PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA

Ministry of Higher Education and Scientific Research

UNIVERSITY KASDI MERBBAH OUARGLA

Memory ACADEMIC MASTERS

Field: Science and Technology

Sector: Electronics

Specialty: Electronics of Embedded Systems

Presented by:

❑ **BENKHIRA Belkhir** ❑ **ROUAS Fatah**

Theme

Electronic components recognition based on deep learning

Publicly defended on: June 2023

In front of the jury:

❑ **Mr. YOUCEFA Abdelmadjid Supervisor UKM OUARGLA**

❑ **Mr. CHAA Morad President UKM OUARGLA**

❑ **Ms. BELKBIR Djalila Examiner UKM OUARGLA**

Academic year: 2022 /2023

ABSTRACT

Deep learning methods, particularly Convolutional Neural Networks (CNNs), have revolutionized image classification, including the classification of electronic components, when compared to traditional methods, deep learning methods have several advantages. Firstly, CNNs are capable of automatically learning hierarchical representations from raw input data, eliminating the need for manual feature engineering. This allows CNNs to effectively capture intricate patterns and features in images, enabling accurate classification. Moreover, it excels in handling large-scale datasets.

In this thesis, we focused on investigating the classification of electronic components using deep learning techniques and evaluated the performance of popular models such as LeNet, AlexNet, GoogleNet, and VGG16, leveraging transfer learning by comparing their performance on a small dataset representing four classes of electronic components: bypass capacitor, transistor, LED, and relay. Notably, our findings demonstrated that VGG-16 achieved superior results, exhibiting higher accuracy within a shorter time frame. This outcome highlights the effectiveness of transfer learning, where pre-trained models can be fine-tuned on specific tasks, in improving the classification accuracy of electronic components.

Key words and terms: Machine Learning, Electronic components recognition, Deep Learning, Convolutional Neural Networks, Image Classification.

خلاصة

أحدثت أساليب التعلم العميق ، ولا سيما الشبكات العصبية التلافيفية (CNN) ، ثورة في تصنيف الصور ، بما في ذلك تصنيف المكونات الإلكترونية ، عند مقارنتها بالطرق التقليدية ، تتمتع طرق التعلم العميق بالعديد من المزايا. أولاً ، يمكن لشبكات CNN أن تتعلم التمثيل الهرمي تلقائيًا من بيانات الإدخال الخام ، مما يلغي الحاجة إلى هندسة الميزات اليدوية. يسمح هذا لشبكات CNN بالتقاط الأنماط والميزات المعقدة بشكل فعال في الصور ، مما يتيح التصنيف الدقيق. علاوة على ذلك ، فهي تتفوق في التعامل مع مجموعات البيانات واسعة النطاق.

. في هذه الأطروحة ، ركزنا على التحقيق في تصنيف المكونات الإلكترونية باستخدام تقنيات التعلم العميق وتقييم أداء النماذج الشائعة مثل LeNet و AlexNet و GoogleNet و VGG16 ، والاستفادة من التعلم عن طريق مقارنة أدائها على مجموعة بيانات صغيرة تمثل أربع فئات من المكونات الإلكترونية: مكثف تجاوز ، ترانزستور ، مصباح ، ومرحل. والجدير بالذكر أن النتائج التي توصلنا إليها أظهرت أن VGG16 حققت نتائج متفوقة ، وأظهرت دقة أعلى في إطار زمني أقصر. تسلط هذه النتيجة الضوء على فعالية تعلم النقل حيث يمكن ضبط النماذج المدربة مسبقًا على مهام محددة ، في تعزيز دقة تصنيف المكونات الإلكترونية.

الكلمات والمصطلحات الأساسية: التعلم الآلي ، التعرف على المكونات الالكترونية ، التعلم العميق ، شبكات الذكاء الاصطناعي العصبية الالتفافية ، تصنيف الصور

Dedication

I would like to express my deepest gratitude to my family, especially my mom and dad For Their boundless love and my friends for their support throughout

this Journey I would like to thank my teacher and supervisor Dr. Youcefa Abd El Majid, who expertly guided me in my work Thesis end studies in particular

I would like to express my sincere thanks to the jury

president Mr. CHAA Morad

examiner Ms. BELKBIR Djalila

I express my gratitude to all employees of the electronics department and Telecommunications, Faculty of Electronics, Kasdi Merbah, University of Ouargla

CONTENT

List of Figures

List of Tables

List of abbreviations

ANN: Artificial neural network

CNN: Convolutional Neural Networks

CPU: central processing unit

DL: Deep Learning

GPU: Graphics Processing unit

ILSVRC: ImageNet Large-Scale Visual Recognition Challenge

ML: Machine Learning

NAG: Nesterov accelerated gradient

RGB: red, green, blue

SGD: Stochastic gradient descen

SVM: Support vector machine

PCA : principal components Analysis

LBP: local binary patterrn

VGG: Visual geometry group

YOLO: You Only Look Once

CHAPTER I

GENERAL INTRODUCTION

I.INTROUDCTION

Electronic components are an essential part of modern technology and are used in a wide range of applications, from consumer electronics to industrial machinery. The ability to accurately and efficiently recognize electronic components is critical for many tasks, such as quality control, inventory management, and equipment maintenance. Hence why Electronic component recognition is an important task in the field of computer vision, with a wide range of applications in various industries. Traditional methods of recognizing electronic components(such as principal component analysis (PCA) and support vector machine (SVM)) rely on hand-crafted features and shallow learning models, which often suffer from limitations such as poor low recall , accuracy , scalability and generalization.

In recent years, deep learning has emerged as a promising approach for improving the accuracy and efficiency of electronic component recognition. By learning hierarchical representations of features directly from raw data, deep learning models can capture more complex and abstract patterns that are difficult to extract using traditional methods.

The objective of this thesis is to propose a deep learning-based approach for recognizing electronic components in real-world scenarios. Specifically, we use a convolutional neural network (CNN) to extract features from images of electronic components and apply a classifier to predict the component type. Our approach is trained and evaluated on a large-scale dataset of electronic components, and we compare the performance between popular existing methods like LENET, AlexNet , GoogleNet and VGG16. Additionally, we investigate the impact of various

factors on the performance of our approach, such as the size and quality of the training data, the architecture of the CNN.

I.1 thesis structure:

In **Chapter II**, we review related works on electronic component recognition . We discuss the limitations of existing methods and the advantages of using deep learning for this task

In **Chapter III**, we describe the methodology of our approach. We introduce the dataset that we use for training and evaluation and explain the data preprocessing steps. We also provide a detailed description of the architecture of the CNN that we use and the training procedure. Finally, we explain the classifier that we use to predict the component type and the evaluation metrics that we use to compare the performance of our approach to existing methods.

In **Chapter IV**, we present the experimental results and analysis. We report the accuracy and efficiency of our approach and and we compare the performance of popular CNN network's such as AlexNet ,LENET, GoogleNet and a pretrained VGG16 , we provide qualitative analysis of the results by visualizing the learned features of the CNN and analyzing the errors made by the classifier.

Finally we conclude the thesis and discuss the contributions and limitations of our approach. We also suggest future research directions in this area, such as incorporating additional modalities (e.g., sound) to improve the accuracy of component recognition.

CHAPTER II

RELATED WORKS

II.Related work

Electronic component recognition is a critical task in the electronics industry, where the ability to accurately and efficiently recognize and classify components is essential for quality control and inventory management. In recent years, there has been a significant increase in research focused on developing effective methods for electronic component recognition, with a particular emphasis on the application of deep learning techniques. In this section, we will review the related works on traditional methods and deep learning methods for electronic component recognition.

The majority of current Image classification techniques fall into two categories. Images are primarily categorized in the first group according to their spatial domains and transforms domains. The second category applies deep learning to categorize photos. Convolutional neural networks (CNNs) and other networks are used to automatically learn image attributes.

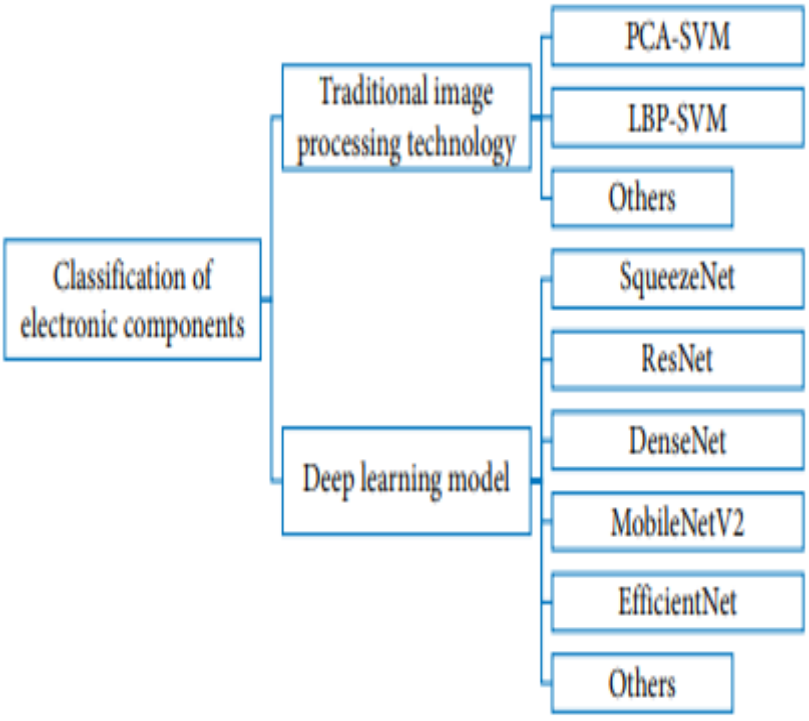

Figure 1: Electronic component classification methods

II.1 TRADITIONAL METHODS

Traditional techniques for classifying images. For many years, researchers have explored and developed traditional picture categorization techniques. These techniques identify the images based on the features they have extracted through a number of difficult picture preprocessing procedures (such morphological transforms). For instance**, Du et al** [2]. utilized the Hough transform and least squares method to classify components by extracting their edge features. Traditional image classification algorithms cannot analyze very large images due to computational cost, and multiple component classification is challenging even if classification accuracy is generally high.

In previous studies such as [3]by D. Lefkaditis and all , This paper gives an account of the construction of an intelligent sorting system for electronic components. Specific focus is given on the comparison of two feature selection methods used to optimize the morphological feature vector. Correlation analysis and support vector machines were considered as they represent two very different approaches for feature selection. The performance of these methods was measured through the successful recognition rates of two neural classifiers, the multilayer perception and the radial basis function network. The best performing combination of methods was then proposed as the classification module of the sorting system. Correlation analysis was utilized to discover any underlying relationships between the features, in essence to find out if they refer to the same property of the sample's outline. Therefore, unwanted repetitions of information were discarded, thus the feature vector was shrunk and simplified with minimum loss of its descriptive ability.

Support vector machines (SVMs) were examined as an alternative technique for feature selection. Here, SVMs were set to perform a supervised classification task on the data. The focus was not to optimize the classification performance but to quantify the discrimination ability of the features and sort them on this basis. The sorting criterion was the squared weight of each variable defined by the support vectors.

In [4] Fast Classification in PCA+SVM Settings PCA is often used to project input samples into a (generally lower dimensional) space where classification is carried out. This is specially useful when the input samples are images. Basically, PCA gives

a set of orthogonal dimensions that maximize the variance of the input samples. In face recognition, this set is called eigenfaces Not all of these dimensions (eigenfaces) are useful for classification. Only the first n eigenfaces are appropriate for classification, with the last eigenfaces typically encoding noise.

When a test sample X is to be projected with PCA, the operation to perform is: Y = XW, where W is the transform matrix. When working with vectorized images in the rows of X, the columns of W are the eigenfaces. As mentioned above, usually only the first n columns of W are used in the multiplication. This is thought to avoid the noise of the last eigenfaces. What should be a good value for n? There are reasons to believe that a large dimensional input space would be needed to separate difficult samples, while 'easy' samples could be separated in simpler spaces (i.e. lower value for n). In this method, a different K value may be used for each specific test sample, instead of a fixed dimension n. Easy samples can be classified in a PCA space of a low number K of dimensions. The necessary number of dimensions to use will be ultimatelygiven by the classifier output. For each test sample the system would classify it in a low dimensional space first. If the classifier output is large enough (i.e. above a fixed threshold) then classification will end and a class label will be retrieved. Otherwise the process should be repeated in a more informative space of a larger dimension, see Figure 2.

