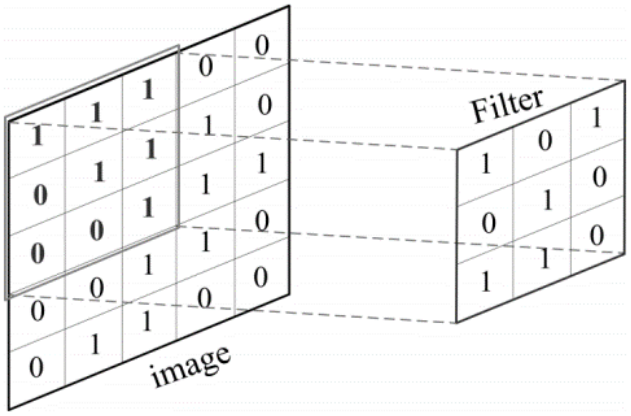


Figure 2.16 Valid Padding

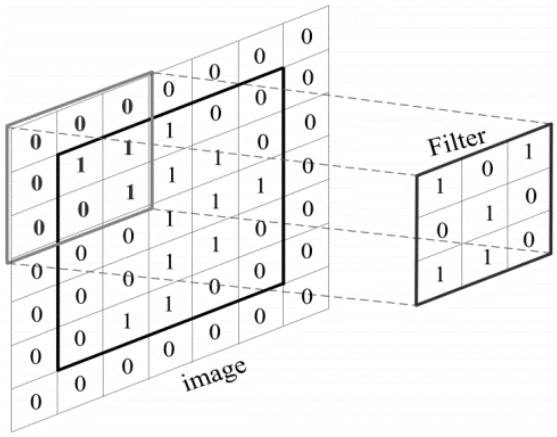


Figure 2.17 Same Padding

2.5.1.4 Non-Linear Activation Function (ReLU)

Non-Linear activation function is used for non-linearization the feature maps which are outing from convolutional layer. There are many non-linear activation functions in CNN. Among of them, Rectified Linear Unit function (ReLU) is mostly used which is displayed in Figure 2.18. The main benefits of using ReLU function is more accurate and faster than using other functions like as tanh and sigmoid function. The main function of ReLU function is to change the minus pixel values of input feature map into zero. The mathematical operation of ReLU function is as following.

$$F(x) = \max(0, x) \qquad \textbf{2.2}$$

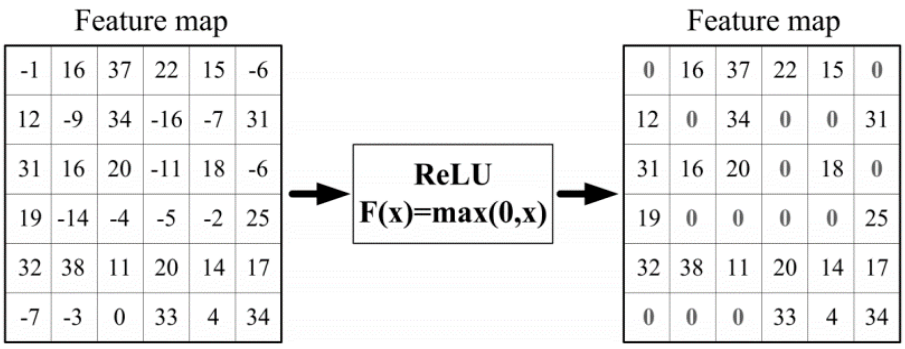


Figure 2.18 Rectified Linear Unit Function (ReLU)

2.5.1.5 Sub-Sampling (Pooling) Layer

The feature maps passed from ReLU function are needed to forward to do sampling. The main function of Sub-sampling is to reduce the size of feature. It is also called pooling layer. By using pooling layer, it can reduce spatial dimension and amount of parameter values. There are many ways to do pooling. The following three ways are mostly used in CNN.

- Maximum pooling
- Average pooling
- L2-Norm Pooling

Maximum pooling: Among above pooling, maximum pooling is mostly used rather than other pooling because it can be better in accuracy. The main function of maximum pooling is to choose maximum value of input feature map which is demonstrated by Figure 2.19. The mathematical operation of maximum pooling is as following.

$$y_{ijk} = max_{p,q} x_{i,j+p,k+q} \quad \textbf{2.3}$$

whereas, y_{ijk} = the value of i^{th} feature map at position j, k

p= the value index in local neighborhood

q= the horizontal index in local neighborhood

y_{ijk}= the pooled image

The following Figure 2.19 shows the processing of maximum pooling.

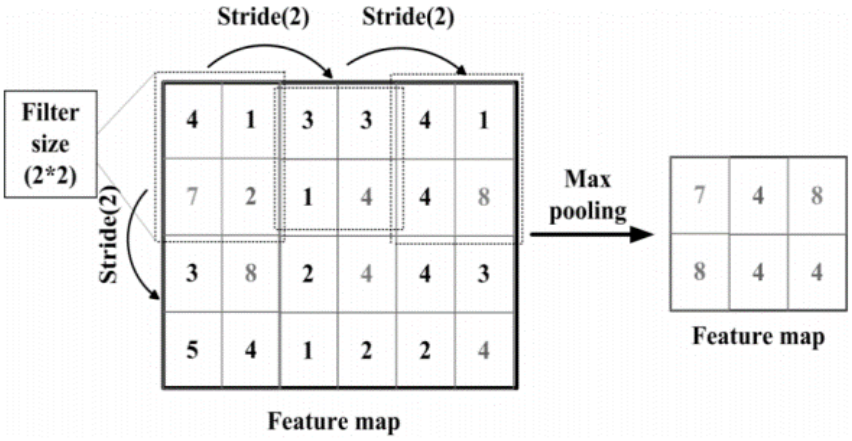


Figure 2.19 Maximum Pooling Processing

Average Pooling: Average pooling is a pooling operation that computes the average value for feature map patches and uses it to generate a pooled feature map. It has no discernible effect on the values of pooled outputs. It extracts smoother features than Max Pooling, whereas Max Pooling extracts more pronounced features such as edges that shown in Figure 2.20.

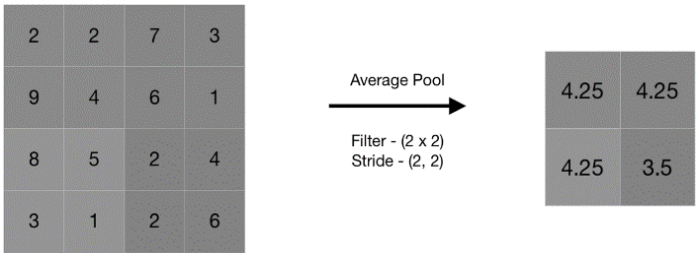


Figure 2.20 Average Pooling Processing

2.5.1.6 Fully Connected Layer

The feature maps are getting out by connecting alternately convolutional and ReLU function. Finally, these feature maps can be hierarchically recognized in fully connected layer. The resulting feature maps are converted into scalar or vector values. These scalar or vector values are used to classify. Therefore, this layer is the final layer of CNN and also called classification layer. The following figure shows the final layer of CNN. Figure 2.21 displays the fully connected layer of CNN.

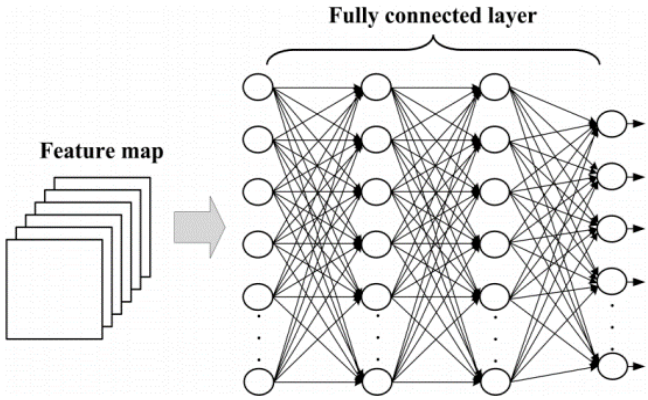


Figure 2.21 Fully Connected Layer

Working procedures of fully connected layer are same as multi-layer Artificial Neural Network. Then, there are also have input, hidden, and output layer. The neurons of input layer are the scalar or vector values of feature maps. The neurons of hidden layer use non-linear activation functions such as hyperbolic tangent function, logistic function and ReLU function. The main process of fully connected layer is the calculation to classify feature maps. Therefore, in this layer, softmax activation function is mostly used. Because, this activation function is based on multiple input multiple output calculation.

2.6 Modern Convolutional Neural Networks

Different types of CNN Models are shown in this section. They are LeNet, AlexNet, VGG, MobileNet, ResNet, MobileNet, Inception/GoogLeNet, NiNet and so on. This section explained detail about some of these architectures [1].

2.6.1 LeNet Model

This model is the first CNN model. It was developed in 1998. This architecture was introduced in 1989 [1]. Convolutional, pooling, and full link layer are the three fundamental deep learning modules that make up LeNet5 in Figure 2.22, a compact network. Other deep learning models are built on this foundation. Develop a deeper grasp of the convolutional layer and pooling layer at the same time through example analysis of this model. This architecture accepts black and white 28x28 image.

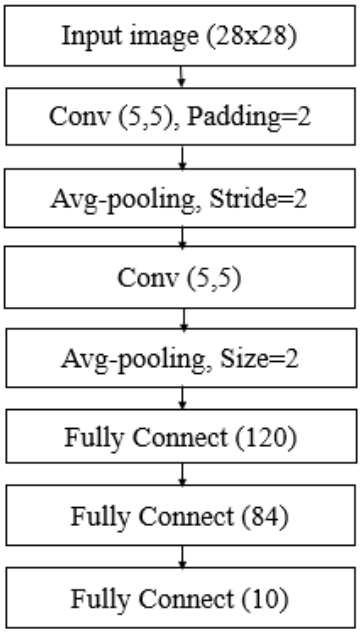


Figure 2.22 Architecture of LeNet5

2.6.2 AlexNet Model

This network is similar to the above model. AlexNet was succeed in 2012 ImageNet Competition [1]. Images were categorized into 1000 different categories using the network. It is constructed of eight layers.

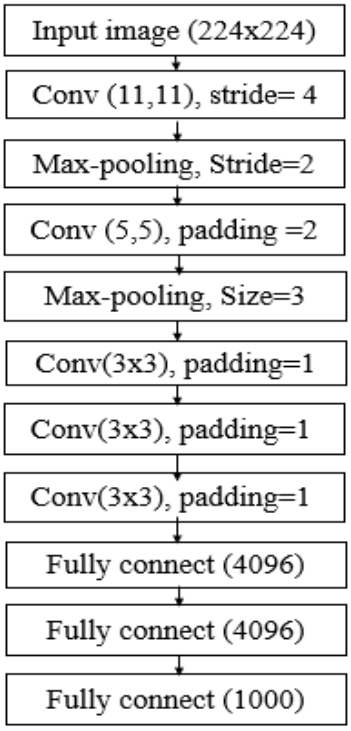


Figure 2.23 Architecture of AlexNet

They are five convolutional layers and three FC hidden layers. The ReLU function was used in this model instead of sigmoid function. Because, model training is facilitated by the ReLU activation function while utilizing various parameter initialization strategies. The above Figure 2.23 shows the model architecture of it.

2.6.4 MobileNet Model

This model is mostly used in mobile applications and it is the first mobile computer vision model of TensorFlow. It uses depth wise separable convolution which cause the light weight deep computational model. The difference between this model and traditional CNN instead of using the batch normalization and ReLU are done after single 3x3 convolutional layer [3]. This model is restricted for embedded computer vision application and light computing devices. This model is built on depth-wise separable convolution. Depth-wise and point-wise convolution are consisting in each depth-wise convolution layer. The working principal flow of depth-wise and point-wise are described in the below. Moreover, the MobileNet architecture can be seen in the Table 2.2.

Table 2.2 Architecture of MobileNetV1

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
$5\times$ Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
$5\times$ Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

2.6.5 ResNet Model

A ground-breaking concept called deep residual networks allowed for the creation of far deeper networks. Deeper networks can learn more complicated functions

and representations of the input, which should improve performance, according to a widely accepted notion. There are many types of ResNet models such as ResNet 18, 34, 50 and so on. In the next chapter, ResNet50 model will be explained in details and how it works in this thesis.

2.7 Chapter Summary

This chapter described about the concept of biological neurons and artificial neurons. The work flow of artificial neural network and the activation functions are also explained. Moreover, deep artificial neural network and how it works in classification/ clustering applications are also presented. And then, this chapter described when training these networks, what types of learning methods are used in training. Various kinds of training methods and working flow of convolutional neural networks are explained. Finally, the architectures of modern CNN layers are also added and each figure can be seen clearly presenting of architectures.