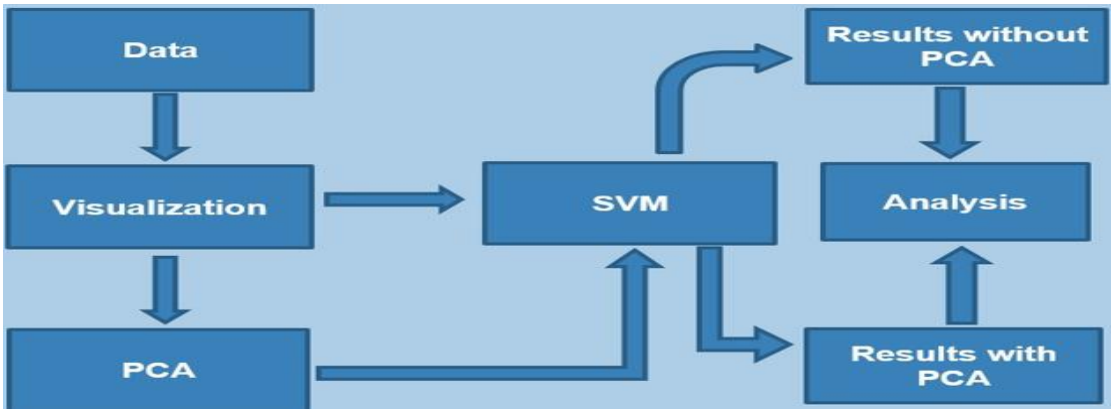

Figure 2 : Top: Typical PCA+SVM classification [4]

Traditional machine learning methods, such as PCA and SVM, have been used for electronic component recognition. PCA is a dimensionality reduction technique that projects high-dimensional data onto a lower-dimensional space. SVM is a popular machine learning algorithm that can be used for classification tasks. However, these

methods have limitations when it comes to recognizing complex images with high variability , such as electronic components.

In **[5]** by Ravi Kumar and all ,The Local Binary Pattern (LBP) method is a feature extraction technique that has been used for electronic component recognition. LBP extracts texture features from an image by comparing the intensity of each pixel in the image to the intensities of its surrounding pixels. The method creates a binary code for each pixel based on whether its intensity is greater than or less than the intensities of its surrounding pixels. By applying LBP to an image, we obtain a binary code for each pixel, which can be represented as a histogram of local patterns .

For electric components recognition ,LBP can be used to extract features from images of electronic components. These features can be used as input to a classifier, such as a support vector machine (SVM) or a neural network, to predict the type of component in the image.

he DS-1D-LBP method has some significant advantages. The first advantage is that this method uses individual values in all marks for feature extraction. The second advantage is that the implementation of this model is easy and fast. Another advantage is that it can extract different feature groups depending on the window length (WL) and related sampling parameters. they tested their proposed DS-1D-LBP + ELM approach, the data set in the Kaggle database repository was used. High performance was obtained for activity recognition by using DS-1D-LBP + ELM approach. However, LBP also has some limitations, such as its sensitivity to noise and its inability to capture spatial relationships between pixel.

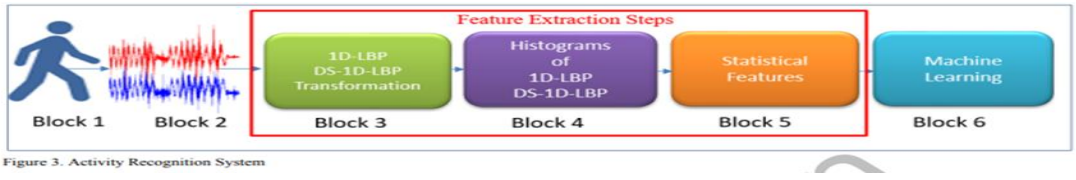

Figure 3: Local Binary Pattern strecture

in [6] by M. Moetesum and others, This paper presents an effective technique for segmentation and recognition of electronic components from hand-drawn circuit diagrams. Segmentation is carried out by using a series of morphological operations on the binarized images of circuits and discriminating between three categories of

components (closed shape, components with connected lines, disconnected components).

Each segmented component is characterized by computing the Histogram of Oriented Gradients (HOG) descriptor while classification is carried out using Support Vector Machine (SVM). The system is evaluated on 100 hand-drawn circuit diagrams with a total of 350 components. A segmentation accuracy of 87.7% while a classification rate of 92% is realized demonstrating the effectiveness of the proposed technique.

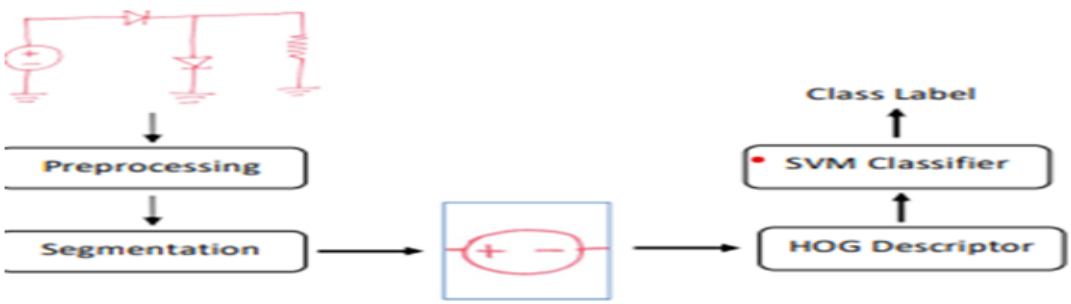

Figure 4 : An overview of the system

Traditional methods of classifying images, such as principal component analysis (PCA) and support vector machine (SVM), have some limitations and drawbacks. One major limitation is their reliance on hand-crafted features, which can be time-consuming and subjective. Additionally, they may not perform well when faced with complex images with high variability, such as electronic components. These methods may also suffer from over fitting or under fitting, leading to poor generalization to new data. Lastly, these methods may not scale well to large datasets and may require significant computational resourcesIn addition to the lack of accuracy and taking a lot of time.

II.2 DEEP LEARNING METHODS

Deep learning methods using convolutional neural networks (CNNs) can automatically extract information from images by utilizing techniques such as convolutions, backpropagation, weight sharing, sparse connections, and pooling. However, general deep learning networks may struggle with handling large amounts of data and operating in real-time due to their complex structure and numerous calculations.

Recent studies, such as [7] by IpekAtik proposed a new CNN model with six convolution layers, four pooling layers, two fully connected layers, softmax , and a classification layer, The training parameters of the network were determined as an ensemble size of 16 maximum period of 100, initial learning rate of 1 × 10−3 which shows excellent performance when optimized with the Sgdm method with accuracy value of 98.99% compared to selected pre-trained models (Google Net shuffle Net...etc) but with its very large parameters and complex architecture it took longer time. .

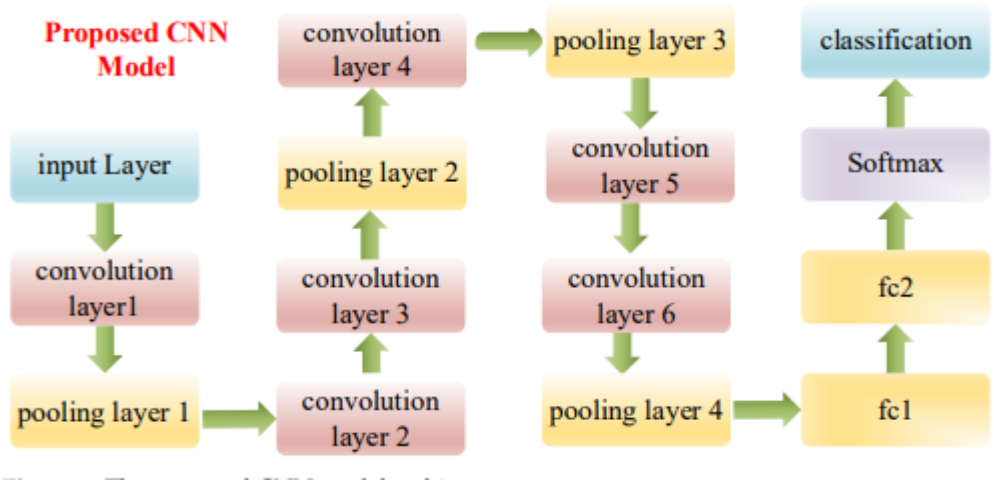

Figure 5 : The proposed CNN model architecture[7]while in [8] by Xu and others , they used a Faster SqueezeNet network algorithm which can reduce the size of network parameters and computational complexity without deteriorating the performance of the network along with tensorflow techniques achieved a reasoning time is about 2.67 ms and 99% accuracy for the industrial application level in terms of time consumption and performance. Since the test only used small data set (22

subcategories of resistor, capacitor and inductor) it can't be taken for granted when it comes to larger amount of dataset and various categories of components.

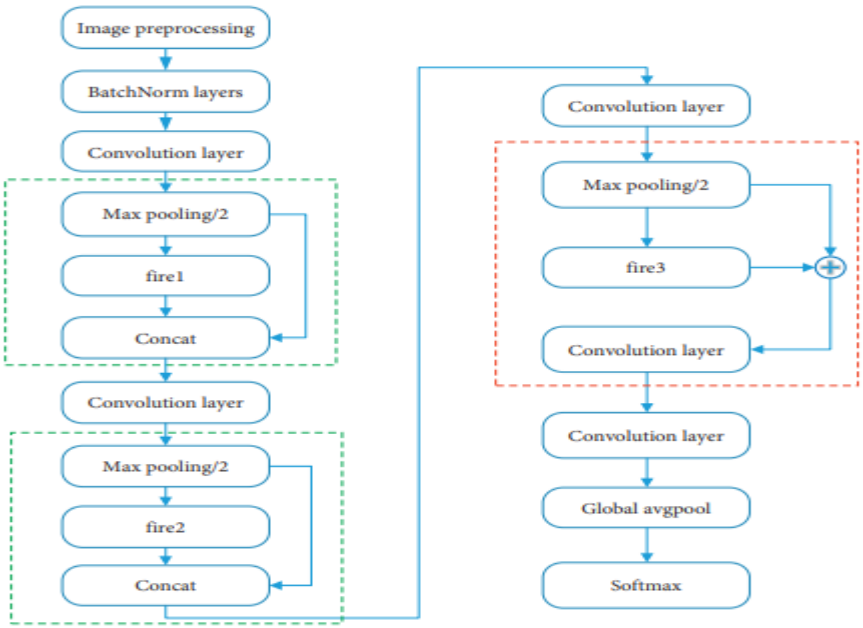

Figure 6: Faster SqueezeNet network structure.[8]

In [9] by Rui Huang, Jinan Gu, Xiaohong Sun, Yongtao Hou and Saad Uddin Introduced a fast recognition method which is an improved YOLO (You Only Look Once)-V3 network model by the lightweight technique that resulted on good detection accuracy and speed (accuracy 95.21% and the speed was 0.0794 s) but there still some a bit gab between them.

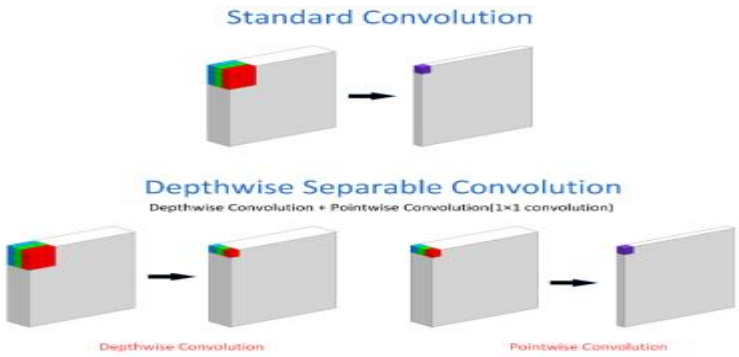

Figure 7: Standard convolution vs. depthwise separable convolution [9]

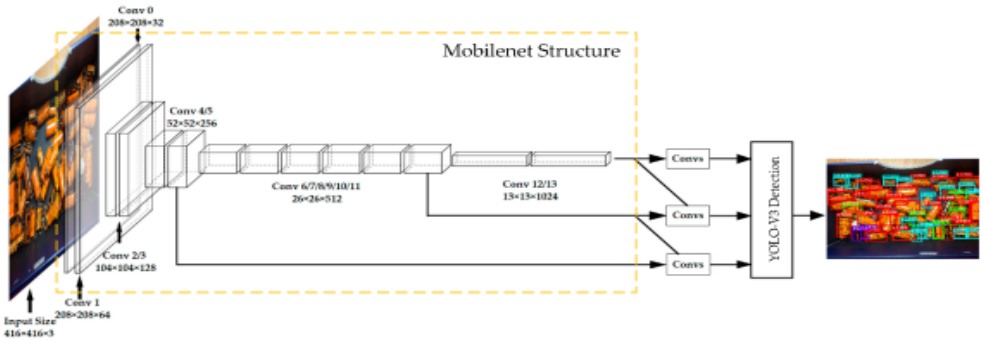

Figure 8: YOLOV3–Mobilenet detection network [9]

In [10] by Sun, Gu and Rui ,they provided a rapid recognition method known as an improved model based on SSD (single shot multi-box detector) which is conducted by adopting feature fusion strategy and adding visual reasoning techniques showed an great balance between accuracy and detection speed like on small dataset only which might degrades on a large dataset.

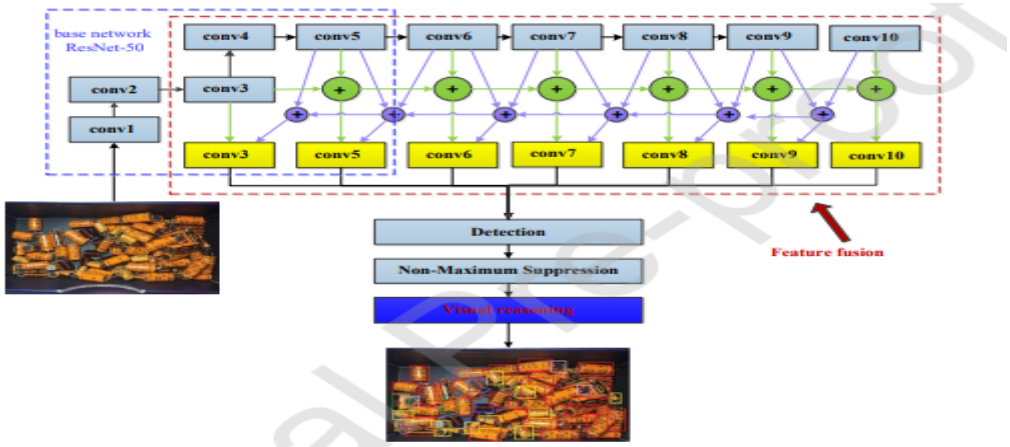

Figure 9 : The improved SSD [10]

in [11] by Tianhong Panand Mian Khuram Ahsan a hand-drawn electronic component recognition method was proposed using a Convolutional Neural Network (CNN) and a soft max classifier is proposed. The CNN composed of a convolutional layer, an activation layer and an average-pooling layer is designed to extract features of a hand-drawn electronic component image with a sparse auto-encoder method has achieved 95% .

recognition accuracy on rotating images most of these deep learning methods has showed Impressive evolution electronic component recognition compared to

traditional methods in terms of performance in real time industrial level of requirements (accuracy and time) .

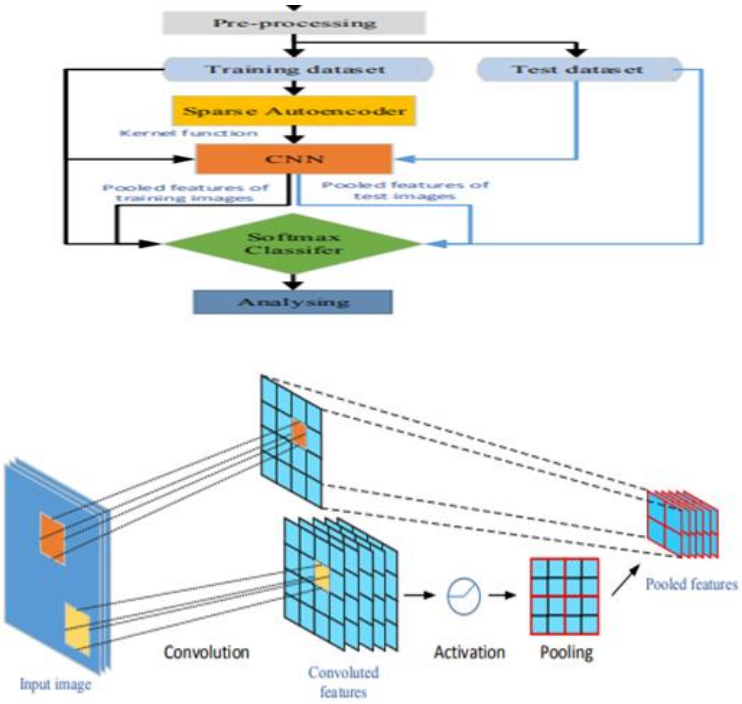

Figure 10 : Hand-drawn model[11]

conclusion

In this thesis , we aim to compare different CNN-based methods, including AlexNet, VGG16, Google Net, Lenet with vgg16 trained with transfer learning technique and the others models from scratch . We will then select the best one to identify electronic components in industries real-time requirements.

CAHPTER III

METHODOLOGY: CONVOLUTIONAL NEURAL NETWORK (CNN)

III.METHOLOGY

III.1 Artificial neural network

Artificial neural networks are a programming paradigm with biological inspiration that enables a computer to learn from empirical data. Artificial neurons, which are a network of interconnected units or nodes loosely fashioned after the human brain and intended to recognize patterns, form the foundation of a [ANN]. Neural networks are capable of labeling, categorizing, or grouping unprocessed inputs to interpret sensory data. They need to translate the patterns that are digital, represented in vectors, and made up of real-world data like photos, sound, text, or time series. They are used to categorize, group, and group unlabeled data based on similarities between sample inputs. When they have a labeled data set to train on, they also classify data. They can extract features that are fed into other clustering and classification algorithms.

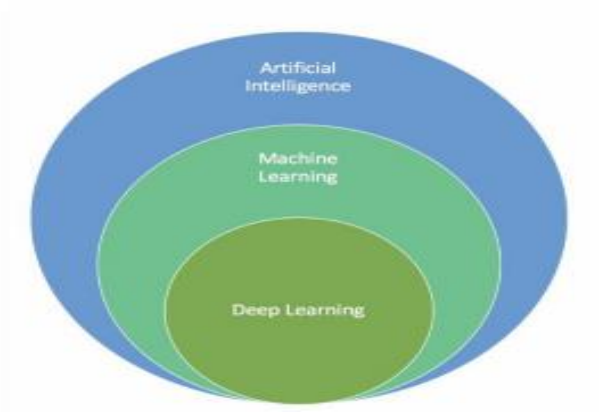

Figure 11: Conceptual hierarchy of Artificial Intelligence and its subsidiaries [41]

III.2 Deep neural network

The neural network was developed with inspiration from the human brain, its functions, and how it operates. Its functionality depends heavily on artificial intelligence and machine learning, a subset of AI. When a developer enters data and creates a machine learning algorithm, they largely employ the ("if...else...") concept of programming to make the system operate. Along with following the procedure, the deep neural network also uses its prior knowledge to forecast solutions to problems and make inferences. You can obtain an answer in this situation without using programming or coding. When autonomous labor needs to replace human labor without sacrificing efficiency, a deep neural network is advantageous. Numerous real-world uses for deep neural networks can be found. For instance, the Chinese business Sensetime has

developed an automatic facial recognition system to discover criminals in crowds using real-time cameras. In the present day, the police and other government agencies have adopted it as a common practice.

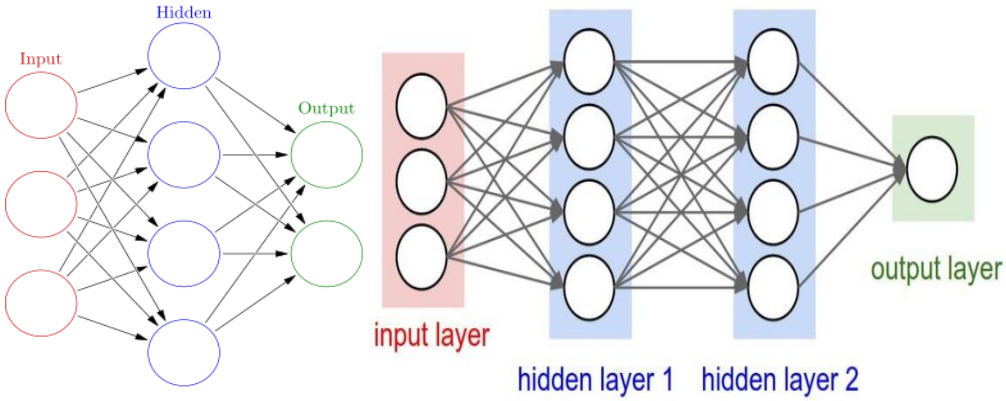

Figure 12: (a) simple neural network and (b) Deep neural network [13]

III.3 COUNVOLUTIONAL NEURAL NETWORK (CNN)

Different neural network types exist, and what sets them apart from one another is how they operate, how they are organized, and in what contexts they are used. Convolutional neural networks (CNN) are primarily employed for image recognition and infrequently for audio recognition. Since it is not necessary to verify each pixel individually, photos are where it is most commonly used. The upper left corner of an image is where CNN starts to check it, going pixel by pixel until it passes. A convolutional layer is then applied after each check, where some data points have connections while others do not. Based on this information, the system may output the results of the checks and determine what is depicted in the image.

Conv nets, also known as convolutional neural networks (CNNs), are a particular type of feed forward neural network. In that they are composed of neurons with trainable weights and biases, The fundamental distinction is that we can encode certain features in the CNN design since it implicitly assumes that the input is image-like Convolutions in particular capture translation invariance (i.e., filters are location independent) As a result, the forward function is more effective, there are much less parameters, the network is easier to improve, and the dependence on the amount of the data is reduced Unlike conventional neural networks, CNNs' layers feature neurons arranged in a few different dimensions, including channels, width, height, and number of filters in the most basic 2D example Similar to an MLP, a convolution neural network

consists of a series of layers, where each layer modifies the activations or outputs of the layer before it using a differentiable function.

The convolution layer, pooling layer, and fully connected layers are the most typical building blocks you will find in most CNN architectures. There are other layers used in CNNs as well, and they will be covered in following sections. These layers essentially function as dimensionality reduction, feature extractors, and classification layers, respectively. These CNN layers are stacked to create a full convolutional neural network We briefly halt at the convolution layer before moving on to a summary of the various layers. In essence, a convolution layer filters the input using a convolutional kernel. These filters typically come in large numbers A filter glides across the input volume during a forward pass, generating its activation map at that location by calculating the pointwise product of each value and combining them to get the activation at the point.

Convolution naturally implements such a sliding filter, and as convolution is a linear operator, it may be efficiently expressed as a dot-product This implies that while training a CNN of this type, the network will discover filters that recognize specific types of visual data, such as edges, orientations, and eventually, in a higher layer of the network, whole patterns. We have a vast array of such filters in each of these convolution layers, and each of these filters will result in a unique activation map. To create the output map or activation volume of this layer, these activation maps are stacked CNN generally. Convolutions are employed in CNN to learn higher-order features from the data. With regard to object recognition in photos, they do remarkably well [12]. CNN may also be utilized for character recognition in text analysis and for analyzing words as separate textual units. CNN also utilizes voice data effectively. For image recognition, CNN is more well-known. Today, CNN is utilized in a variety of systems, such as autonomous vehicles, robots, and drones. Data that has some structure and spatial correlation typically performs well when using CNN. CNNs are employed in computer vision, as seen in Figure 13.

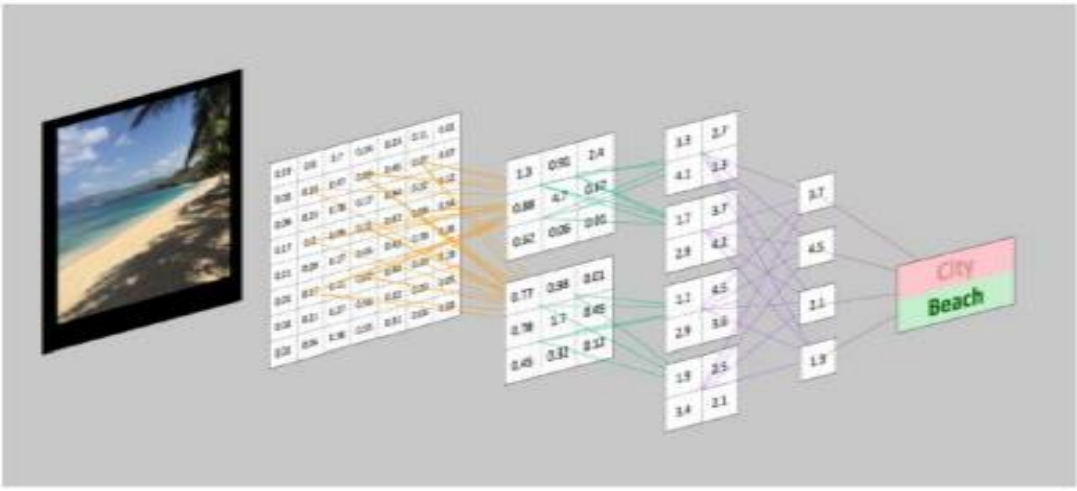

Figure 13: CNNs and computer vision [12]

The inefficiency of classic neural networks when dealing with large amounts of picture input is the basis for CNN [11]. Network architecture can be changed with the help of image data. The length, height, and depth properties of CNNs can therefore be mapped to the image width, height, and RGB channels, allowing the neurons to be aligned in a three-dimensional structure. To put it simply, CNNs take an input image and convert it through a series of connected layers to produce a collection of class probabilities. As seen in Figure 14 CNN architectures will have some similar layers.