CHAPTER 3

SYSTEM IMPLEMETATION

In this chapter, the system's overview and method usage are explained in detail. Moreover, the working flow of CNN that are used for trash segregation is also described. The step by step of system flow diagram of the trash segregation is explained in detailed. Firstly, how to choose the dataset for classification is explained. And then, the preprocessing functions are described. After that, the structure of classification model is building and the number of parameters and feature mappings of models was calculated. Finally, the dataset is trained by using training processing with CNN. When training, the loss for each train steps are calculated and optimized by using optimization algorithm. These steps by steps for classification of waste type is described in this section.

3.1 Proposed System

Solid waste generation is growing to be a serious issue that requires immediate attention. Because different types of trash require different methods of disposal, a reliable and accurate categorization process is an essential stage in garbage disposal. Due to the varied architecture networks used, the current deep learning-based waste classification models are difficult to generate accurate results and still require improvement.

Their performances on various datasets are different and there are not many particular large-scale training datasets available. For processing the garbage classification dataset and achieving high classification result. Since suggested model is not a straightforward system for classifying waste, the main objective of this system is to provide a system for the segregation of solid waste, particularly recyclables. It is put after thoroughly analyzing all of the limitations and characteristics of waste categories.

To analyze the garbage dataset and achieve high classification accuracy, two classification model based on CNN models have been suggested. The learning model based on each training process is built as a candidate classifier in our proposed model and the best output of the two different classifiers is chosen as the last outcome classification.

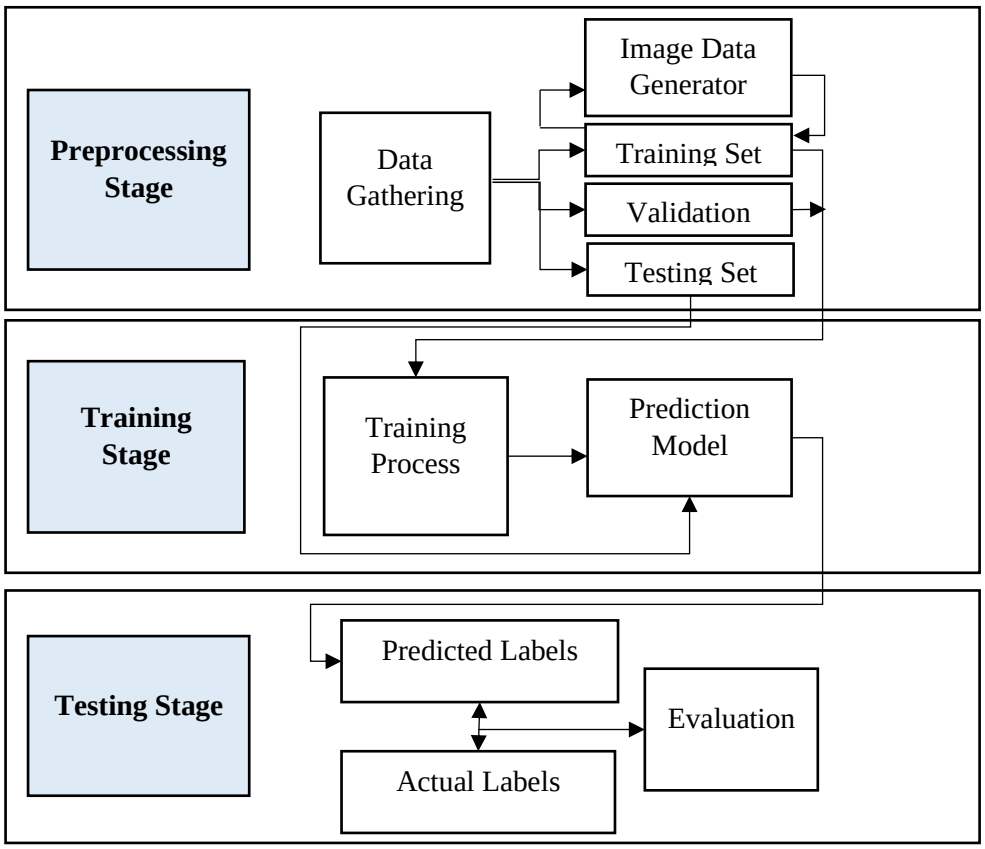


Figure 3.1 System Design for Solid Trash Segregation System

The current method of sorting trash/garbage is hand-picking, in which someone is hired to separate the various objects/materials. Due to the dangerous contaminants in the rubbish, the person who separates waste is more susceptible to illnesses. In light of this, it inspired us to create an automatic waste-sorting system.

Additionally, this technology can sort waste quickly and with greater accuracy than hand sorting. With the system in place, the waste that is usefully separated can still be recycled and turned into fuel and energy for the expansion of the economy. Based on the integration of ANNs that aid in waste classification and convolutional neural networks, the system developed for the separation of accumulated garbage is based on image classification.

ResNet-50 model is employed that is a type of convolutional neural network structure, due to the suitable size of the trash image dataset. The convolutional neural network's recognition accuracy can be improved by increasing the depth, however as the depth increases, the signal that should change the weight at the previous layer of the CNN decreases. This is known as a vanishing gradient and will render learning at the lower layers irrelevant. There are always training errors when a network has more

layers than necessary. The Residual Network (ResNet-50) differs from the conventional convolutional by using modules known as residual models, the ResNet model, and the basic block while creating the Convolutional neural network, neural networks can get around the issue of vanishing gradient. In Figure 3.1, firstly data is needed to gather from open source or manually. And then, this dataset is split into three parts such as training, validation and testing. Training data is processed in data augmentation methods. The models of convolutional neural networks are chosen for training for classification. After working the training process, the prediction model is saved and the test dataset is used for evaluating the accuracy performance of model.

3.1.1 System Flow Diagram

Firstly, Kaggle Garbage data set is chosen. Dataset is divided into the train set, development set and test set. The training phase of the system is demonstrated in Figure 3.2. After splitting, as data augmentation stages, the training data set is preprocessed by using image data generating function.

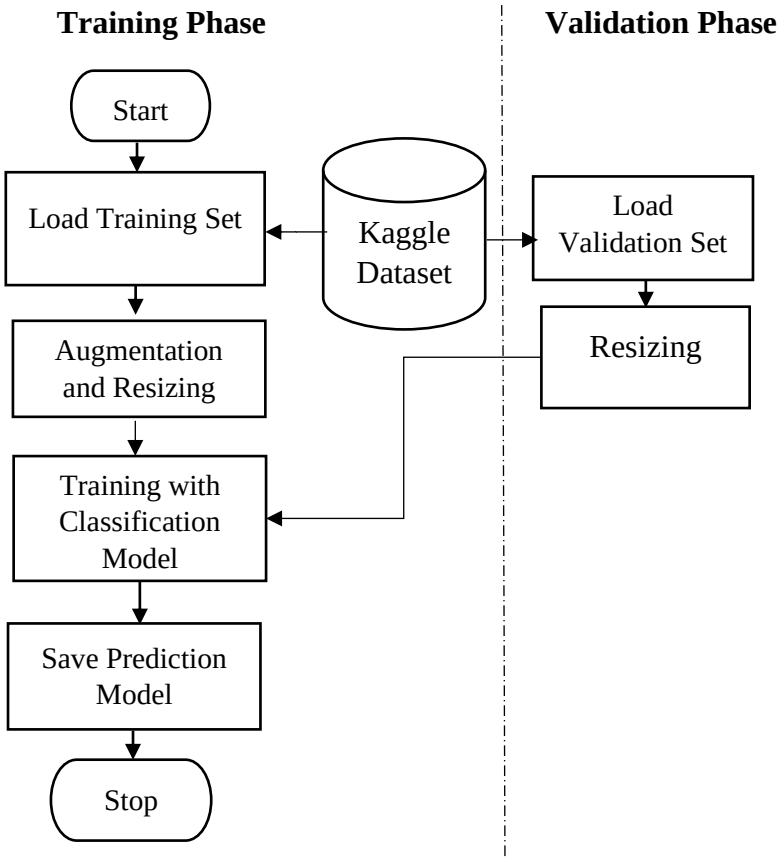


Figure 3.2 System Flow Diagram of Training Phase and Validation Phase

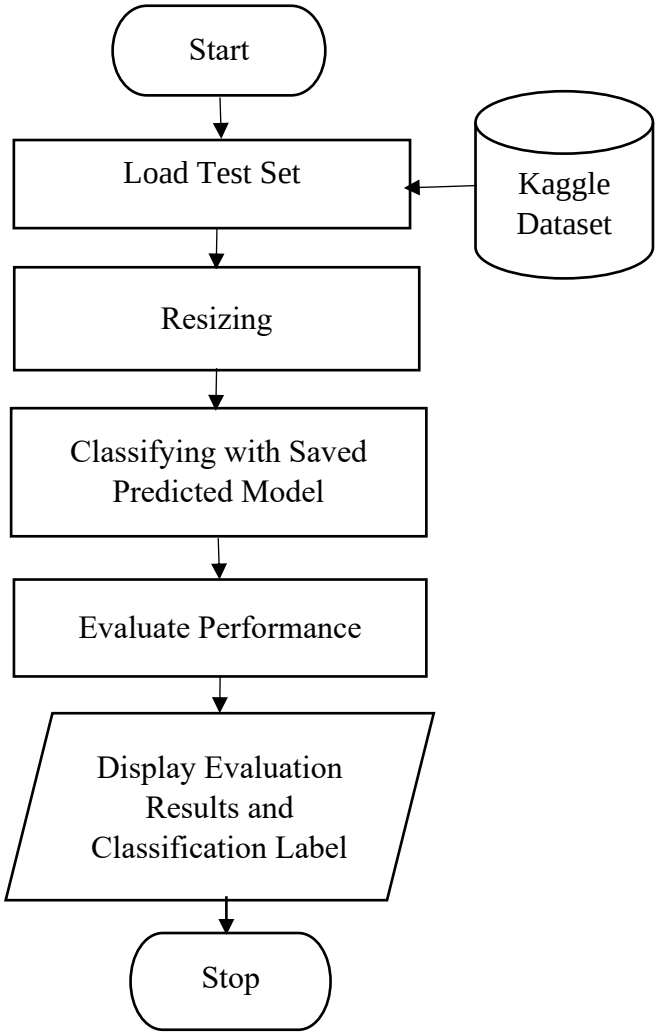


Figure 3.3 System Flow Diagram of Testing Phase

These functions include image rescaling, resizing, flipping, shifting, zooming. The valid and test set are needed to resize as the acceptable input of two models. Training images are trained by using simple CNN based model and modern CNN model (ResNet50) for comparing. Before training process, the validation set is applied to validate the training performance processing of model. Finally, this prediction model is saved. In consequent, this testing images are used to evaluate for accuracy of model performance to compare two models. The evaluation results are getting out and the trash images can be classified. Figure 3.3 displays the trash segregation system flow diagram.

3.1.2 Software Requirements

Google Colab: To combine machine learning with cloud storage, Google has created Colaboratory, a web-based integrated development environment for Python.

This internal technology, which was quietly released to the public in late 2017, is expected to have a significant impact on machine learning, artificial intelligence, and data science work. Colaboratory or Colab, which is particularly well appropriated for data analysis, machine learning and learning, enables anyone to develop and run arbitrary Python code through the browser. In only a few lines of code, Colab drive is used to import the dataset an image and then, train an image classifier on it. Finally, the model is evaluated for classification results. No matter how powerful your computer is, you can take advantage of Google technology, such as GPUs and TPUs, by using Colab laptops, which run code on Google's cloud servers.

Python Programing Language: Guido Van Rossum created the high-level, dynamically typed language Python in the early 1980s. It is an interpretive (bytecode-compiled), dynamic language. Variable, parameter, function, and method types are not declared in the source code. As a result, the source code's compile-time type verification is lost, making the code small and flexible. Python is utilized in video games, operating systems, artificial intelligence, machine learning, and mobile application development. Python is regarded as one of the easiest programming languages to learn for a novice, but it can be challenging to master.

3.2 Data Gathering

Using an open-source dataset is the most straightforward and efficient way to gather data for your ML model. Thousands of open-source datasets are also accessible online, similar to coding snippets. They are free, simple to find, and extremely time-efficient to use. The main drawback of using public datasets is that these may be needed to sanitize them in accordance with particular purpose even though they may contain an infinite number of rich, detailed data. The greatest places to find free datasets are: my absolute favorite source is Kaggle, Amazon, datasets from Google Engine of Search, Microsoft's, AI Lionbridge and so on.

The classification of recyclable materials is the main goal of this study; thus, waste management and recycling programs are crucial components of a sustainable economy. A crucial necessity for affordable and secure recycling is the use of systems rather than people as workers at the landfill. Our goal is to identify some of the most widely used recyclable materials, including glass, paper, cardboard, plastic, metal, and rubbish. There are six classifications of recycled objects in the TrashNet dataset which

can be downloaded from Kaggle Garbage dataset. The dataset's images have a white background and scenes from Stanford's trash and recycling. Each photograph was given a different lighting and attitude, and the dataset has multiple versions. With each image reduced to 512 × 384 pixels [2]. Following Figure 3.4 contains sample images from the dataset, and the number of images for each of six classes are presented by table.

(a) (b) (c) (d) **(e)** **(f)**

Figure 3.4 Sample Dataset: (a) cardboard, (b) glass, (c) metal, (d) paper, (e) plastic, (f) trash

3.3 Data Splitting

When data is divided into two or more subgroups, this is known as data splitting. A two-part split often involves testing or evaluating the data in one part and training the model in the other. Deep learning typically uses data splitting to prevent overfitting. In that case, a machine learning model is unable to reliably fit fresh data because it fits existing training data too well. Usually, the initial data in a machine learning model is split into three or four sets. Training set, development set, and test set are the three sets that are frequently used. These datasets are explained in detail.

- Training dataset: the training set refers to the subset of data used to train the model. The model must observe and learn from the training set in order to enhance any of its parameters.
- Validation dataset: the development set (also called validation set) is a data set that used to alter the settings for the learning process. The cross-validation set or model validation set are other names for it. This data set's goal is to rank the model's accuracy, which can help with model selection.
- Testing dataset: The data set that is compared to the prior data sets and tested in the final model is referred to as the testing set. The testing set is used to gauge how well the algorithm and mode are working.

In this trash classification system, data is split into as following Figure 3.5: This splitting is the most suitable to get highest accuracy.

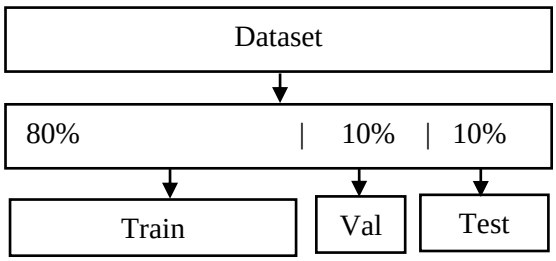


Figure 3.5 Splitting Data for Classification

Table 3.1 Splitting of Kaggle Garbage Dataset

Recyclable Waste Type	Each of category	Train set	Validation set	Test set
Paper	594	476	59	59
Glass	501	401	50	50
Plastic	482	386	48	48
Metal	410	328	41	41
Cardboard	403	323	40	40
Trash	137	111	13	13
Total	2527	2025	251	251

3.4 Data Augmentation Methods

If given enough training data, machine learning algorithms are capable of amazing feats. Unfortunately, obtaining high-quality data continues to be difficult for many applications. Data augmentation, a method that creates new training examples from old ones, is one approach to this issue.

In contexts with little data, data augmentation is a simple and efficient way to increase the efficiency and precision of machine learning models. Models created by machine learning frequently "overfit" when they are trained on a small number of samples. Overfitting is the term used to describe when an ML model performs well on its training examples but fails to generalize to new data. In machine learning, there are various strategies to prevent overfitting, including using different algorithms, changing the model's architecture, and tweaking hyperparameters. But ultimately, increasing the

amount of high-quality data in the training dataset is the key way to combat overfitting [9][10].

Take the convolutional neural network (CNN), for instance. This machine learning architecture is particularly effective at image categorization tasks. A CNN will end up misclassifying images in the actual world without a substantial and varied set of training data. On the other hand, a CNN will grow more reliable in recognizing items in the real world if it is trained on pictures of objects taken at various angles and in various lighting situations. However, obtaining additional training materials can be costly, time-consuming, or occasionally impossible. In supervised learning applications, where training examples must be classified by human specialists, this task is made significantly more challenging.

Making copies of the current data and making minor changes to them is one technique to improve the diversity of the train dataset. The methods of "data augmentation" applies here. For instance, image classification dataset has 20 images of trash. The number of training examples for the "trash" class have been increased by doubling them by making duplicates of images and flipping them horizontally. Other adjustments including rotation, cropping, zooming, and translation are also available.

For training dataset, the data augmentation techniques are used to improve model correctness as part of a machine learning technique to control overfitting, and generalize models is to create generated data based on observed data. The idea behind this technique, which is also known as data augmentation, is that we follow a preset process, taking existing data, such photographs from our training set, and applying specific image alteration operations on them to create a new collection of alternative images. These are some of the most popular image data augmentation methods. As a result, the deep learning model is a little less probable to overfit to the local training data. The augmentation techniques are explained in detail.

Image Flipping: Image flipping is the following method. Flipping can be thought of as a progression of rotation. It enables us to flip the picture both up and down as well as left to right. In this system, horizontal flip and vertical flip are used. In the following Figure 3.6, the most left figure is the original image. The second figure is the horizontal flipping of the image and the third is the vertical flipping.

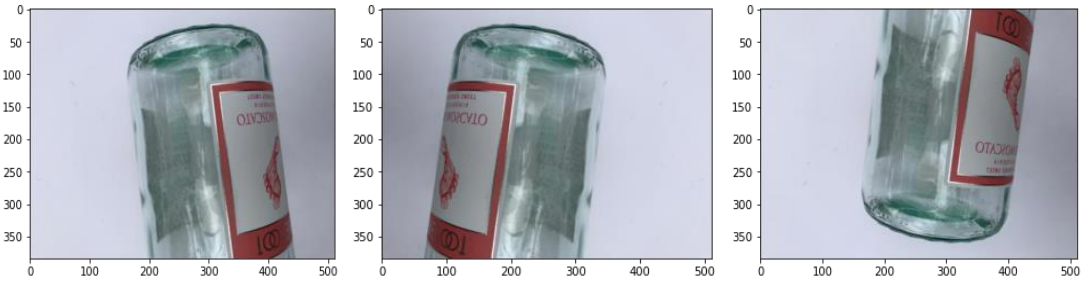


Figure 3.6 Horizontal Flipping and Vertical Flipping

Image Shifting: Image shifting is the following method of image enhancement. The position of the portions in the image can be changed by changing the photos, which gives the model greater diversity. It ultimately might lead to a more comprehensive model. Height shift range moves the pixels up or down in the vertical direction. Width shift range randomly moves the pixels in the horizontal direction to the left or right. This Figure 3.7 shows the shifting with 0.1 value. The left one is width shifting and the right is heigh shifting.

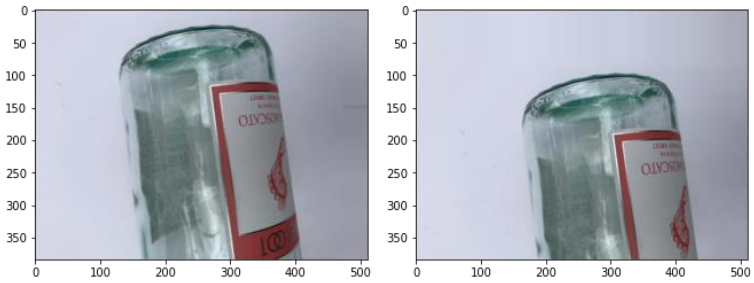


Figure 3.7 Width Shifting (Left) and Heigh Shifting (Right)

Image Zooming: With the zoom Augmentation, the image can be enlarged or reduced in size. A list of two values or a single float value can be entered into the Image Data Generator class. The zoom range is [1-value, 1+value] when only one value is provided. One value is used as the lower limit and the other as the upper limit if a list is provided. The Figure 3.8 shows the zooming range of 0.3 from the original image.

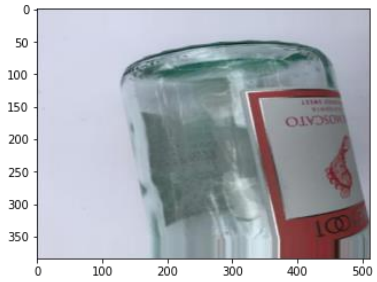


Figure 3.8 Zooming Image

Image Shearing: Shear refers to an axis-based distortion of the image, usually done to produce or correct the perception angles. Typically, it is used to enhance photos so that computers can view objects from various perspectives as people do. The Figure 3.9 presents the shear range 0.1 of the original image.

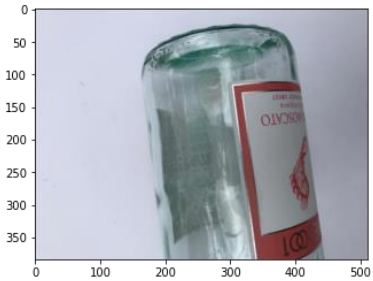


Figure 3.9 Shearing image

Image Rescaling: The maximum pixel value is 255 in image. Rescale 1. /255 will change each pixel's value from [0,255] to [0,1]. Each pixel in a digital image has a value between 0 and 255. White is 255 while black is 0. Three maps are red, green, and blue. All of the pixels are still in the 0 to 255 range that make up a colored image.

3.5 Choosing Architectures of CNN Used in Trash Classification

A deep learning network design known as a convolutional neural network (CNN) learns directly from data. CNNs are incredibly helpful for identifying patterns in images that represent objects. For categorizing voice, time series, and electromagnetic data are examples of non-image data, they can be highly useful. For this trash classification system, simple CNN and ResNet50 architectures are chosen for comparing the performances to achieve the higher accuracy.

It is a challenge that arises when backpropagation and gradient-based learning are a method for instructing artificial neural networks. Gradients are used to update a network's weights in backpropagation, as is well known. But occasionally, this happens and the gradient shrinks to the point where the weights can no longer vary. Because the same values are repeated endlessly and no productive work is accomplished, the network stops learning as a result. Residual neural networks are used to address such issues. By enabling this additional short-cut conduit for the gradient to flow through, ResNet's skip connections address the issue of disappearing gradient in deep neural networks. The identity functions that the model learns from these linkages assure that if not better performance will be achieved by the higher layer than the bottom layer.

3.5.1 Simple CNN Architecture

Two essential elements make up a CNN architecture. The process of feature extraction separates and identifies the distinctive characteristics of an image for analysis using a convolution tool. The feature extraction network has a large number of pairs of convolutional or pooling layers. a fully connected layer that utilizes the results of the convolutional process and establishes the image class utilizing the features that were previously extracted. The goal of this CNN feature extraction model is to reduce the number of features in a dataset. It produces new features that combine the characteristics of an original group of features into a single new feature.

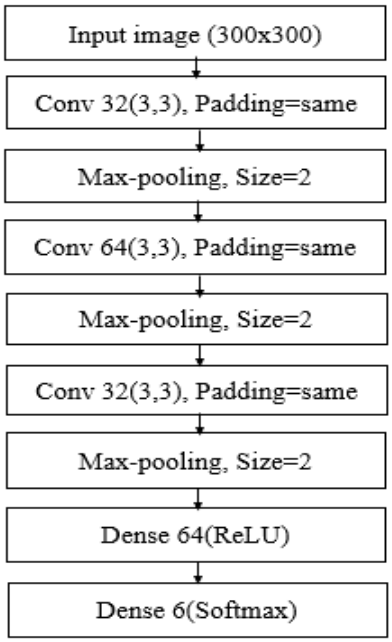


Figure 3.10 Simple CNN Architecture

To assess the performance of a fundamental CNN, a straightforward CNN architecture that necessitates a broad examination of the performance variance between models. This architecture makes use of 2D convolutional (conv.) layers to extract visual features. Filters of size 3X3 have fewer parameters and more uses for nonlinear activation functions than larger filters. In order to reduce the input dimensions and the quantity of training parameters, we add the max pooling layers between the 2D conv. layers. This might aid in preserving important traits following layer conversion and prevent overfitting. According to design implementation of the simple CNN architectural diagram, there are many CNN levels. It is shown in the Figure 3.10.

3.5.2 Residual Neural Network (ResNet)

A number of innovations have been made in the sectors of computer vision and applications of deep learning. These models, particularly with the introduction of very deep Convolutional neural networks, aided in achieving state-of-the-art results on problems such as image recognition and classification. In order to solve increasingly complicated tasks, deep learning architectures have evolved through time (by adding additional layers), which has also improved classification and recognition task performance and strengthened its robustness [11].

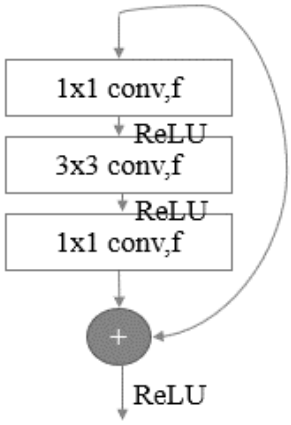


Figure 3.11 Identity Block

However, continuing to layer a neural network, the difficulty of training increases and the accuracy of the model begins to saturate declining. Residual block can reduce this situation. ResNet is built on VGG's convolutional layer design. Every convolutional layer in a residual block is followed by a batch normalization layer and a ReLU activation functionIt then skips these three convolution operations when input is added right before the final ReLU activation function. This connection is called the shortest path. The block with this skip connection is called identity block in Figure 3.11.