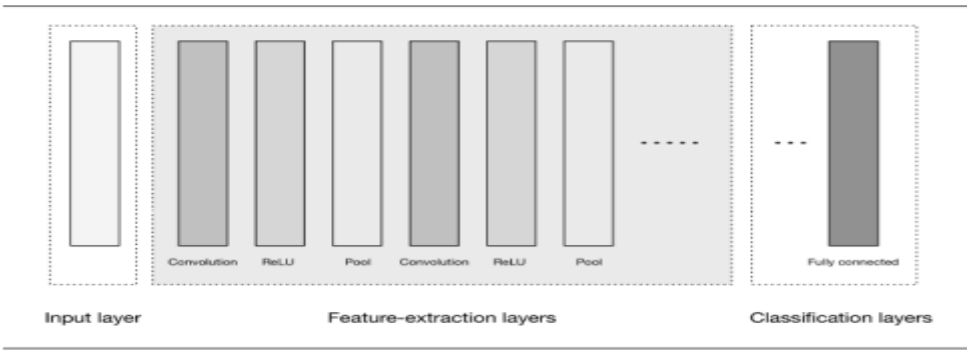

Figure 14: High-level CNN architecture [12]

III.3.1 convolution

A "kernel" is used in convolution to pull out specific "features" from an input image. A kernel is a matrix that is applied to the image and multiplied by the input in order to enhance the output in a desired manner. See an example of this below.

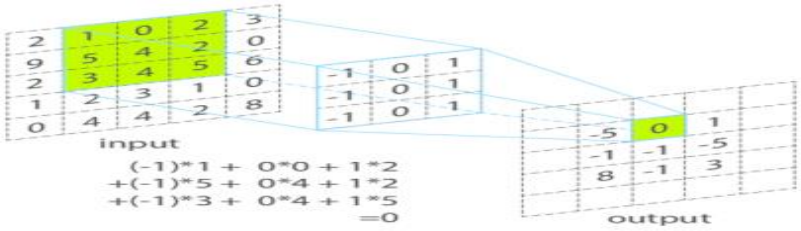

Figure15: how convolution is performed on an input image to extract features [19]

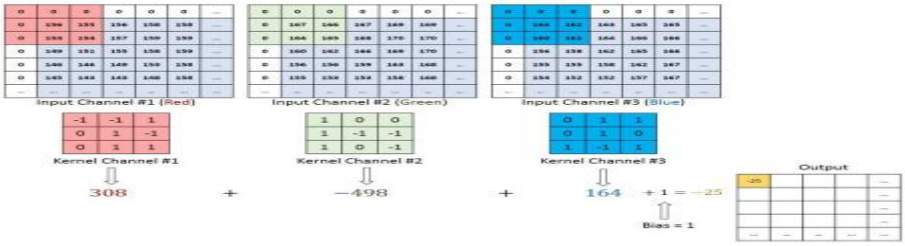

Figure 16: Convolution operation on an MxNx3 image matrix with a 3x3x3 Kernel [18]

III.3.2 CNN Base architecture

The base architecture consist of input layer , convolution layer . pooling layer and fully connected layer

III.3.2.1 Input layer

The CNN input data is loaded to the input layer for processing, as shown in figure 17- The input layer uses the image's three dimensions (width,height, and RGB channel)

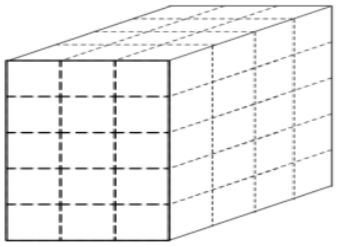

Figure 17: 3D data input [12]

III.3.2.2 convolutional layers

The first layer utilized to extract the different features from the input photos is this one. Convolution is a mathematical process that is carried out at this layer between the input image and a filter of a specific size, MxM. The dot product between the filter and the portions of the input image with regard to the filter size (MxM) is taken by sliding the filter across the input image.The result is known as the Feature map, and it provides details about the image, including its corners and edges. This feature map is later supplied to further layers to teach them additional features from the input image Once the convolution operation has been applied to the input, CNN's convolution layer passes the output to the following layer. The spatial link between the pixels is preserved thanks to convolutional layers of CNN.

The foundation of CNN designs are convolutional layers. Applying a filter or kernel to the input image causes convolution layers to transform it. To create the feature map, the layer executes a dot product operation between the filters and the region of the input layer's neurons. Convolution layer with input and output volume is shown in Figure 18.

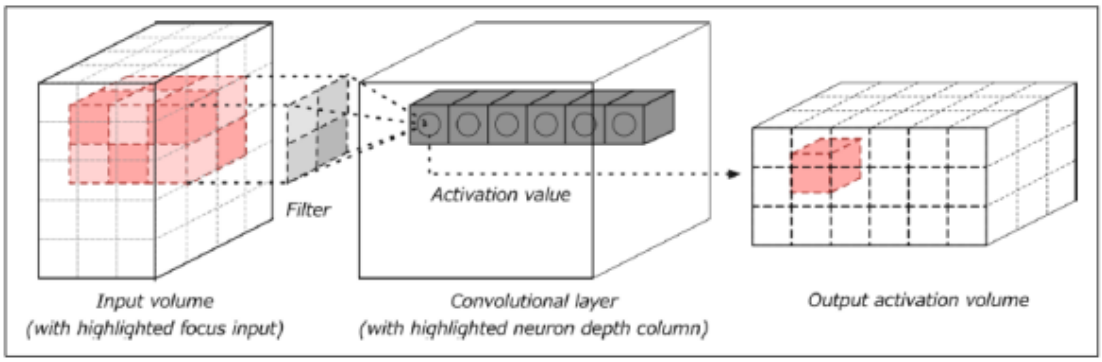

Figure 18: Convolution layer with input and output volume [12]The output generated after the convolution has the same usually as the input seen in figures 19

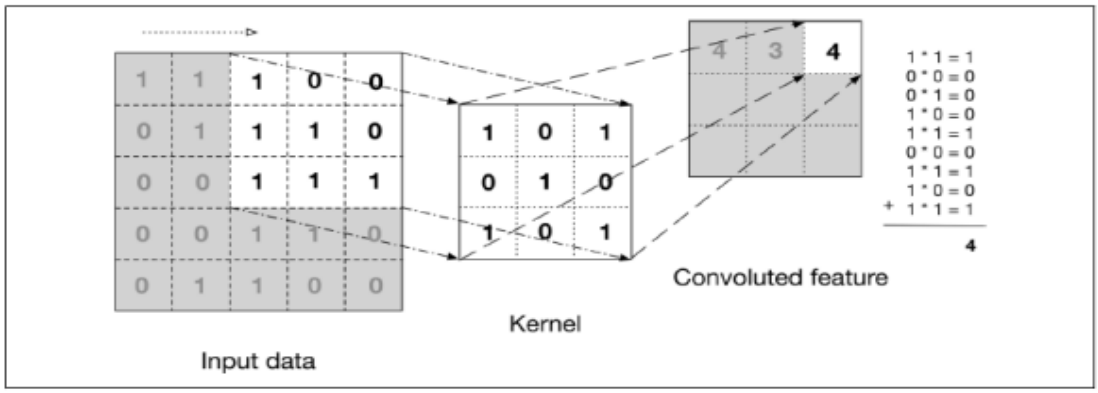

Figure 19: The convolution operation [12]

Figure 19 illustrates how the filter or kernel is smaller than the input size slid. It produces a convolved feature by applying a specified stride value to the input data. Feature detector is another name for this procedure. The 3D output is created by adding the feature map or activation map (as shown in figure 20) for each filter along the depth axis. The activation value reflects the feature detector's learning process. Each filter thereafter learns to recognize a certain feature. A two-dimensional filter activation map is created by sliding the filter on the input.

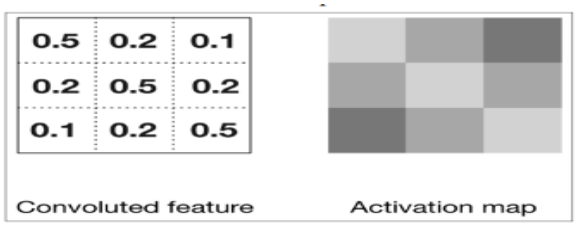

Figure 20 Convolution and activation maps [12]

The staked activation maps form output volume. The values in activation volume correspond to neurons outputs that cover a small area of the input volume

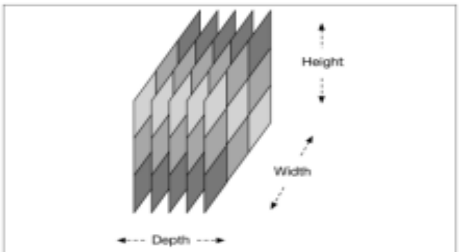

Figure 21: Activation volume output of convolutional layer [12]

Neurons are linked to the input layer through the eceptive field. The size of filter maps is set using

III.3.2.3 The pooling layer

In order to make feature maps smaller, groups of layers are used. As a result, it lessens the amount of network computation and the number of parameters that must be taught The feature map created by a convolution layer's feature clustering layer lists the features that are present in a specific area. As a result, rather than using the precisely positioned features produced by the convolution layer, other operations are done on the summarized features. As a result, the model is more resistant to changes in the features' positions in the input image.

III.3.2.3.1 Pooling layer types

Max Pooling is a pooling process that chooses the most objects from the area of the feature map that the filter covers a feature map comprising the most crucial features from the prior feature map would be the output following the maximum clustering layer.

The average of the elements in the feature map area filtered by the filter is determined via average pooling. Thus, average pooling provides the average of the features present in a patch whereas maximum common implementation provides the most significant feature in a certain patch of the map of features.

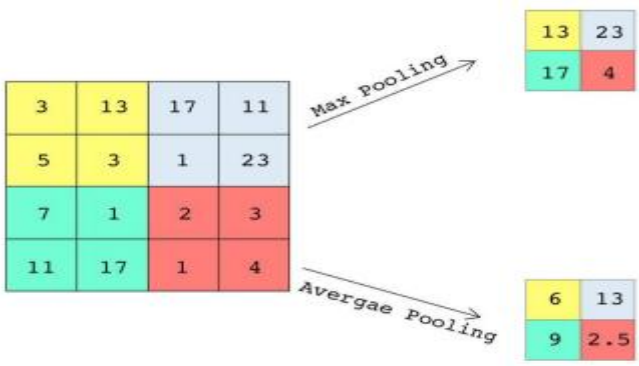

Figure 22: Illustration of maximum pooling and average pooling [17]

III.3.2.4 Fully connected layers

To connect the neurons between two layers, the Fully Connected (FC) layer, which also includes weights and biases, is utilized. These layers make up the final few layers of a CNN architecture and are often positioned before the output layer.

The input image from the layers below is flattened and supplied to the FC layer in this. The flattened vector is then put through a few additional FC layers, where the standard operations on mathematical functions happen. The classification procedure starts to take place at this point. Because two fully connected layers will function better than one connected one, two layers are connected. These CNN layers lessen the need for human oversight.

III.3.3 Transfer learning

Machine learning uses transfer learning as a key approach to address the fundamental issue of insufficient training data. By relaxing the presumption that training data and test data should be distributed equally, it attempts to transfer knowledge from the source domain to the target domain. This has a profoundly good impact in a variety of areas. The figure 23 depicts the transfer learning process.

Figure 23: Learning Process of Transfer Learning [15]

III.3.4 ImageNet

The fundamental driving force behind deep learning is the need for vast volumes of labeled data for supervised models, which may be highly challenging given the time and effort needed to label data points. Most models that address complicated problems require a lot of data. The ImageNet dataset, which includes millions of images falling under several categories, serves as a straightforward illustration.

Figure 24: A sample of the ImageNet dataset [16]

III.3.5 Tuning Hyperparameters

Deep learning models are parameterized such that their behavior can be changed from data for a specific task. He will be able to change the model's parameters by training it with existing data. However, there is another type of parameter known as hyperparameters that cannot be learned directly from the standard training procedure. They are usually corrected before the actual training process begins. These parameters represent crucial model aspects such as the model's complexity or the rate at which it must learn.

A model parameter is an internal configuration variable of the model whose value can be inferred based on the data presented. The model requires them to make predictions. The values of these parameters are learned from data and define the model's performance on a given problem. They are frequently not set by the practitioner manually and are saved as part of the learnt pattern. Weights in a neural network, support vectors in a support vector machine, and linear or logistic regression coefficients are examples of model parameters.