An additional 1x1 convolutional layer is required to transform the input into the target shape for the addition operation in order to adjust the number of channels. The network with this additional layer, is called convolutional residual block. Figure 3.12 shows the demonstrating of identity block and convolutional block.

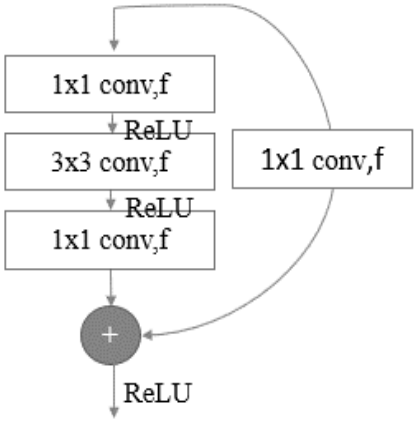


Figure 3.12 Convolutional Block

Batch Normalization: When training very deep neural networks using the batch normalization technique, the contributions to a layer for each mini-batch are adjusted. As a result, the learning process is stabilized and a significant reduction in the number of training epochs required to build deep neural networks. Creating a deep neural network with several layers is necessary for deep learning since these networks might be sensitive to the learning algorithm's architecture and underlying initial random weights.

One possible cause of this issue is that when the weights are refreshed after each mini-batch, the distribution of the inputs to layers below in the network may shift. The difficulty lies in the fact that the model is updated by back warding layer-by-layer from the input to the output while using an estimate of error that assumes the weights in the layers before the current layer are constant. A rich approach of parametrizing virtually any deep neural network is provided by batch normalization. Reparameterization significantly lessens the problem of updating plans over many levels.

Resnet 50 Model Architecture: The ResNet-50 model is divided into 5 stages, each comprising a convolution and an identity block in Figure 3.13. Each identity block has three convolution layers, as do each of the convolution blocks. In the following, the general architecture of ResNet50 model is described. convolution, batch normalization, ReLU, and max pooling are the four blocks that make up the first stage. The other four stages each have a convolution and an Identity block. Both each identity block and each convolution block have three convolution layers apiece. The ResNet-50 has a total of about 23 million trainable parameters.

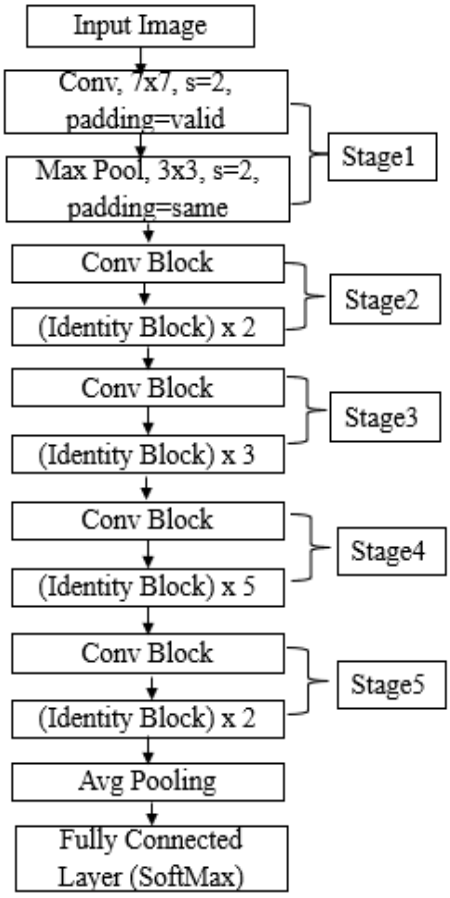


Figure 3.13 Resnet50 Model Architecture

3.6 Calculating the Number of Parameters of CNN

After choosing architectures of CNN, the parameters of each layer which including in CNN models are needed to identified. Depending on the following three consideration groups, it can be calculated the parameters of each layer of models shown in Table 3.2.

- Number and size of filters, number of strides, methods of padding and types of activation function which all are used in convolutional layers.
- Number of pooling methods, pooling size and number of strides used in pooling layer.
- Number of neurons and types of activation functions in multilayer network of fully connected layers.

Parameters of CNN is also called the large amount of trainable parameter. So that, the filter numbers of convolutional layer and weigh and bias values of fully connected layer are needed to calculate the total number of parameters [13].

Input layer: At its essence, the input layer merely provides the shape of the input image; it has nothing to learn. So, there are no learnable parameters here. As a result, there are no parameters.

Table 3.2 Calculation of the Number of Parameters in Simple CNN

Layer	Filter/Weight	Number of Parameters
Conv2D	((3x3x3) +1)32	896
Max-Pooling		0
Conv2D	((3x3x32) +1)64	18496
Max-Pooling		0
Conv2D	((3x3x64) +1)32	18464
Max-Pooling		0
Dense	(64x43808) +(1x64)	2803776
Dense	(6x64) +(1x6)	390

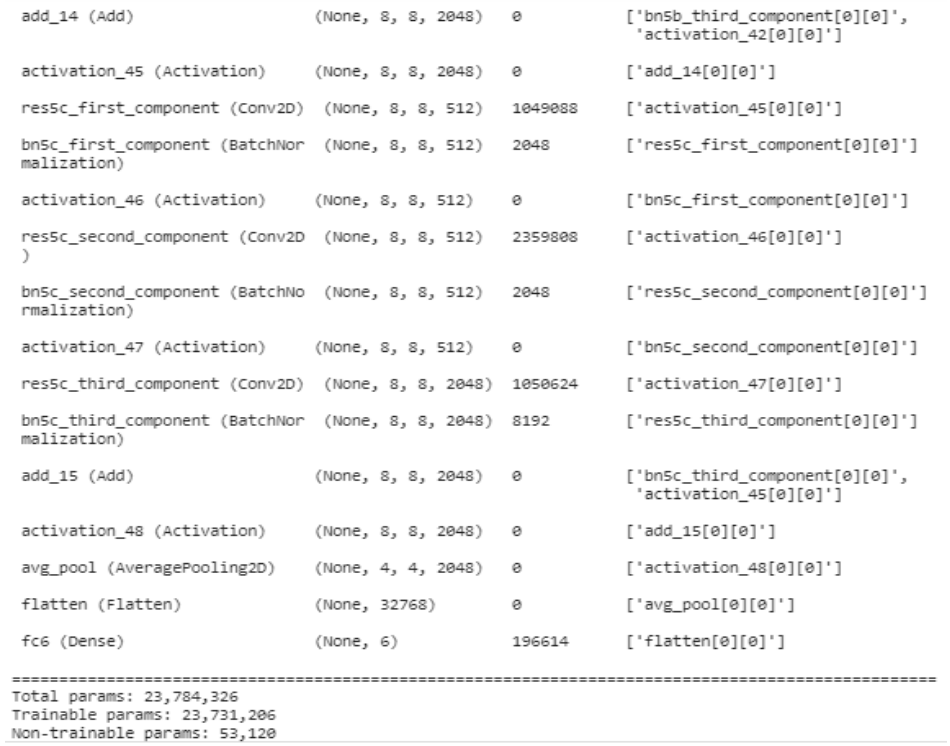
```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 300, 300, 32)      896

 max_pooling2d (MaxPooling2D  (None, 150, 150, 32)     0
 )

 conv2d_1 (Conv2D)           (None, 150, 150, 64)      18496

 max_pooling2d_1 (MaxPooling  (None, 75, 75, 64)       0
 2D)

 conv2d_2 (Conv2D)           (None, 75, 75, 32)        18464

 max_pooling2d_2 (MaxPooling  (None, 37, 37, 32)       0
 2D)

 flatten (Flatten)           (None, 43808)             0

 dense (Dense)               (None, 64)                2803776

 dense_1 (Dense)             (None, 6)                 390

=================================================================
Total params: 2,842,022
Trainable params: 2,842,022
```

Figure 3.14 Compiling Simple CNN Model

Convolutional Layer: Since CNN learns at the convolutional layer, weight matrices are unavoidable. Simply multiplying the shape of width, a, previous layer's filters d, height b, and accounting for all such filters k in the current layer will yield the learnable parameters in this case. For each filter, don't forget the bias word. A CONV layer would have the following number of parameters: ((a * b * d) +1) * k), with 1 added for each filter's bias term. It is possible to write the same expression as ((shape

of height of the filter * shape of width of the filter * number of filters in the previous layer+1) *number of filters). The number of filters in a system is indicated by the term "filter" in current layer.

Pooling Layer: Since all it does is calculate a certain number, there are no learnable parameters in this layer because there is no backprop learning involved! As a result, there are no parameters.



```
add_14 (Add)                    (None, 8, 8, 2048)   0           ['bn5b_third_component[0][0]',
                                                                  'activation_42[0][0]']

activation_45 (Activation)      (None, 8, 8, 2048)   0           ['add_14[0][0]']

res5c_first_component (Conv2D)  (None, 8, 8, 512)    1049088     ['activation_45[0][0]']

bn5c_first_component (BatchNor  (None, 8, 8, 512)    2048        ['res5c_first_component[0][0]']
malization)

activation_46 (Activation)      (None, 8, 8, 512)    0           ['bn5c_first_component[0][0]']

res5c_second_component (Conv2D  (None, 8, 8, 512)    2359808     ['activation_46[0][0]']
)

bn5c_second_component (BatchNo  (None, 8, 8, 512)    2048        ['res5c_second_component[0][0]']
rmalization)

activation_47 (Activation)      (None, 8, 8, 512)    0           ['bn5c_second_component[0][0]']

res5c_third_component (Conv2D)  (None, 8, 8, 2048)   1050624     ['activation_47[0][0]']

bn5c_third_component (BatchNor  (None, 8, 8, 2048)   8192        ['res5c_third_component[0][0]']
malization)

add_15 (Add)                    (None, 8, 8, 2048)   0           ['bn5c_third_component[0][0]',
                                                                  'activation_45[0][0]']

activation_48 (Activation)      (None, 8, 8, 2048)   0           ['add_15[0][0]']

avg_pool (AveragePooling2D)     (None, 4, 4, 2048)   0           ['activation_48[0][0]']

flatten (Flatten)               (None, 32768)        0           ['avg_pool[0][0]']

fc6 (Dense)                     (None, 6)            196614      ['flatten[0][0]']

==================================================================================================
Total params: 23,784,326
Trainable params: 23,731,206
Non-trainable params: 53,120
```

Figure 3.15 Compiling ResNet50 Model

Fully Connected Layer: It has unquestionably teachable parameters; in fact, as compared to the other layers, this layer type has the most parameters. Since every neuron communicates with every other neuron. It is the result of multiplying the number of neurons on layer p at the moment by the number of neurons on layer c before it. The bias term is needed to be considered. As a result, there are ((current layer neurons p * preceding layer neurons c) +1*p) parameters here. Calculating of parameters of each layer is described in the Table 3.2.

In the above Figure 3.14, the total trainable parameter of the simple CNN architecture is 2,842,022. In ResNet50 model, the trainable parameter is over 23 million and compiling model is shown in Figure 3.15. By comparing these parameters, the ResNet50 is more than parameter according to the layers. But, ResNet50 has non-trainable parameters cause of batch normalization.

3.7 Training Process

The process of altering the weights' values is referred to as "training" the neural network. The CNN starts off by using random weights. During CNN training, a sizable dataset of pictures with their matching class labels are input into the neural network. Each image is processed by the CNN network with values assigned at random, and comparisons are then made with the class label of the original image.

If the output does not match the class label (which typically occurs early in the training process), the CNN algorithm makes a slight modification to the CNN neurons' weights to ensure that the output accurately fits the class label image. Through a process called backpropagation, adjustments are being made to the weights' values. Backpropagation streamlines tuning and facilitates modifications for greater precision.