A hyperparameter is a constant parameter whose value is determined prior to the start of the learning process and cannot be approximated from data. Learning is used to derive parameter values, which are then employed in procedures to help estimate model parameters. The values may be mutually dependent. These parameters are frequently modified for a particular predictive modeling task. Learning rate, number of masked layers, and batch size are examples of hyper parameters.

In general, no best values for a model hyper parameter on a specific problem are known. A practitioner can utilize rules of thumb, copy values from other problems, or trial and error to obtain the best value. When you tune a machine learning algorithm for a specific problem, you're effectively tuning the model's hyper parameters to find the model parameters that produce the best accurate predictions. These parameters will aid the model's learning and convergence. The sections that follow will go over the hyper parameters that were employed in this project to optimize performance on the Identify and Classify Electronic Components dataset.

III.3.5.1 Learning rate

The learning rate algorithm. Depending on the optimizer used for the neural network, the model can employ Xe learning rate, steadily decreasing learning rate, momentum-based approaches, or adaptable learning rates.

III.3.5.2 Epoch

The number of epochs is the number of times the training is processed by the neural network. The model's number of epochs should be increased until there is a modest gap between the test error and the training error.

III.3.5.3 Batch Size

In the training of a convolutional network, the mini-batch method is commonly preferred. A range of 16-128 is a decent place to start when testing. Typically, CNN are batch size dependent ,In this experiment, leads were created to train with batches ranging from 16 to 256. However, there is a catch. The larger the batch size, the more accurate the validation, but there is a catch. When the batch size is increased, the training time is reduced and the quantity of learning is reduced when compared to a smaller batch size. Because the training was done on GPUs, a batch size of 64 was enough to train the network.

III.3.5.4 Optimizer

Optimize makes changes to the weight settings in order to reduce the loss function. The loss function functions as a guide to the terrain, informing the optimizer whether it is heading in the proper direction to achieve the valley's bottom, the global

minimum. Adam, Adagrad, Nadam, Nesterov[12] Accelerated Gradient (NAG), RMSprop, and SGD are examples of optimizers. In most circumstances, Adam performs admirably.

Adam can be thought of as a hybrid of RMSprop and Stochastic Gradient Descent with momentum. It uses squared gradients to scale the learning rate, similar to RMSprop, and it takes advantage of momentum by using the gradient's moving average rather than the gradient itself, similar to SGD with momentum. It is an adaptive learning rate method that determines individual learning rates based on various characteristics. It gets its name from adaptive moment estimation, as it employs estimates of the gradient's first and second moments to adjust the learning rate for each weight in the neural network.

III.3.5.5 Fine tuning

Boost performance even further. Fine tuning during transfer learning entails unfreezing part or all of the layers of the pre-trained model and allowing it to adapt more to the task at hand. This project's scope includes transfer learning and the usage of pre-trained models trained on ImageNet classes. It employs ImageNet pre-trained weights and adjusts for the properties of electronic components. set of data We will employ a typical strategy for fine-tuning transfer learning As illustrated in Fig. 25, fine tuning consists of the four phases listed below.

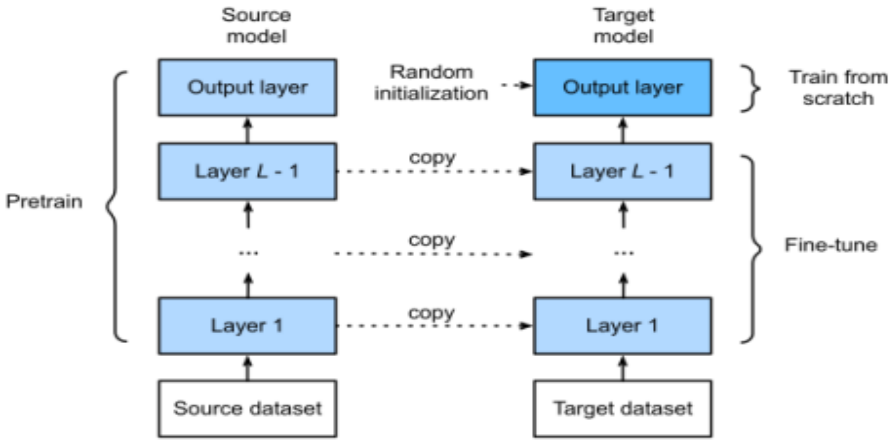

Figure 25 : Fine tuning. [14]

1.Train a neural network model (the source model) on a source dataset (the ImageNet dataset).

.2.Create a new neural network model, which will serve as the target model. Except for the output layer, this transfers all model designs and their settings to the source model. We expect that these model parameters include the information gained from the source dataset and that this knowledge will be applied to the target dataset as well. We also assume that the source model's output layer is strongly related to the labels in the source dataset; thus, it is not employed in the target model.

.3.Add an output layer to the target model with the same number of outputs as the target dataset's categories. Then, at random, set the model parameters for this layer.

4.Train the target model on the target dataset, for example, a chair dataset The output layer will be trained from scratch, while the parameters of all other layers will be refined using the source model's parameters.

III.3.5.6 Dropout

Normally, overfitting in the training dataset might result from all features being connected to the FC layer. When a given model performs so well on training data that it has a negative effect on the model's performance when applied to new data, this is known as overfitting. To solve this issue, a dropout layer is used, in which a small number of neurons are removed from the neural network during training, reducing the size of the model. Thirty percent of the nodes in the neural network are randomly removed upon passing a dropout of 0.3 A machine learning model performs better thanks to dropout since it reduces overfitting by simplifying the network. During training, neurons are removed from the neural networks.

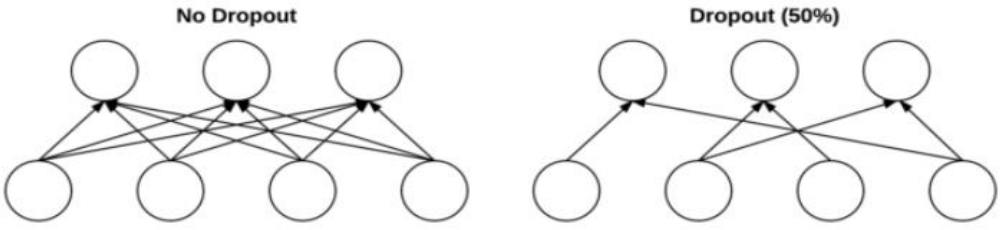

Figure 26: Left: Two layers of a neural network that are fully connected with no dropout. Right: The same two layers after dropping 50% of the connections[13]

III.3.5.7 Activation functions

The activation function is one of the most crucial elements of the CNN model. They are employed to discover and approximation any type of continuous and complex link between network variables. Simply said, it determines which model information should shoot forward and which information should not at the network's end.

The network gains nonlinearity as a result. The ReLU, Softmax, tanH, and Sigmoid functions are a few examples of regularly used activation functions. Each of these operations has a particular use. Sigmoid and softmax functions are preferred for a CNN model for binary classification, and softmax is typically employed for multi-class classification. To put it simply, activation functions in a CNN model decide whether or not to activate a neuron. It determines through mathematical processes if the input to the work is significant or not.

Sigmoid function

It is a function which is plotted as ‘S’ shaped graph.

Equation : $A = 1/(1 + e^{-x})$

Nature : Non-linear. Notice that X values lies between -2 to 2, Y values are very steep. This means, small changes in x would also bring about large changes in the value of Y.

Value Range : 0 to 1

Uses : Usually used in output layer of a binary classification, where result is either 0 or 1, as value for sigmoid function lies between 0 and 1 only so, result can be predicted easily to be 1 if value is greater than 0.5 and 0 otherwise

Tanh function

The activation that works almost always better than sigmoid function is Tanh function also known as Tangent Hyperbolic function. It’s actually mathematically shifted version of the sigmoid function. Both are similar and can be derived from each other.

Equation :-

$$f(x) = tanh(x) = \frac{2}{1+e^{-2x}} - 1$$

Value Range :- -1 to +1

Nature :- non-linear

Uses :- Usually used in hidden layers of a neural network as it's values lies between -1 to 1 hence the mean for the hidden layer comes out be 0 or very close to it, hence helps in centering the data by bringing mean close to 0. This makes learning for the next layer much easier

Relu function

It Stands for Rectified linear unit. It is the most widely used activation function. Chiefly implemented in hidden layers of Neural network.

Equation :- A(x) = max(0,x). It gives an output x if x is positive and 0 otherwise.

Value Range :- [0, inf)

Nature :- non-linear, which means we can easily backpropagate the errors and have multiple layers of neurons being activated by the ReLU function.

Uses :- ReLu is less computationally expensive than tanh and sigmoid because it involves simpler mathematical operations. At a time only a few neurons are activated making the network sparse making it efficient and easy for computation.In simple words, RELU learns much faster than sigmoid and Tanh function

Softmax function

The softmax function is also a type of sigmoid function but is handy when we are trying to handle multi- class classification problems.

Nature :- non-linear

Uses :- Usually used when trying to handle multiple classes. the softmax function was commonly found in the output layer of image classification problems.Thesoftmax function would squeeze the outputs for each class between 0 and 1 and would also divide by the sum of the outputs.

Output:- The softmax function is ideally used in the output layer of the classifier where we are actually trying to attain the probabilities to define the class of each input.

The basic rule of thumb is if you really don't know what activation function to use, then simply use RELU as it is a general activation function in hidden layers and is used in most cases these days. If your output is for binary classification then, sigmoid function is very natural choice for output layer. If your output is for multi-class classification then, Softmax is very useful to predict the probabilities of each classes.

III.3.6 CNN Models

Researchers are creating new architectures and algorithms to solve image categorization challenges as machine learning for images continues to grow. CNNs come in a variety of architectures, and these are essential when creating algorithms. We will examine some of the most significant computer vision Olympics in this chapter. There are over 15 million photos in the ImageNet's database, which has more than 22,000 categories. The top-1 error rate and top-5 error rate are used as benchmarks for evaluating the models. Top-1 error rate is the percentage of incorrectly classified test images or inputs, whereas top-5 error rate is the percentage of incorrectly categorized test inputs that were not placed in one of the top-5 most likely classes. LeNet, Alex Net, and other convolutional neural network designs are some example..

III.3.6.1 LeNet

In 1998, the LeNet-5 architecture was introduced in a research paper titled "Gradient-Based Learning Applied to Document Recognition" by Yann LeCun, and all [32]. It is one of the earliest and most basic CNN architecture.

.

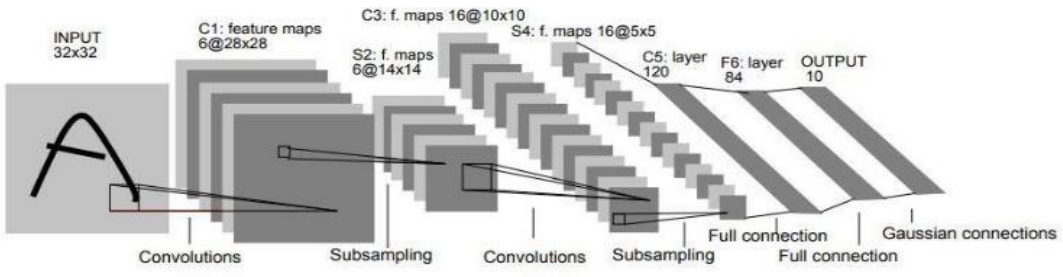

Figure27 : LENET architecture**[32]**

It consists of 7 layers. The first layer consists of an input image with dimensions of 32×32. It is convolved with 6 filters of size 5×5 resulting in dimension of 28x28x6. The second layer is a Pooling operation which filter size 2×2 and stride of 2. Hence the resulting image dimension will be 14x14x6. Similarly, the third layer also involves in a convolution operation with 16 filters of size 5×5 followed by a fourth pooling layer with similar filter size of 2×2 and stride of 2. Thus, the resulting image dimension will be reduced to 5x5x16Once the image dimension is reduced, the fifth layer is a fully connected convolutional layer with 120 filters each of size 5×5. In this layer, each of the 120 units in this layer will be connected to the 400 (5x5x16) units from the previous layers. The sixth layer is also a fully connected layer with 84 units The final seventh layer will be a softmax output layer with 'n' possible classes depending upon the number of classes in the dataset.

III.3.6.2 AlexNet

AlexNet is a convolutional neural network architecture that was introduced by Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton in 2012. This deep learning model won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2012, which is a benchmark competition in image classification , AlexNet was one of the first deep convolutional neural networks that was designed to handle large-scale image classification tasks. It consists of eight layers, including five convolutional layers, two fully connected layers, and a softmax classifier at the output layer, One of the key features of AlexNet is the use of Rectified Linear Units (ReLU) as activation functions in the hidden layers. ReLU is a non-linear activation function that is faster and more efficient than other activation functions, such as sigmoid and hyperbolic tangent. Another important aspect of AlexNet is the use of dropout regularization, which helps to prevent overfitting by randomly dropping out units in the fully connected layers during training, AlexNet was trained on the ImageNet dataset, which consists of 1.2 million images and 1000 classes.