Every iteration of the picture dataset's training is referred to as an "epoch." Throughout the training process, the CNN travels through many series of epochs, altering its weights by the necessary little amounts. The neural network gets a little bit better at correctly categorizing and predicting the class of the training images after each epoch step. The weights are adjusted, but as CNN becomes better, the weight modifications get smaller and smaller.

```
Epoch 1/100
60/60 [==============================] - ETA: 0s - loss: 0.9334 - acc: 0.6433
Epoch 1: val_acc improved from 0.52778 to 0.53993, saving model to /content/drive/MyDrive/Simple CNN/trained_model_Epoch100.h5
60/60 [==============================] - 45s 758ms/step - loss: 0.9334 - acc: 0.6433 - val_loss: 1.2964 - val_acc: 0.5399
Epoch 2/100
60/60 [==============================] - ETA: 0s - loss: 0.8806 - acc: 0.6658
Epoch 2: val_acc did not improve from 0.53993
60/60 [==============================] - 45s 743ms/step - loss: 0.8806 - acc: 0.6658 - val_loss: 1.2903 - val_acc: 0.5278
Epoch 3/100
60/60 [==============================] - ETA: 0s - loss: 0.8795 - acc: 0.6721
Epoch 3: val_acc improved from 0.53993 to 0.54167, saving model to /content/drive/MyDrive/Simple CNN/trained_model_Epoch100.h5
60/60 [==============================] - 46s 773ms/step - loss: 0.8795 - acc: 0.6721 - val_loss: 1.2322 - val_acc: 0.5417
Epoch 4/100
60/60 [==============================] - ETA: 0s - loss: 0.8827 - acc: 0.6689
Epoch 4: val_acc improved from 0.54167 to 0.55903, saving model to /content/drive/MyDrive/Simple CNN/trained_model_Epoch100.h5
60/60 [==============================] - 45s 752ms/step - loss: 0.8827 - acc: 0.6689 - val_loss: 1.1962 - val_acc: 0.5590
Epoch 5/100
60/60 [==============================] - ETA: 0s - loss: 0.8391 - acc: 0.6825
Epoch 5: val_acc improved from 0.55903 to 0.57812, saving model to /content/drive/MyDrive/Simple CNN/trained_model_Epoch100.h5
60/60 [==============================] - 45s 750ms/step - loss: 0.8391 - acc: 0.6825 - val_loss: 1.1539 - val_acc: 0.5781
Epoch 6/100
60/60 [==============================] - ETA: 0s - loss: 0.8296 - acc: 0.6831
Epoch 6: val_acc did not improve from 0.57812
60/60 [==============================] - 46s 764ms/step - loss: 0.8296 - acc: 0.6831 - val_loss: 1.2957 - val_acc: 0.5538
Epoch 7/100
```

Figure 3.16 Training Process of Simple CNN

In the Figure 3.16 and 3.17, the train loss, validation loss and validation accuracy are shown during training process of simple CNN and ResNet50. In the experiment, the training process is done in epoch100.

```
Epoch 1: train_loss: 1.4595, val_loss: 1.2311, val_acc: 0.8366
Epoch 2: train_loss: 1.1683, val_loss: 1.1500, val_acc: 0.9428
Epoch 3: train_loss: 1.0968, val_loss: 1.1467, val_acc: 0.9272
Epoch 4: train_loss: 1.0743, val_loss: 1.1254, val_acc: 0.9144
Epoch 5: train_loss: 1.0602, val_loss: 1.1180, val_acc: 0.9273
Epoch 6: train_loss: 1.0606, val_loss: 1.1183, val_acc: 0.9351
Epoch 7: train_loss: 1.0599, val_loss: 1.1380, val_acc: 0.8884
Epoch 8: train_loss: 1.0568, val_loss: 1.1143, val_acc: 0.9532
Epoch 9: train_loss: 1.0591, val_loss: 1.1311, val_acc: 0.9195
Epoch 10: train_loss: 1.0564, val_loss: 1.1118, val_acc: 0.9455
Epoch 11: train_loss: 1.0542, val_loss: 1.1153, val_acc: 0.9248
Epoch 12: train_loss: 1.0534, val_loss: 1.1167, val_acc: 0.9247
Epoch 13: train_loss: 1.0523, val_loss: 1.1227, val_acc: 0.9273
Epoch 14: train_loss: 1.0548, val_loss: 1.1180, val_acc: 0.9273
Epoch 15: train_loss: 1.0500, val_loss: 1.1231, val_acc: 0.9144
Epoch 16: train_loss: 1.0528, val_loss: 1.1164, val_acc: 0.9352
Epoch 17: train_loss: 1.0514, val_loss: 1.1038, val_acc: 0.9403
Epoch 18: train_loss: 1.0511, val_loss: 1.1098, val_acc: 0.9325
Epoch 19: train_loss: 1.0506, val_loss: 1.1128, val_acc: 0.9300
Epoch 20: train_loss: 1.0481, val_loss: 1.1110, val_acc: 0.9325
Epoch 21: train_loss: 1.0505, val_loss: 1.1268, val_acc: 0.9195
Epoch 22: train_loss: 1.0516, val_loss: 1.1077, val_acc: 0.9480
Epoch 23: train_loss: 1.0504, val_loss: 1.1047, val_acc: 0.9507
Epoch 24: train_loss: 1.0476, val_loss: 1.1279, val_acc: 0.9092
Epoch 25: train_loss: 1.0488, val_loss: 1.1232, val_acc: 0.9117
```

Figure 3.17 Training Process of ResNet50

3.8 Loss Functions

Loss function is fundamentally very straightforward: That is a way to gauge how algorithm models well the data it uses. Loss function will produce a greater value if forecasts were completely incorrect. If they're decent, it will produce a lower number. Loss function will indicate whether making progress of your algorithm to try to enhance model. Loss functions are related to model accuracy. Different types of loss functions are here. It can be difficult to choose a loss function or even understand what a loss function is and how it affects neural network training because there are so many options available.

- Mean Square Error
- Categorical Cross Entropy Loss
- Likelihood Loss
- Cross Entropy Loss

3.8.1 Categorical Cross Entropy

When there are two or more output labels in a multi-class classification model, it is used as a loss function. One-hot category encoding value in the form of 0s and 1

are allocated to the output label. The network has learned the data more effectively when the loss is smaller. Categorical cross entropy loss has to be the most widely used loss function in neural network classification applications. Computation of loss function is in the following Table 3.3. As the predicted probability departs from the labeled probability, cross-entropy loss grows. Think about the classification issue with the following SoftMax predictions (P) and the actual labels (T). The calculation process for cross-entropy loss is presented. The sample information is here.

Table 3.3 Sample Information of Prediction and Actual Value for Loss Calculation

Class	SoftMax Prediction(P)	Next Iteration	Actual Label(T)
Plastic	0.6	0.9	1
Paper	0.116	0.1	0
Glass	0.039	0	0
Metal	0.07	0	0
Trash	0.1	0	0
Cardboard	0.075	0	0

$$\text{Categorical Cross Entropy Loss} = -\sum_{i=1} \text{Tilog(Pi)} \quad \textbf{3.1}$$

By substitution the values in the table, the loss will be gained 0.73. Consider that next some iterations of training model, the loss of model is 0.15. Therefore, the step of iterations is reducing the loss function by updating parameters.

3.9 Optimization Algorithms for Training Processing

Deep learning models are trained with the aid of optimization methods. allowed us to keep altering the model's parameters while minimizing the loss function's value as determined by the training set. Deep learning requires optimization algorithms to be effective. On the one hand, it can take hours, days, or even weeks to train a complicated deep learning model. The effectiveness of training the model is directly influenced by the optimization algorithm's performance. Over the past few years, numerous optimizers have been studied, each with their own benefits and drawbacks. There are various types of optimizers here.

- Gradient Descent

- Adaptive Gradient
- Stochastic Gradient Descent
- Mini Batch SGD
- SGD with Momentum
- Adam

By altering the hyperparameters in each step until achieving the best results, the outcomes in each iteration cab be compared. develop a reliable model is developed with a lower error rate. There are numerous techniques that may be used to optimize a model. In my study, SGD and Adam optimizers are only used for optimization.

3.9.1 Stochastic Gradient Descent Optimizer

Stochastic Gradient Descent is referred to as SGD. For each epoch of SGD, a small number of samples are chosen at random rather than the entire data set. The number of samples from a dataset that are utilized to calculate the gradient for each iteration is referred to as the "batch" in the Gradient Descent algorithm. The batch is assumed that the entire dataset in normal Gradient Descent optimization techniques like Batch Gradient Descent. The issue emerges when our datasets are particularly large, even though using the entire dataset is really important for reaching the minima in a less noisy or less random manner.

The SGD approach improves upon the GD algorithm and addresses a number of its flaws. The disadvantage of gradient descent is that in order to compute the derivative of the loss function, it must load the entire n-point dataset at once, which consumes a significant amount of memory. The derivative is calculated using the SGD algorithm. The advantages of SGD are here.

- Model parameters are often updated; as a result, convergence occurs faster.
- less memory is used because loss function values don't need to be stored.
- perhaps new minima.

Gradient Descent is a version of it. It aims to perform more frequent parameter updates for the model. The model parameters in this are changed when the loss on each training example is computed. Therefore, instead of updating the model parameters once like in Gradient Descent if the dataset has 1000 images, SGD will do it 1000 times.

As a result of the model's frequent parameter updates, there are large variances and fluctuations in the loss functions at various intensities.

3.9.2 Adaptive Moment Estimation Optimizer

This optimizer is a different algorithm of optimization that can be used to train deep learning models instead of stochastic gradient descent. The greatest features of the AdaGrad and RMSProp algorithms are combined to provide an optimization method that can handle sparse gradients in noisy environments. Adam is a technique that leverages first-order gradients to optimize stochastic objective functions by adaptively estimating lower-order moments. The method is simple to use, computationally effective, requires little memory, is adaptable to gradient scale that is diagonal, and works well for problems with a lot of parameters or data.

The strategy is equally appropriate for non-stationary objectives and severe noise and/or sparse gradient concerns. The hyper-parameters may usually be tuned to a reasonable degree and have intuitive interpretations. Adam performs admirably in real-world applications and outperforms other stochastic optimization techniques. The advantages of this optimizer are shown in following.

- The approach converges too quickly and moves too quickly.
- Corrects large variance and vanishing learning rate.

3.10 Chapter Summary

The step-by-step implementation of solid trash segregation system is explained in detailed in this chapter. The system flow diagram and it’s each of the steps have been described. This chapter presented about detailed explanation of dataset, architectures of CNN, training process of these models, loss functions and optimization techniques used in this system.

CHAPTER 4

EXPERIMENTAL RESULTS AND ANALYSIS

In this chapter, the output classification results of the solid trash segregation system have been shown step by step. And then, the performance of two model is evaluated by comparing in two different iterations and different splitting of dataset. The analysis results are shown and how the results are different by seeing comparisons.

4.1 Experimental Results of the System

In this section, the results of simple CNN and ResNet50 model are presented. Firstly, the data augmentation of the training dataset. The experimental results of augmentation have been shown in chapter 3. And then, the training processes and loss out puts are also described. The classification results of the system are shown in the following. The following results are Simple CNN and ResNet50 model.

4.1.1 Experimental Results of Simple CNN

In the following Figure 4.1 and 4.2, the results of the testing of simple CNN have been shown.

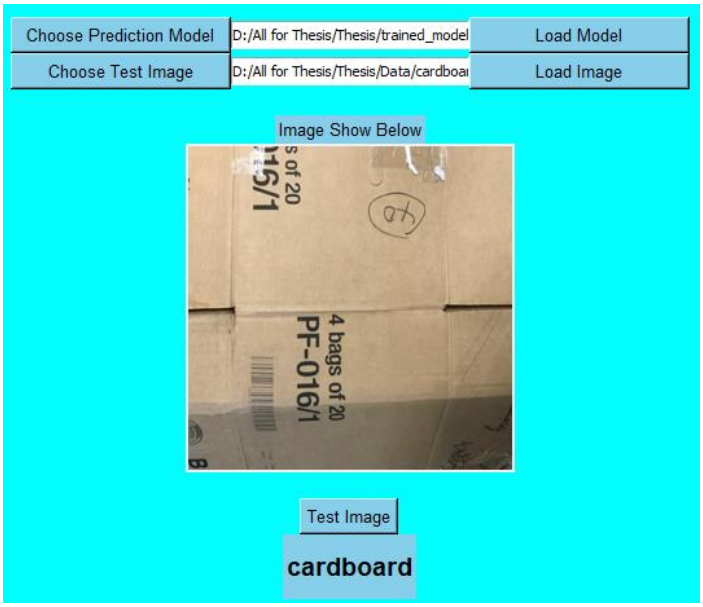


Figure 4.1 Incorrect Prediction of Simple CNN

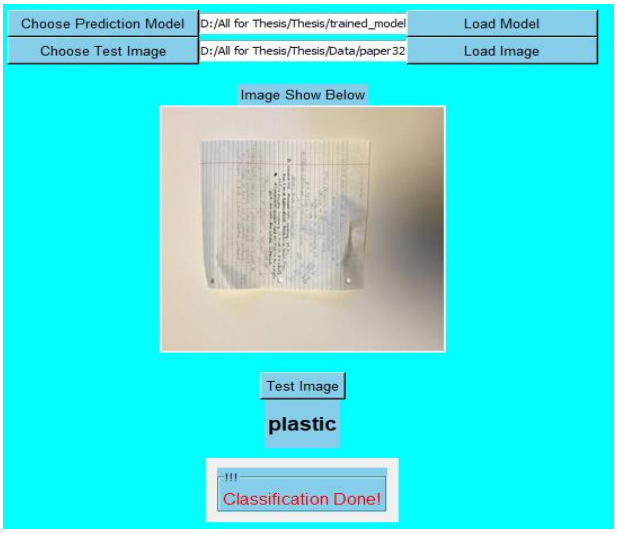


Figure 4.2 Correct Prediction of Simple CNN

The Figure 4.1 show the correct prediction of simple CNN. As consequently, the wrongly testing is shown in Figure 4.2.

4.1.2 Experimental Results of ResNet50

When experimentation of the ResNet50, the following results are getting out. The ResNet50 is achieving 96% of accuracy. The experimental results of prediction model are shown in the following user interface.

Figure 4.3 Correct Prediction of ResNet50

The Figure 4.3 show the correct prediction of ResNet50. As consequently, the wrongly testing is shown in Figure 4.4.

Figure 4.4 Incorrect Prediction of ResNet50

4.2 Performance Evaluation of the System

There are various ways to check the performance of machine learning or deep learning model. Model evaluation is the practice of employing several evaluation measures to comprehend the performance and strengths and weaknesses of a machine learning model. With the early stages of research, it is crucial to evaluate a model's effectiveness. Model evaluation also aids in model monitoring. Accuracy, precision, recall, f1 score and confusion matrix are the three most prominent measures for gauging categorization performance [14].

Confusion: The confusion matrix, often known as the confusion table, provides a more thorough description of the proper and improper groupings for each class. When attempting to comprehend the differences across classes, using a confusion matrix might be helpful, especially if one class has significantly more test data than the other or the cost of misclassification could be different for the two groups.

In the confusion matrix, there are parameters are represented. These are True Positives (TP): Predicted positive and are actually positive. False Positives (FP): Predicted positive and are actually negative. True Negatives (TN): Predicted negative

and are actually negative. False Negatives (FN): Predicted negative and are actually positive.

Accuracy: As the ratio of the number of accurate predictions to all guesses, accuracy indicates how frequently the classifier predicts correctly. The most popular criteria for evaluating a model are not a reliable gauge of its performance. When classes are unbalanced, things get worse. The formula is here.

$$\text{Accuracy} = \frac{\text{TP+TN}}{\text{TP+TN+FP+FN}} \quad \textbf{4.1}$$

Precision: The number of waste classes predictions that actually belong to the positive class.

$$\text{Precision} = \frac{\text{TP}}{\text{TP+FP}} \quad \textbf{4.2}$$

Recall: The number of positive trash class predictions made out of all positive examples in the dataset.

$$\text{Recall} = \frac{\text{TP}}{\text{TP+FN}} \quad \textbf{4.3}$$

F1 score: The harmonic mean of precision and recall is what the term "F1-score" refers to. It utilizes the formula below to combine recall and precision into a single number: It is important to note that F1-score accounts for both FPs and FNs because it considers both precision and recall. The F1-score increases with increased precision and recall. F1-score values vary from 0 to 1. The model is more accurate the closer it is to 1.

$$\text{F1 score} = 2 * \frac{\text{Precision}*\text{Recall}}{\text{Precision+Recall}} \quad \textbf{4.4}$$

Macro Average: Among the several averaging techniques, macro averaging is possibly the simplest. The arithmetic mean of all the per-class F1 scores is used to get the macro-averaged F1 score, also known as the macro F1 score. Regardless of the tash values, all waste classes are treated equally by this method.

Weighted Average: The weighted-average F1 score is determined by averaging all of the per-class F1 scores while accounting for the support of each waste class. The output average would have taken into consideration the contribution of each waste class as weighted by the quantity of samples belonging to that particular class if weighted averaging had been used.

4.3 Performance Analysis of Models

In this study, two models are compared in different situations of the partition of data set in different epochs of training. Two models such as ResNet50 and simple CNN are analyzed. The test data set are used to analyze the model performance. The dataset is divided into different ways.

For the first experiment, dataset is divided into 80 percent training and 10 percent validation and 10 percent testing. For the second experiment, the dataset is split into 70 percent training, 15 percent validation and 15 percent testing. Moreover, the training process is experimented in two different iterations. The first experiment is done during 50 epochs and the second is done during 100 epochs. Performance evaluations of the different models in different partitions are shown.

4.3.1 Performance Analysis of Simple CNN Model

Figure 4.5 Plotting Accuracy of Simple CNN: 50 Epochs and 100 Epochs

When the dataset is split into 80% training, 10% validation, and 10% testing, the following charts display the training loss and accuracy in Figure 4.5.

Figure 4.6 Plotting Loss of Simple CNN: 50 Epochs and 100 Epochs

The accuracy performances parameter computation of the model is displayed in Figure 4.6 when testing the test data.

	precision	recall	f1-score	support
0	0.88	0.57	0.70	40
1	0.72	0.82	0.77	50
2	0.79	0.66	0.72	41
3	0.85	0.95	0.90	61
4	0.60	0.65	0.62	48
5	0.69	0.85	0.76	13
accuracy			0.75	253
macro avg	0.76	0.75	0.74	253
weighted avg	0.76	0.75	0.75	253

	precision	recall	f1-score	support
0	0.16	0.12	0.14	40
1	0.22	0.22	0.22	50
2	0.14	0.15	0.14	41
3	0.19	0.20	0.20	59
4	0.18	0.19	0.18	48
5	0.00	0.00	0.00	13
accuracy			0.17	251
macro avg	0.15	0.15	0.15	251
weighted avg	0.17	0.17	0.17	251

Figure 4.7 Performance Evaluation of Simple CNN: 50 Epochs and 100 Epochs

Figure 4.7 shows the evaluation of each category. The support is the number of classes in testing dataset.

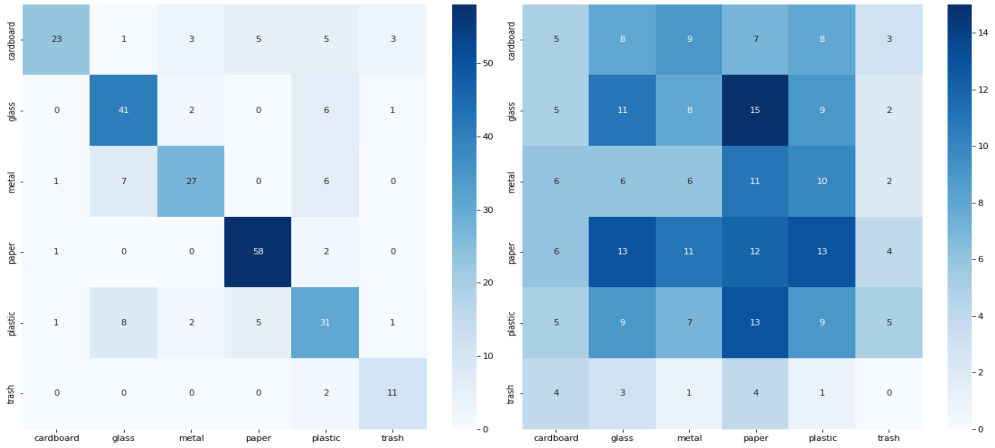


Figure 4.8 Confusion Matrix of Simple CNN: 50 Epochs and 100 Epochs

The confusion matrixes of two models are shown in Figure 4.8.

When the dataset is split into 70% training, 15% validation, and 15% testing, the following charts display the training loss and accuracy.

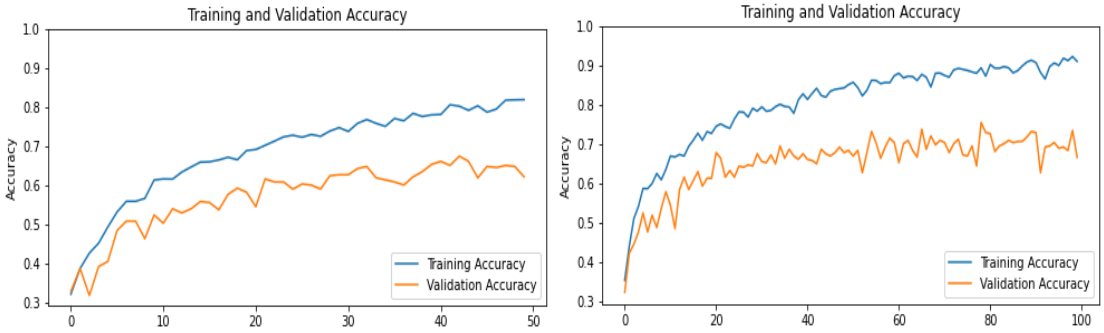


Figure 4.9 Plotting Accuracy of Simple CNN: 50 Epochs and 100 Epochs

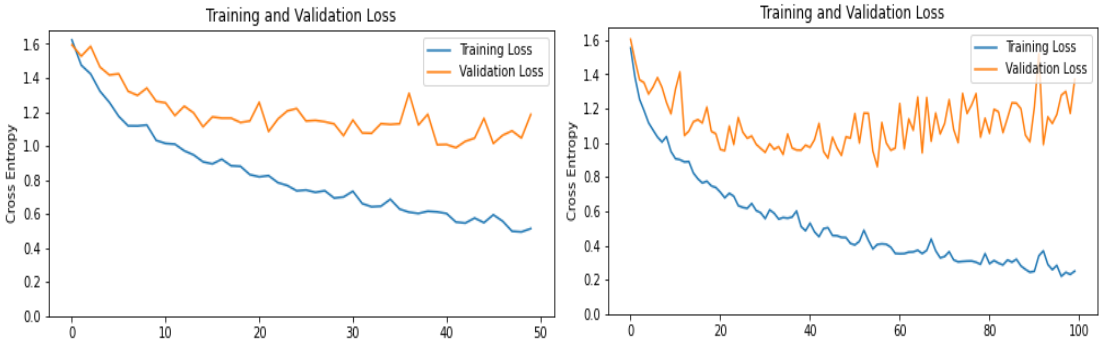


Figure 4.10 Plotting Loss of Simple CNN: 50 Epochs and 100 Epochs

The training and validation accuracy and loss during the training process of simple CNN are shown in Figure 4.10.

	precision	recall	f1-score	support
0	0.84	0.60	0.70	60
1	0.53	0.85	0.66	75
2	0.63	0.66	0.65	61
3	0.82	0.96	0.88	91
4	0.59	0.28	0.38	72
5	0.69	0.45	0.55	20
accuracy			0.68	379
macro avg	0.68	0.63	0.63	379
weighted avg	0.69	0.68	0.66	379

	precision	recall	f1-score	support
0	0.17	0.13	0.15	60
1	0.18	0.17	0.18	75
2	0.19	0.25	0.21	61
3	0.25	0.26	0.25	89
4	0.16	0.15	0.16	72
5	0.00	0.00	0.00	20
accuracy			0.19	377
macro avg	0.16	0.16	0.16	377
weighted avg	0.18	0.19	0.18	377

Figure 4.11 Performance Evaluation of Simple CNN: 50 Epochs and 100 Epochs

The accuracy performances parameter computation of the model is displayed in Figure 4.11 when testing the test data. The comparisons between 50 and 100 epochs show these numbers. The confusion matrixes of simple CNN is shown in Figure 4.12.

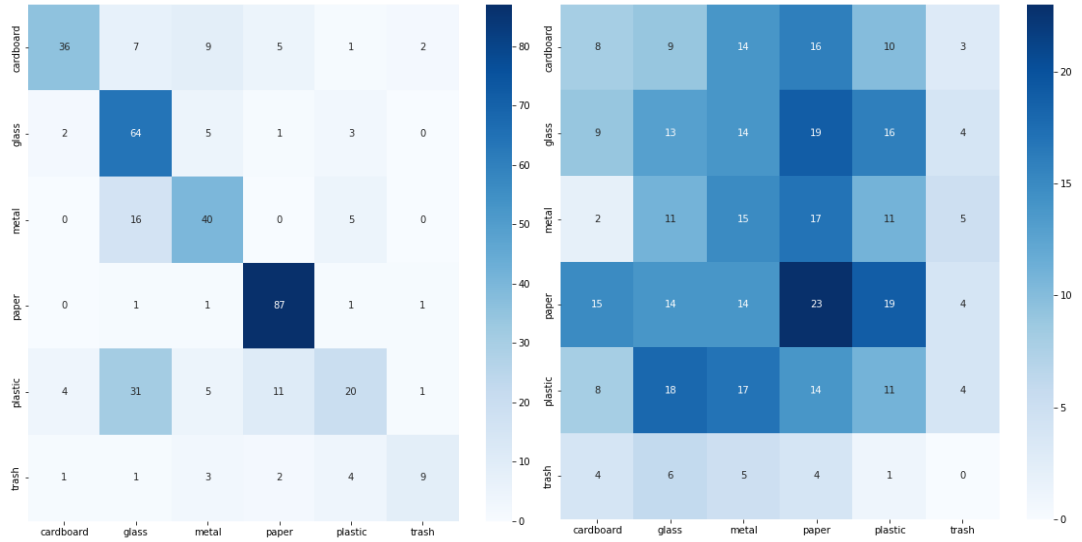


Figure 4.12 Confusion Matrix of Simple CNN: 50 Epochs and 100 Epochs

By analyzing the above steps, now these results are summarized and shown in the following table. The accuracy analyzing on test data of two different epoch on two different splitting datasets of training.