The network was trained using a stochastic gradient descent (SGD) optimizer with a learning rate of 0.01, a momentum of 0.9, and weight decay of 0.0005. The network was trained on two NVIDIA GTX 580 GPUs for five to six days The

performance of AlexNet on the ImageNet dataset was groundbreaking at the time. It achieved a top-5 error rate of 15.3% and a top-1 error rate of 37.5%, which was a significant improvement over the previous state-of-the-art results. The success of AlexNet sparked a revolution in deep learning and led to the development of many other deep convolutional neural network architectures for image classification and other computer vision tasks, AlexNet is a pioneering deep learning model that introduced several important concepts, such as the use of ReLU activation functions and dropout regularization, which have become standard in modern deep learning architectures. Its success on the ImageNet dataset paved the way for many other deep learning models and applications in computer vision and beyond.

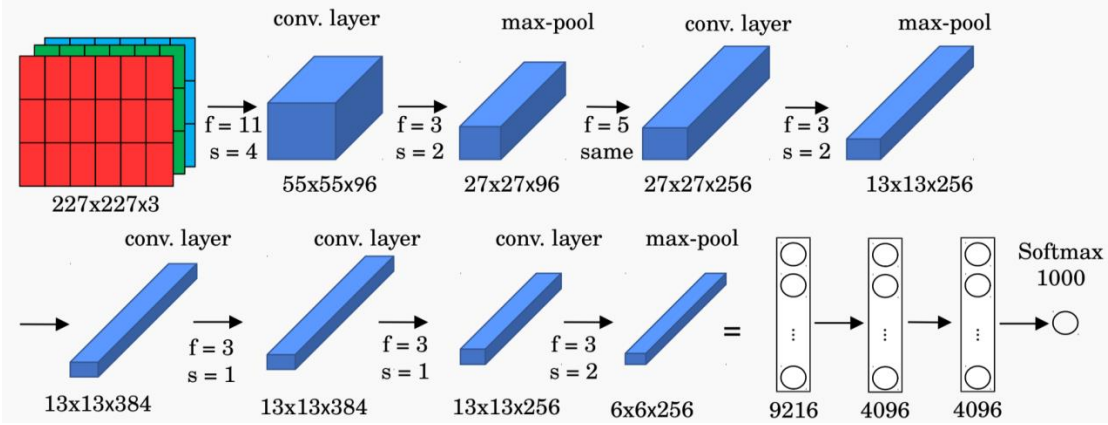

Figure 28 : ALEXNET architecture[33**]**

AlexNet is a deep convolutional neural network that consists of eight layers, including five convolutional layers, three fully connected layers, and a softmax classifier at the output layer. The architecture of AlexNet is as follows:

Input layer: The input to the network is a 224 x 224 RGB image.

Convolutional layers: The first layer of AlexNet is a convolutional layer that consists of 96 filters with a size of 11 x 11 and a stride of 4. The second and the third convolutional layers have 256 and 384 filters with a size of 5 x 5, respectively. The fourth and the fifth convolutional layers have 384 and 256 filters with a size of 3 x 3, respectively

Max pooling layers: After each convolutional layer, there is a max pooling layer with a size of 3 x 3 and a stride of 2.

Fully connected layers: The output of the last convolutional layer is flattened and fed into two fully connected layers with 4096 neurons each. The last fully connected layer has 1000 neurons, corresponding to the number of classes in the ImageNet dataset.

Softmax classifier: The output of the last fully connected layer is fed into a softmax classifier to produce the final probability distribution over the classes

How does it works ?

AlexNet is primarily used for image classification tasks. It works by learning features of images in a hierarchical manner. The input image is first passed through a series of convolutional layers, where each layer learns different features of the image. These features become more complex and abstract as the image moves through the layers. The max pooling layers reduce the spatial dimension of the feature maps, making the network more efficient and reducing overfitting.

The fully connected layers at the end of the network combine the learned features from the convolutional layers and produce a final probability distribution over the classes in the ImageNet dataset. The softmax classifier ensures that the probabilities sum up to 1 and produces the final output of the network.AlexNet uses ReLU activation functions in the hidden layers, which help to speed up the training process and prevent the vanishing gradient problem. Dropout regularization is also used in the fully connected layers to prevent overfitting and improve the generalization performance of the network

.

III.3.6.3 GoogleNet

The ILSVRC 2014 competition was won by Google's GoogleNet (a.k.a. Inception V1). It had a top-five error rate of 6.67%! This was quite near to human level performance, which the challenge organizers were now obligated to analyze As it turns out, beating GoogleNets accuracy was actually quite difficult and required some human training. The human expert (Andrej Karpathy) was able to attain a top-5 error rate of 5.1% (single model) and 3.6% (ensemble) after only a few days of training. The

network employed a CNN inspired by LeNet but included a new element known as an inception module. Batch normalization, picture distortions, and RMSprop were all employed. This module relies on a series of extremely small convolutions to dramatically minimize the amount of parameters. Their architecture included a 22-layer deep CNN, however the number of parameters was lowered from 60 million (AlexNet) to 4 million.

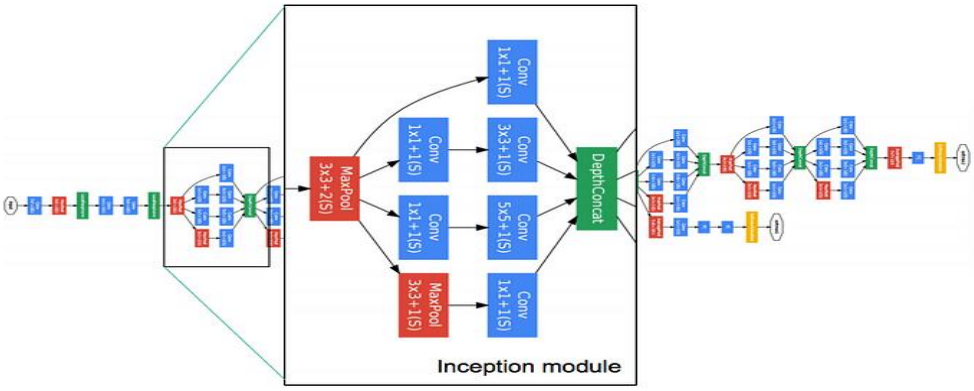

Figure 29 : GoogleNet Inception Module**[34]**

The alternate view of this architecture in a tabular format in below

type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

Table 1 : GoogleNet Architecture Tabular View.

III.3.6.4 VGG-16

. VGG-16 is a convolutional neural network (CNN) architecture proposed by Karen Simonyan and Andrew Zisserman [34] from the University of Oxford in 2014. It is a widely-used deep learning model for image classification and other computer vision tasks .

III.3.6.4.1 VGG16 Architecture:

A picture with the dimensions (224, 224, 3) is used as the network's input. The same padding and 64 channels with a 3*3 filter size are present in the first two layers. Then, two layers have convolution layers of 128 filter size and filter size (3, 3), followed by a max pool layer of stride (2, 2). The next layer is a max-pooling stride (2, 2) layer that is identical to the layer before it. There are then 256 filters spread across 2 convolution layers with filter sizes of 3 and 3. There are then two sets of three convolution layers, followed by a max pool layer. Each filter has the same padding and has 512 filters of size (3, 3). The stack of two then receives this image.

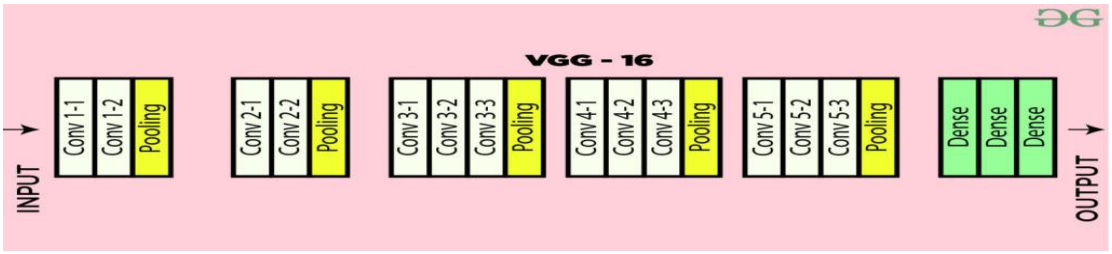

Figure 30 : VGG-16 Map [34]

We obtained a (7, 7, 512) feature map after adding a convolution and max-pooling layer to the stack. This output is flattened to create a (1, 25088) feature vector. There are then 3 fully connected layers; the first layer uses the most recent feature vector as input and produces a vector of size (1, 4096); the second layer also produces a vector of size (1, 4096); however, the third layer produces a vector of size (1, 1000), which is used to implement the softmax function to classify 1000 classes. ReLU is used by every hidden layer as its activation function. Because ReLU promotes quicker learning and lessens the likelihood of vanishing gradient issues, it is more computationally efficient ,One of the main contributions of VGG-16 is its use of very small 3x3 convolutional filters, which allows for a deeper architecture while keeping the number of parameters relatively low. This makes the model easier to train and less prone to overfitting , VGG-16 has achieved excellent performance on the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) dataset, surpassing the previous state-of-the-art performance by a large margin. Its architecture has also been used as a base model for many other computer vision tasks, such as object detection and semantic segmentation .

Overall, VGG-16 is a simple yet powerful deep learning architecture that has contributed greatly to the advancement of computer vision research.

III.3.6.5 VGG-19

The Visual Geometry Group Network (VGGNet) is a process multilayer deep neural network. VGGNet is based on the CNN model and is applied to the ImageNet dataset. VGG-19 is useful because of its simplicity as 3 3 convolution layers they are installed on top to expand with depth level. To reduce the volume size, the maximum grouping layers were used as a parameter in VGG-19. Used as input data for VGGNet. In the training phase, convolutional layers are used Extraction of features and maximum pooling layers associated with certain convolutional layers Reduce the dimensions of the features. In the first convolutional layer, there were 64 cores (3*3 filter size). Apply to extract the entity from the captured images. Fully connected layers were used to prepare feature vectors. The acquired feature vector is additionally exposed for PCA and SVD to the dimensions Reduce and select image data function for better classification results. Reduce high [31]. Dimensional data using PCA and SVD is an important task. PCA and SVD are more beneficial because it is faster and numerically more stable than other reduction techniques. Technical. The performance of the VGG-19 based system was compared against other feature mining architectures including AlexNet and SIFT. AlextNet is a multi-layered retrieval architecture used in CNN. Scaled Fixed Feature Transformation (SIFT) is a classical feature extraction [31]. A technique introduced by Mansour to detect local features of the input image in a computer vision field.

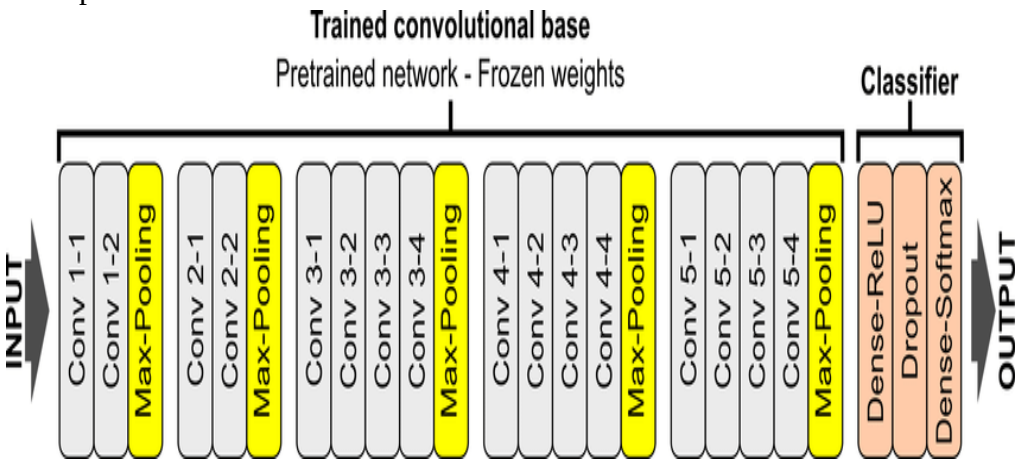

Figure 31 : layers in the vgg19 architecture[36]

III.3.6.6 ResNet

Deep residual learning for image identification developed by Microsoft researchers was incorporated into the ResNet model, which won the ISRVC2015[38] with a top-5 error rate of 3.57%, which is even lower than the standard for humans (5-10%). In comparison to its forerunners, the AlexNet and VGG nets, the ResNet is substantially deeper. One of the models created using residual networks has 152 layers, which is nearly eight times as many as VGG nets Although the accuracy of the CNN

model tends to increase with depth training deep networks would encounter the vanishing/exploding gradient problem [39, 40], which prevents convergence But normalized initialization handles this issue and allows deep networks to converge for stochastic gradient descent (SGD) with backpropagation. The accuracy of the training set saturates and consistently worsens with increasing depth, as seen in figure 32-33, even while deeper networks are able to converge a degradation This is not the result of greater depth or overfitting. This means that not all models behave in optimization in the same way.

Residual Network: This architecture introduced the idea of Residual Blocks to address the vanishing/exploding gradient issue. We apply a method known as skip connections in this network. The skip connection bypasses some levels in between to link layer activations to subsequent layers. This creates a leftover block. These leftover blocks are stacked to create resnets .

The strategy behind this network is to let the network fit the residual mapping rather than have layers learn the underlying mapping. Thus, let the network fit instead of using the initial mapping H(x) .

F(x) = H(x) - x which gives H(x) = F(x) + x

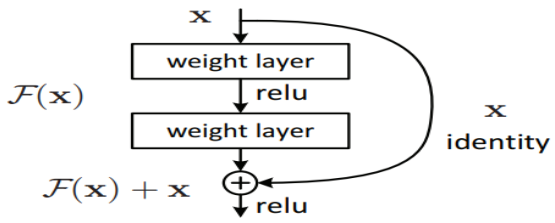

Figure 32 :Skip (Shortcut) connection**[12]**

The benefit of including this kind of skip link is that regularization will skip any layer that degrades architecture performance. As a result, training an extremely deep neural network is possible without encountering issues with vanishing or expanding gradients. The CIFAR-10 dataset's 100–1000 layers were used for experimentation by the paper's authors. The term "highway networks" refers to a similar method that also uses skip connections. These skip connections also make use of parametric gates, just as LSTM. The amount of data that flows across the skip connection is controlled by

these gates. However, this architecture has not offered accuracy that is superior to ResNet architecture. **Network Architecture:** This network uses a 34-layer plain network architecture inspired by VGG-19 in which then the shortcut connection is added. These shortcut connections then convert the architecture into a residual network.

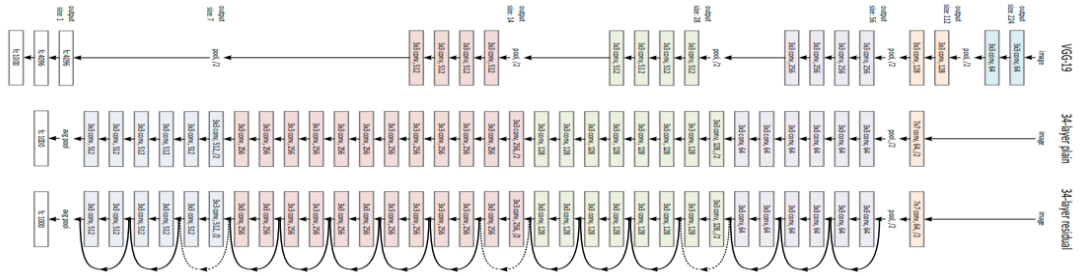

Figure 33: ResNet architecture **[12]**

III.3.6.7 DenseNet

DenseNet (Densely Connected Convolutional Networks) is a deep neural network architecture that was introduced by Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Weinberger in their 2016 paper titled "Densely Connected Convolutional Networks". It has gained popularity due to its ability to mitigate the vanishing gradient problem, promote feature reuse, and improve the accuracy of image classification tasks.

DenseNet architecture

The DenseNet architecture is based on the concept of dense connections. In traditional convolutional neural networks, the output of a layer is fed as input to the next layer. In DenseNet, every layer is connected to every other layer in a feedforward fashion. This means that the output of each layer is fed as input to all subsequent layers. This creates dense connections between layers, which allows for the reuse of features learned by earlier layers in the network.

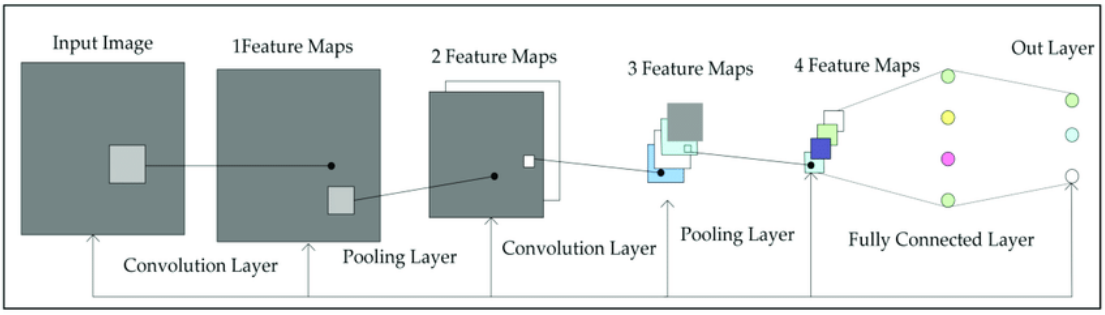

Figure 34 : DenseNet Architecture[37]

The core building block of DenseNet is the dense block. A dense block is composed of a series of convolutional layers followed by a concatenation operation that merges the output of all the previous layers. The concatenation operation produces a feature map that is then passed as input to the next dense block. By connecting each layer to every subsequent layer, the number of feature maps increases in a geometric progression, resulting in a high-dimensional feature space. This allows DenseNet to learn more discriminative features while minimizing the number of parameters in the network In addition to dense blocks, DenseNet also includes transition layers, which reduce the spatial dimensions of the feature maps. A transition layer consists of a batch normalization layer, a 1x1 convolutional layer, and a 2x2 average pooling layer. The batch normalization layer normalizes the output of the previous layer, the 1x1 convolutional layer reduces the number of feature maps, and the average pooling layer reduces the spatial dimensions of the feature maps.

CNN has great advantage in the field of image processing and recognition . it also has several popular models , among them will chose LeNet, AlexNet,GoogleNet and VGG-16 to test them in our dataset of electronic components .we will also compare them and chose the best one in terms of performance.

Chapter IV

RESULTS AND DISCUSSION

IV.INTRODUCTION

Following feature engineering, dataset selection, and model implementation that produced an output as a probability or a class, the next stage was to assess the performance of the metric-based model using ensemble test data. Evaluation of deep learning models is a crucial component of a project.

IV.1 Dataset preprocessing

The dataset consists of 1378 images belonging to four distinct classes: transistor, capacitor, LED, and relay in some cases, the dataset may be insufficient to achieve better results in deep CNN networks To enhance the dataset's diversity and size.as shown in figure 35.

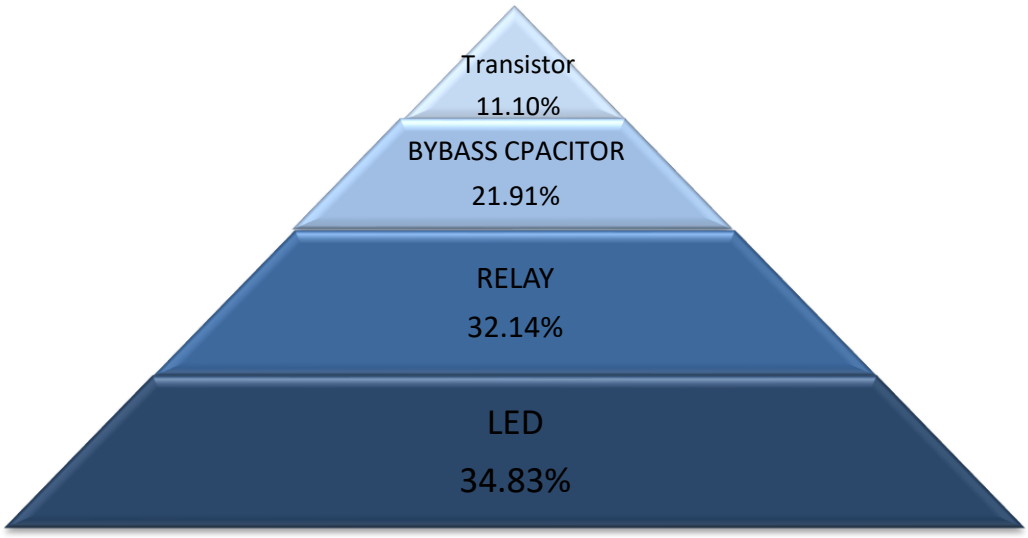

Figure 35 : Distribution of dataset class data

Before the training we split our dataset into 70% for training and 15% for testing and 15% for validation then we apply a series of data augmentations such horizon flip gaussian blur …etc , The reason we do this is very simple. If we don't split the data into different sets, the models would be evaluated on the same data it saw during the training

Figure 36 : Bypass Capacitor

Figure 37 : LED

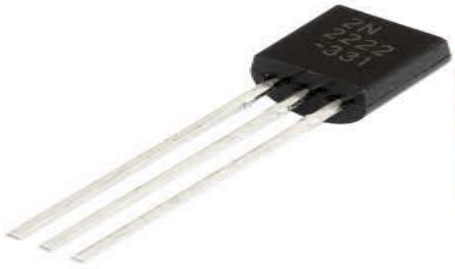

Figure 38 : Relay

Figure 39 :Transistor

The performance metrics used to evaluate the knowledge and outcomes attained as a result of the network experiments covered in the previous chapter are presented in this Chapter.

2 Performance indicators

As vital as it is to prepare data and train a machine learning model, it is also crucial to evaluate the effectiveness of the model after it has been trained. It is crucial to understand how well the model generalizes the hidden data and whether it can be used to address the issue , Deep learning activities are related to evaluation measures. Different metrics are available for various jobs. The focus will be on measures that can assess a classification task because this experiment is based on classification.

IV.3 Confusion matrix

Machine learning entails giving an algorithm with data so that it can figure out how to carry out a particular task on its own. In classification issues, it makes predictions that must be compared to actual outcomes to see how well it performed. The confusion matrix, also known as the contingency table, is typically used. It will not only show which forecasts were right and wrong, but more importantly, it will show what kinds of mistakes were made. One needs a set of test data and another set of

validation data that contains the values of the findings acquired in order to calculate a confusion matrix. confusion matrix An N x N matrix, where N is the number of expected classes, is a confusion matrix. There are three classes and a 3x3 grid in the current issue. The values shown in the confusion matrix are as follows The rows of the table's columns correspond to the actual classes, and each column contains a class predicted by the algorithm.

True Positive (TP) - A result where the model correctly predicts the positive class.

True Negative (TN) - A result where the model correctly predicts the negative class.

False Positive (FP) - A result where the model incorrectly predicts the positive class.

False Negative(FN) -A result where the model incorrectly predicts the negative Class

Table 2: confusion Matrix

confusion matrix		True/Reel			
		True	False		
Model Predictions	True	TP (True positive)	FP (False positive)	Precision =	TP/ (TP+FN)
	False	FN (False negative)	TN (Truenegative)	Negative Predictive Value	TP/ (FN+TN)
		Sensitivity/recall = TP/ (TP+FN)	Specificity =TN/ (FP+TN)	Accuracy= (TP+TN)/(TP+FP+TN+FN)	

To evaluate a model's performance, the confusion matrix can be used to derive the following values :

Accuracy: The proportion of the total number of predictions that were correct.

Precision: The proportion of positive cases that were correctly identified.

Sensitivity or Recall: The proportion of actual positive cases that are correctly identified.

Specificity: The proportion of actual negative cases that are correctly identified.

F1-score: precision and recall weighted mean or harmonic mean. This score considers both false positives and false negatives. The F1 score is a number between 0 and 1.

IV..4 Multi-class classification metrics

IV.4.1 Marco accuracy

the ratio between the amount of correct predictions. It averages over the number of labels in the classification. The numerator finds how many labels in the predicted vector have in common with the ground truth, and the ratio calculates how many of the predicted true labels are actually in the ground truth.

Accuracy=(TP+TN)/(TP+FP+TN+FN)

IV.4.2 Macro recall

the ratio of predicted labels to actual labels. The numerator determines how many labels in the predicted vector share similarities with the ground truth, then calculates the ratio to the number of real labels, yielding the proportion of actual labels predicted.

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

IV.4.3 Macro F1-Score

the most commonly used F-measure variation in multi-class settings due to its equal emphasis on rare classes. It applies the F-measure to multi-class parameters by averaging precision and recall levels across all classes It depicts the delicate balance of precision and recall. In simple terms, we might state that it is the harmonic mean of the two.

$$F1\ score = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}} = \frac{2 * (Precision * Recall)}{(Precision + Recall)}$$

$$F1 = 2.\frac{PRE.REC}{PRE + REC}$$

IV.5 Experiments and results

All experimentation in this thesis is conducted using the Windows platform. the machine uses an Intel Core i5- 8400H 2.5 GHz CPU, an NVIDIA GTX1050 (2G) GPU and 12 GB RAM . In addition to selecting several deep learning networks such as LENET, ALEXNET, and GOOGLE NET ,VGG16 This experiment performed a preliminary examination of the networks' training performance to identify the final network to be processed for the detection challenge. To examine network efficiency, performance was measured against a micro set of 1378 images of Electronic components divided into 4 classes (153 transistor ,443 relay,480 Led ,302 bypass-capacitor).