Table 4.1 Summarizing the Results of Simple CNN

	Data Split (80 train /10valid /10 test)		Data Split (70train/15valid/ 15test)	
	Epoch 50	Epoch 100	Epoch 50	Epoch 100
Accuracy	75%	17%	68%	19%

4.3.2 Performance Analysis of ResNet50 Model

The following plots show the training loss and accuracy when the dataset is divided into 80% training, 10%validation and 10%testing.

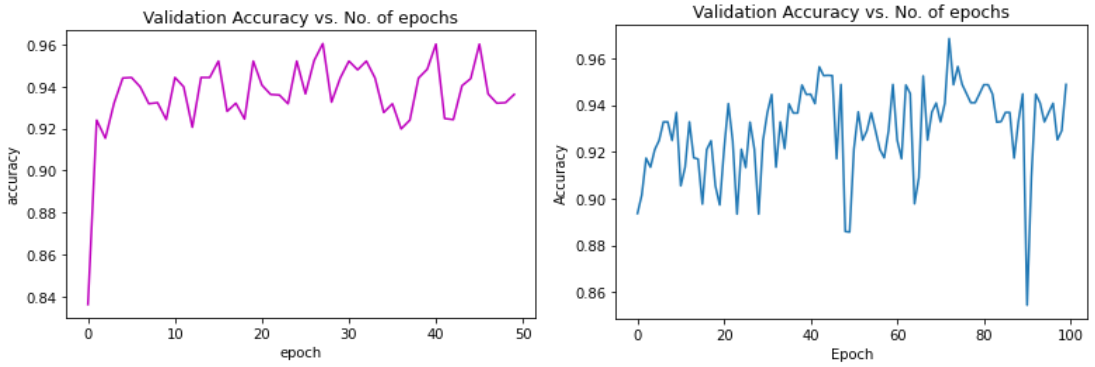


Figure 4.13 Plotting Accuracy of ResNet50: 50 Epochs and 100 Epochs

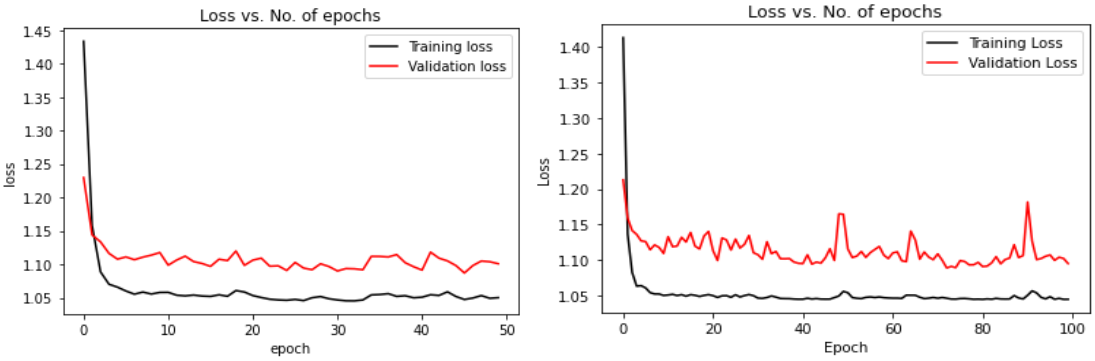


Figure 4.14 Plotting Loss of ResNet50: 50 Epochs and 100 Epochs

When testing the test data, the accuracy performances parameter calculation of the model is shown in Figure 4.13 and 4.14. These figures can be seen the comparisons between 50 and 100 epochs.

	precision	recall	f1-score	support
cardboard	0.97	0.95	0.96	37
glass	0.93	0.91	0.92	44
metal	0.91	1.00	0.95	50
paper	0.96	1.00	0.98	53
plastic	1.00	0.92	0.96	51
trash	1.00	0.94	0.97	17
accuracy			0.96	252
macro avg	0.96	0.95	0.96	252
weighted avg	0.96	0.96	0.96	252

	precision	recall	f1-score	support
cardboard	1.00	0.98	0.99	45
glass	0.90	0.92	0.91	48
metal	0.93	0.95	0.94	44
paper	0.97	0.97	0.97	67
plastic	0.95	0.88	0.91	40
trash	0.83	1.00	0.91	10
accuracy			0.94	254
macro avg	0.93	0.95	0.94	254
weighted avg	0.95	0.94	0.94	254

Figure 4.15 Performance Evaluation of ResNet50: 50 Epochs and 100 Epochs

The performance evaluation results are shown in Figure 4.15. The confusion matrixes of ResNet50 model are shown in Figure 4.16.

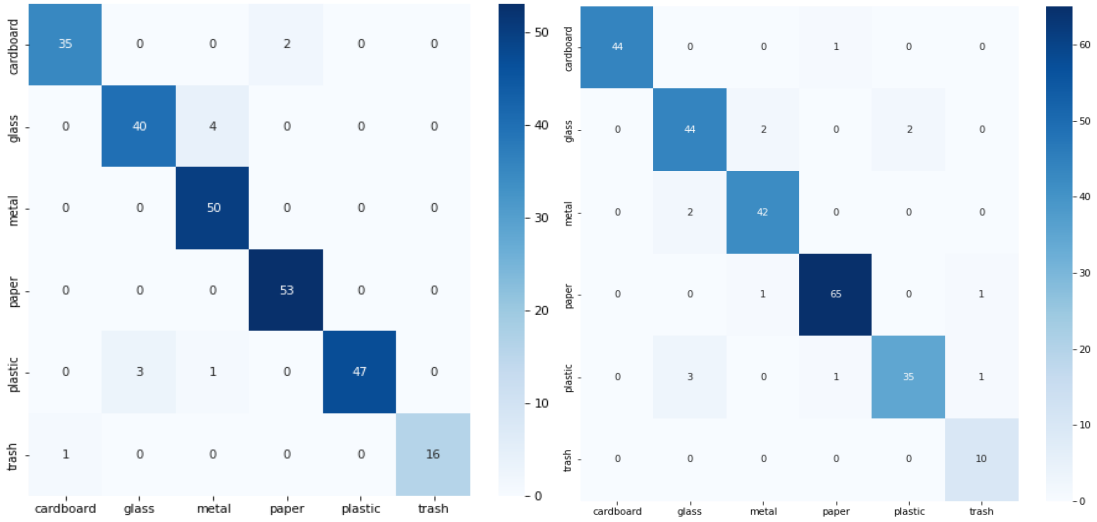


Figure 4.16 Confusion Matrix of ResNet50: 50 Epochs and 100 Epochs

The following plots show the training loss and accuracy when the dataset is divided into 70% training, 15%validation and 15%testing.

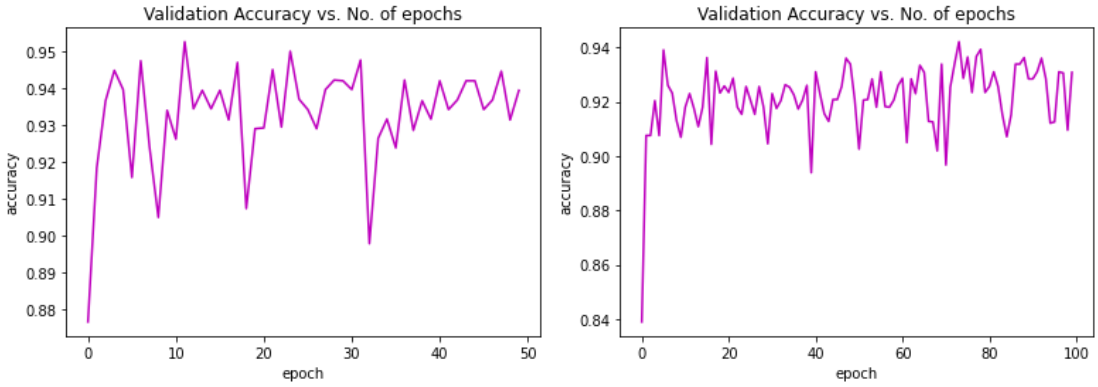


Figure 4.17 Plotting Accuracy of ResNet50: 50 Epochs and 100 Epochs

Figure 4.17 and 4.18 show the plotting of accuracy and loss during the training process.

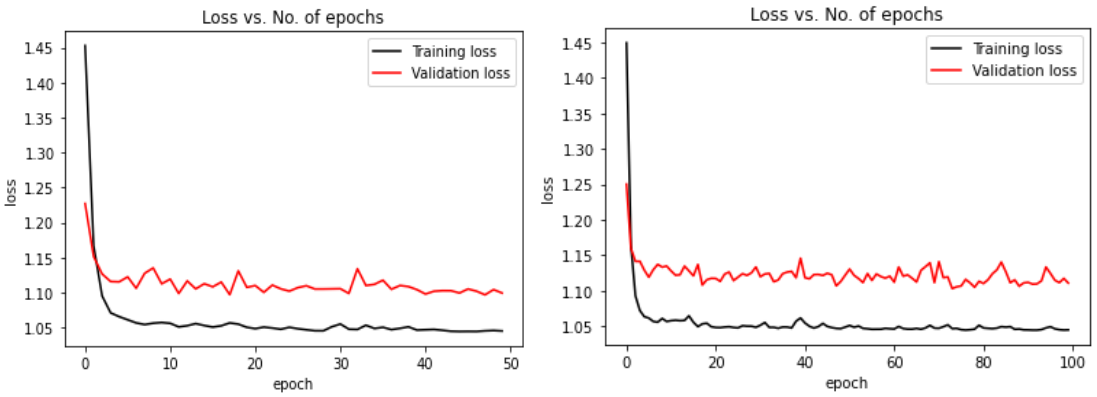


Figure 4.18 Plotting Loss of ResNet50: 50 Epochs and 100 Epochs

	precision	recall	f1-score	support
cardboard	0.95	0.97	0.96	59
glass	0.95	0.92	0.93	59
metal	0.96	1.00	0.98	77
paper	0.94	0.97	0.96	77
plastic	0.96	0.90	0.93	79
trash	0.96	0.96	0.96	28
accuracy			0.95	379
macro avg	0.95	0.95	0.95	379
weighted avg	0.95	0.95	0.95	379

	precision	recall	f1-score	support
cardboard	0.95	0.97	0.96	60
glass	0.94	0.97	0.96	78
metal	0.97	0.98	0.97	58
paper	0.96	0.95	0.96	106
plastic	0.92	0.92	0.92	59
trash	1.00	0.78	0.88	18
accuracy			0.95	379
macro avg	0.96	0.93	0.94	379
weighted avg	0.95	0.95	0.95	379

Figure 4.19 Performance Evaluation of ResNet50: 50 Epochs and 100 Epochs

When testing the test data, the accuracy performances parameter calculation of the model is shown in Figure 4.19. These figures can be seen the comparisons between 50 and 100 epochs. Figure 4.20 shows the confusion matrixes on testing data.

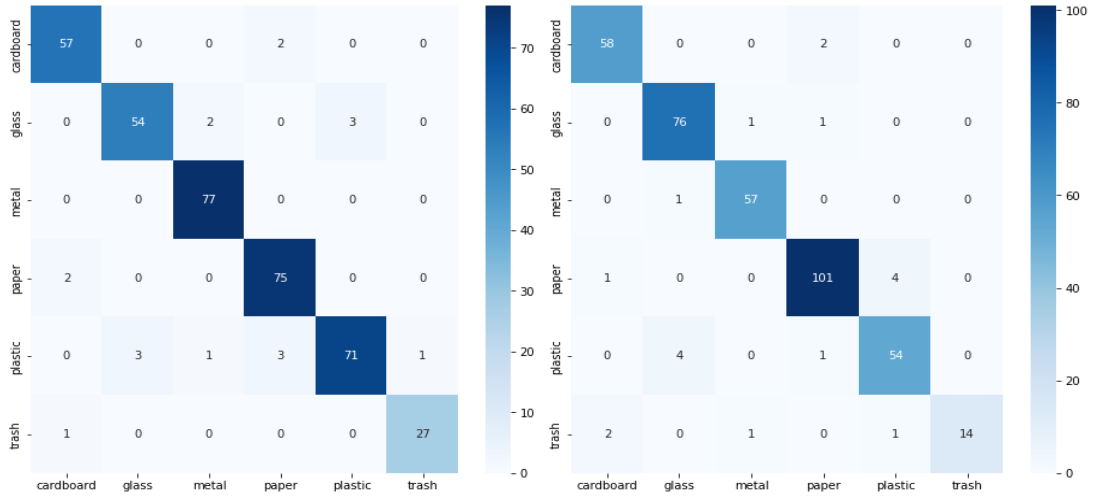


Figure 4.20 Confusion Matrix of ResNet50: 50 Epochs and 100 Epochs

By analyzing the above steps, now these results are summarized and shown in the following Table 4.2. The accuracy analyzing on test data of two different epochs on two different splitting datasets of training.

Table 4.2 Summarizing the Results of ResNet50

	Data Split (80 train /10valid /10 test)		Data Split (70train/15valid/ 15test)	
	Epoch 50	Epoch 100	Epoch 50	Epoch 100
Accuracy	96%	94%	95%	95%

4.3.3 Performance Analysis on Testing Non-Dataset

	precision	recall	f1-score	support
0	0.68	0.48	0.57	27
1	0.75	0.86	0.80	21
2	0.81	0.75	0.78	28
3	0.55	0.72	0.63	29
4	0.77	0.85	0.81	20
5	0.50	0.25	0.33	8
accuracy			0.69	133
macro avg	0.68	0.65	0.65	133
weighted avg	0.69	0.69	0.68	133

	precision	recall	f1-score	support
0	0.86	0.70	0.78	27
1	0.88	0.71	0.79	21
2	0.75	0.86	0.80	28
3	0.70	0.79	0.74	29
4	0.81	0.85	0.83	20
5	0.62	0.62	0.62	8
accuracy			0.77	133
macro avg	0.77	0.76	0.76	133
weighted avg	0.78	0.77	0.77	133

Figure 4.21 Performance Evaluation on Non-dataset of Simple CNNResNet50

The Figure 4.21 show the results of using simple CNN and Figure 4.22 shows the confusion matrixes of model performance.

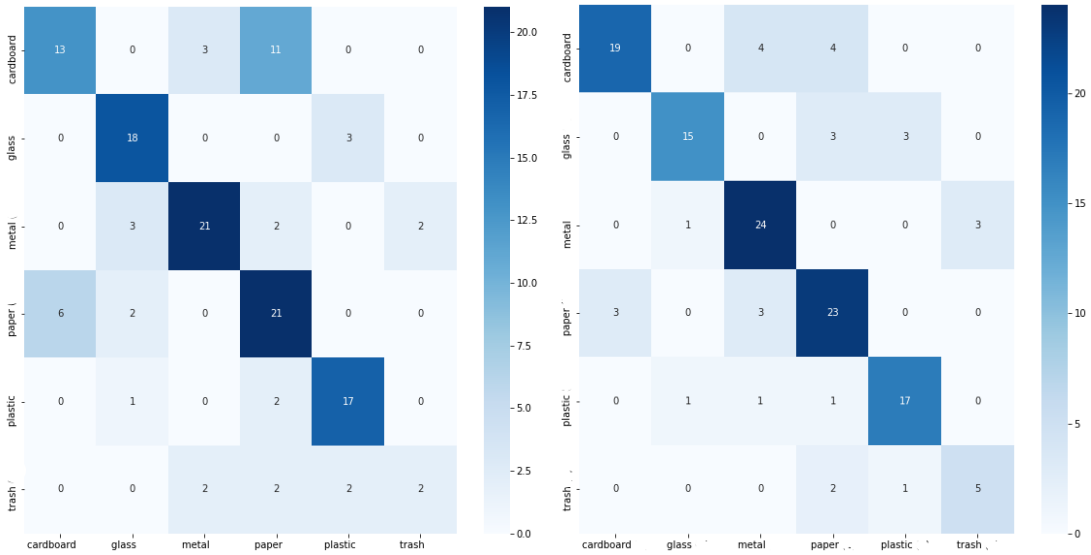


Figure 4.22 Confusion Matrix of Simple CNN and ResNet50

This section shows the testing of another data which is not included in the Kaggle Garbage dataset. These data are collected from the real world. This data is predicted by using prediction models. The prediction results are shown. There are various file types can be tested such as JPG, PNG, GIF, JFIF. The following Figure 4.23 shows the summarizing of results.

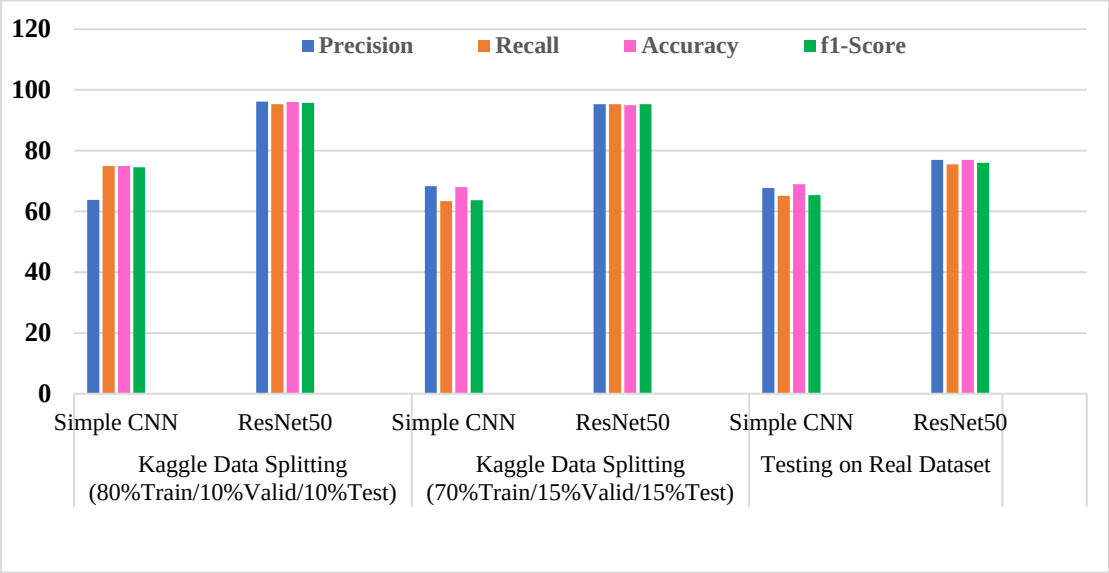


Figure 4.23 Summary of Evaluation Results

4.4 Chapter Summary

The performance evolution of the system in different partitions of dataset in different epochs. The experimental outputs of two models are shown in this chapter. Moreover, the accuracy, precision, recall, f1score, macro average and weighted average are displayed in this chapter. The confusion matrixes of the multi-class's classification are also displayed. By summarization, the comparison between two models in different splitting data in different training epochs can be seen and these analysis results have been described.

CHAPTER 5

CONCLUSION AND FURTHER EXTENSIONS

5.1 Conclusion

In conclusion, the RRR approach is very helpful in garbage control. Reduce waste by recycling and reusing. Public education is crucial. Waste management is an urgent problem that calls for rapid government action. Currently, there is not much awareness of this issue in our society. The practice of producing rubbish is too risky for not only the current generation but also for the generations to come. People must be informed and inspired to practice recycling, reusing and waste reduction rather than creating waste. The management of waste should be a top concern for local governments. Individual participation is crucial. A popular and active topic of research is the separation of waste classification. Several researches have suggested several solutions to this issue including conventional, statistical, and machine learning methods. However, this approach is unproductive since it takes a lot of time and effort to categorize waste by category.

In this study, CNN is used to create a trash categorization model that performs better and is more accurate while making the fewest errors possible. Based on the suggested model, garbage was classified by using the CNN algorithm, which does not require a lot of time or effort to provide the results. In contrast to prior work in the field, this article gives comprehensive information about the model's training utilizing a methodical methodology. These training approaches, which include how to split the dataset, how to choose training parameters and how to choose data augmentation strategies, have been summarized as a framework for training recycling waste picture classification models. These enhancement methods include flipping, rotation, shearing, and zooming.

Following that, the training parameters are chosen by evaluating the validation precision and these criteria are learning patience epochs, loss, and rate schedulers batch sizes, learning rates, and function. That is work, the 5.5e-5 scheduler with continuous learning rate is utilized along with Adam optimizer to maximize the categorical cross-entropy reduction. Secondly, the final classification's node number, the ResNet50 model's SoftMax layer is condensed only the foundation convolutional layers are

locked, with up to 6 classes. Images are then passed in a 32-piece batch size. Finally, if a higher validation accuracy is not attained in the passing 100 iterations, The training program has come to an end. The prediction model is used again in testing data for evaluating the model performance. By comparing simple model and ResNet50 model, the modern convolutional Neural Network is achieving higher accuracy of 96%.

5.2 Advantages and Limitations of the System

By classification of solid trash segregation, it keeps the environment clean and fresh, reduce environmental pollution. Landfills of waste disposal is much smaller. By segregating garbage, it can reduce landfill impact. More materials are recycled and lessen the volume of rubbish that is dumped in landfills. As a result, it has a smaller overall environmental impact. Recycling has several benefits, including protecting the environment and conserving a lot of energy.

In this system, the data of the input has been only read as image. Therefore, it can take some time to classify result. The dataset doesn't have same number of each category. As a result, the trash category has error range than other categories. The dataset is needed to add more images and adjust each category.

5.3 Further Extensions

Moreover, there are many things can be done in the future to attain good outcomes, including:

- Increasing the number of layers in the CNN algorithm.
- Testing of other models.
- Training on more models over a wider range of epochs.
- Adding some additional waste object photos to the dataset.
- Extending the project to allow for deployment on a sensor-equipped device.

This system is needed to intend to concentrate efforts in the future on trash classification at larger inventory systems, which frequently use live streams. To make it possible for waste to be automatically classified at a large industrial scale when objects may be overlapping and traveling through a conveyor belt, this system is intended to upgrade our model and gather more data.

PUBLICATION

[1] Moh Moh Thet Aung and Amy Tun, “Solid Trash Segregation System Using Convolutional Neural Network”, University of Computer Studies, Yangon, Myanmar,2022.

REFERENCES

[1] Aston Zhang, Zachary C. Lipton, Mu Li, and Alexander J. Smola, “Dive into Deep Learning”, Jul 30, 2022.

[2] Dataset-https://www.kaggle.com/datasets/asdasdasasdas/garbage-classification

[3] Dhulekar, P., Gandhe, S. and Mahajan, U. P. (2018). Development of bottle recycling machine using machine learning algorithm, 2018 International Conference On Advances in Communication and Computing Technology (ICACCT), IEEE, pp. 515–519.

[4] ECD, MONREC (2018): National Waste Management Strategy and Master Plan for Myanmar, the Republic of the Union of Myanmar, Nay Pyi Taw, Myanmar.

[5] G. White, C. Cabrera, A. Palade, F. Li, and S. Clarke, “WasteNet: Waste Classification at the Edge for Smart Bins,” arXiv. arXiv, Jun. 10, 2020, Accessed: Jan. 16, 2021. [Online]. Available: https://arxiv.org/abs/2006.05873v1.

[6] Gupta, D., Jain, S., Shaikh, F. and Singh, G. (2017). Transfer learning & the art of using pre-trained models in deep learning, Analytics Vidhya.

[7] H. Zhou, Y. Long, A. Meng, Q. Li, and Y. Zhang, “Classification of municipal solid waste components for thermal conversion in waste-to- energy research,” Fuel, vol. 145, pp. 151–157, 2015.

[8] Himanshu Gupta x18203302, “Trash Image Classification System using Machine Learning and Deep Learning Algorithms”, National College of Ireland, 2022.

[9] https://bdtechtalks.com/2021/11/27/what-is-data-augmentation/.

[10] https://dataaspirant.com/data-augmentation-techniques-deep-learning/.

[11] https://iq.opengenus.org/residual-neural-networks/

[12] https://msatechnosoft.in/blog/artificial-neural-network-types-feed-forward-feedback-structure-perceptron-machine-learning-applications/

[13] https://towardsdatascience.com/understanding-and-calculating-the-number-of-parameters-in-convolution-neural-networks-cnns.

[14] https://towardsdatascience.com/various-ways-to-evaluate-a-machine-learning-models-performance-230449055f15

[15] https://www.xenonstack.com/blog/artificial-neural-network-applications

[16] Lai, Z. and Deng, H. (2018). Medical image classification based on deep features extracted by deep model and statistic feature fusion with multilayer perceptron, Computational intelligence and neuroscience 2018.

[17] M. Yang and G. Thung, "Classification of trash for recyclability status," CS229 Project Report, vol. 2016.

[18] Mohammad Diqi, “Waste Classification Using CNN Algorithms” 1st International Conference on Science and Technology Innovation (ICoSTEC) February, 26 2022. ISBN: 978-623-331-338-4.

[19] Satvilkar, M. (2018). Image Based Trash Classification using Machine Learning Al- gorithms for Recyclability Status, PhD thesis, Dublin, National College of Ireland.

[20] Tarun, K., Sreelakshmi, K. and Peeyush, K. (2019). Segregation of plastic and non-plastic waste using convolutional neural network, IOP Conference Series: Materials Science and Engineering, Vol. 561, IOP Publishing, p. 012113

[21] U. Özkaya and L. Seyfi, “Fine-Tuning Models Comparisons on Garbage Classification for Recyclability,” Proc. 2nd Int. Symp. Innov. Approaches Sci. Stud., 2018.