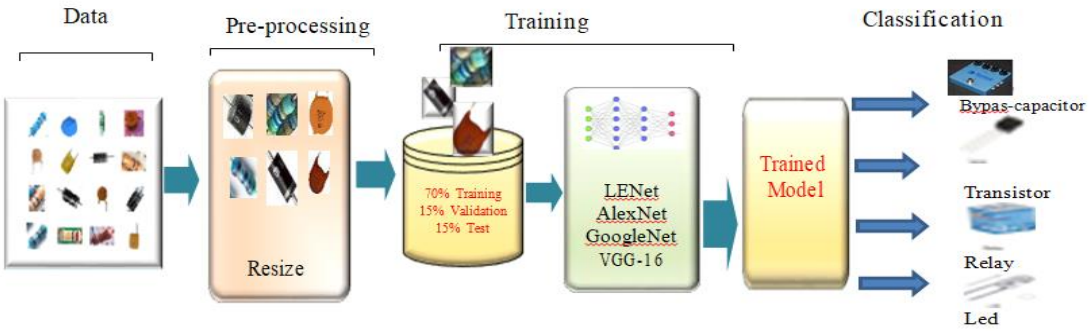

Figure 40: Flowchart of our Models training process

Ⅳ.5.1 Comparison of performance between models

Ⅴ.5.1.1 LeNet performance

LeNet was trained for 100 epochs from scratch before setting the other half of the layers. By unlocking these layers, the associated weights were learned and changed based on the specific dataset's properties. The architecture acquired after the layer change process is shown in the table below.

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 28, 28, 6)	456
max_pooling2d (MaxPooling2D)	(None, 14, 14, 6)	0
conv2d_1 (Conv2D)	(None, 10, 10, 16)	2416
max_pooling2d_1 (MaxPooling2D)	(None, 5, 5, 16)	0
flatten (Flatten)	(None, 400)	0
dense (Dense)	(None, 120)	48120
dense_1 (Dense)	(None, 84)	10164
dense_2 (Dense)	(None, 4)	340

Total params: 61,496
Trainable params: 61,496
Non-trainable params: 0

Table 3: Layers in the new model architecture

The following training settings were also chosen categorical_ cross entropy loss, 'Adam' optimizer, 100 epochs, 32 batch size The following curves were obtained after 1 hours 15 min of training on the above parameters.

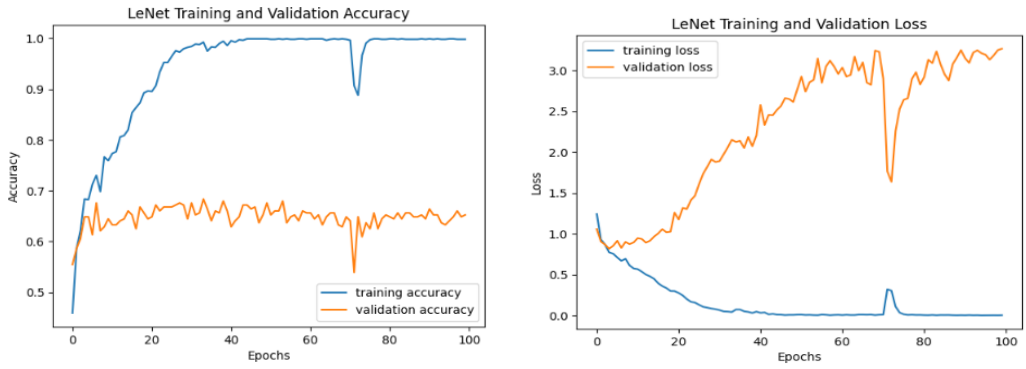

Figure 41: Training graphs of LeNet

The network reached its maximum training accuracy of 99% at the end of 50 epochs with a commit accuracy of 67.58%. The training time for LeNet was estimated at 1 hours 15 minutes. The trained model was evaluated against the same mini-test set and achieved an accuracy of 67 % and a Macro f1 score of 0.66 and recall of 0.66.

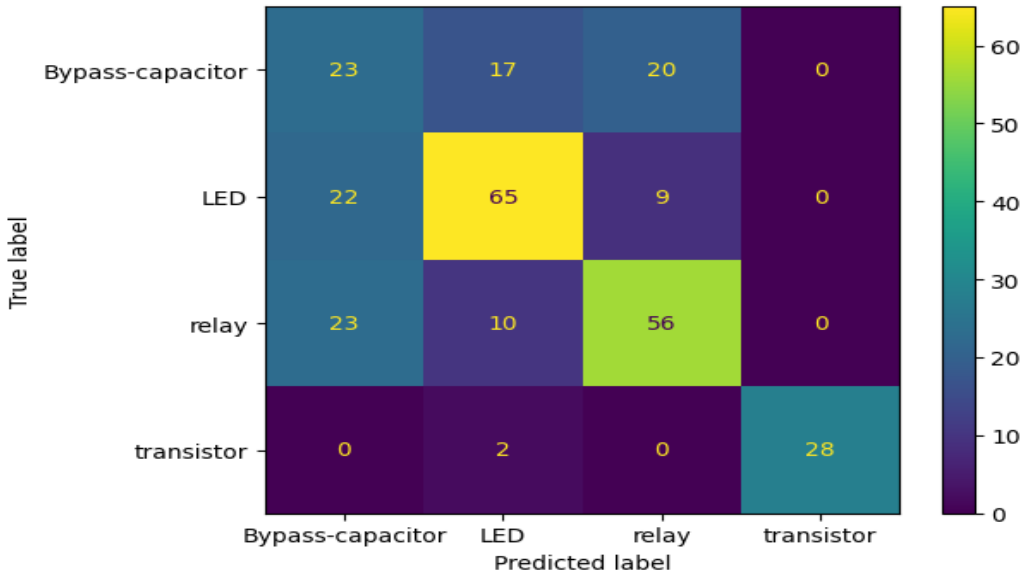

Figure 42 : Confusion matrix of LeNet on mini-testes

IV.5.1.2 AlexNet performance

AlexNet was trained for 100 epochs from scratch before setting the other half of the layers. By unlocking these layers, the associated weights were learned and changed based on the specific dataset's properties. The architecture acquired after the layer change process is shown in the table below.

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 227, 227, 3)]	0
conv2d (Conv2D)	(None, 55, 55, 96)	34944
max_pooling2d (MaxPooling2D)	(None, 27, 27, 96)	0
conv2d_1 (Conv2D)	(None, 27, 27, 256)	614656
max_pooling2d_1 (MaxPooling2D)	(None, 13, 13, 256)	0
conv2d_2 (Conv2D)	(None, 13, 13, 384)	885120
conv2d_3 (Conv2D)	(None, 13, 13, 384)	1327488
conv2d_4 (Conv2D)	(None, 13, 13, 256)	884992
max_pooling2d_2 (MaxPooling2D)	(None, 6, 6, 256)	0
flatten (Flatten)	(None, 9216)	0
dense (Dense)	(None, 4096)	37752832
dense_1 (Dense)	(None, 4096)	16781312
dense_2 (Dense)	(None, 4)	16388

Total params: 58,297,732
Trainable params: 58,297,732
Non-trainable params: 0

Table 4: Layers in the new model architecture

The following training settings were also chosen categorical_ cross entropy loss, 'Adam' optimizer, 100 epochs, 32 batch size The following curves were obtained after 2 hours 50 min of training on the above parameters.

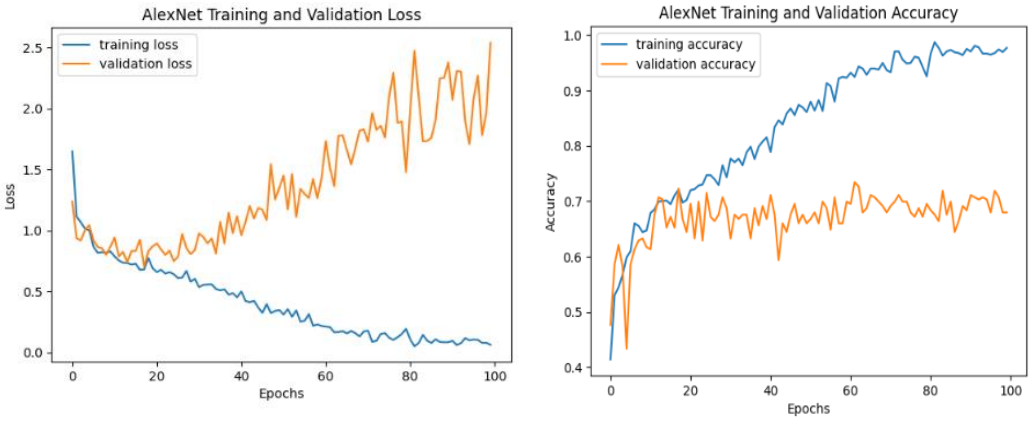

Figure 43: Training graphs of AlexNet

The network reached its maximum training accuracy of 98.86% at the end of 82 epochs with a commit accuracy of 67.58%. The training time for AlexNet was estimated at 2 hours 50 minutes. The trained model was evaluated against the same mini-test set and achieved an accuracy of 74% and a Macro f1 score of 0.74 and recall of 0.73.

The following image represents the confusion matrix during the test phase

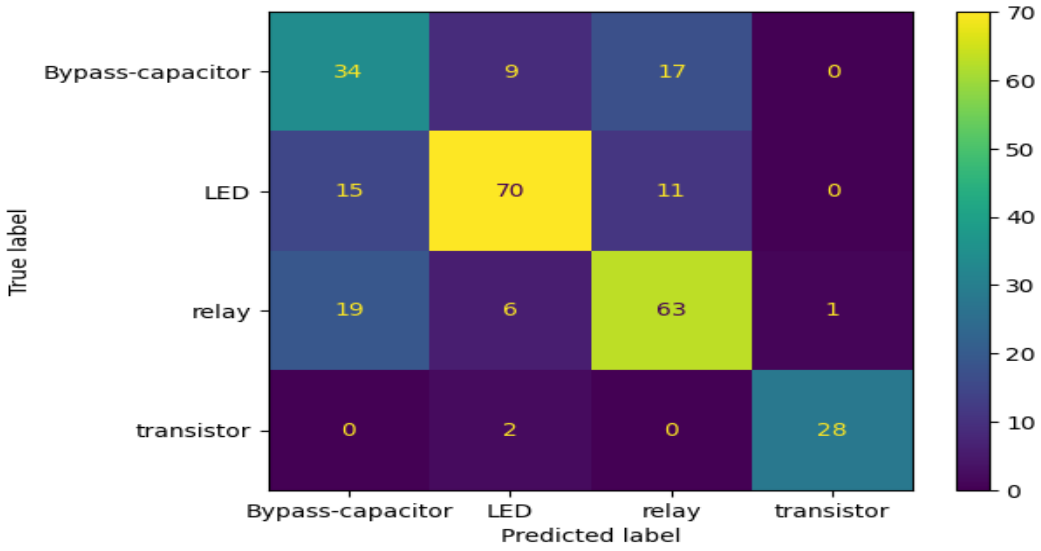

Figure 44: Confusion matrix of AlexNet on mini-testes

IV.5.1.3 GoogleNet performance

GoogleNet was trained for 100 epochs from scratch before setting the other half of the layers. By unlocking these layers, the associated weights were learned and changed based on the specific dataset's properties. The architecture acquired after the layer change process is shown in the table below.

concatenate_7 (Concatenate)	(None, 6, 6, 832)	0	['conv2d_45[0][0]', 'conv2d_47[0][0]', 'conv2d_49[0][0]', 'conv2d_50[0][0]']
conv2d_52 (Conv2D)	(None, 6, 6, 192)	159936	['concatenate_7[0][0]']
conv2d_54 (Conv2D)	(None, 6, 6, 48)	39984	['concatenate_7[0][0]']
max_pooling2d_12 (MaxPooling2D)	(None, 6, 6, 832)	0	['concatenate_7[0][0]']
conv2d_51 (Conv2D)	(None, 6, 6, 384)	319872	['concatenate_7[0][0]']
conv2d_53 (Conv2D)	(None, 6, 6, 384)	663936	['conv2d_52[0][0]']
conv2d_55 (Conv2D)	(None, 6, 6, 128)	153728	['conv2d_54[0][0]']
conv2d_56 (Conv2D)	(None, 6, 6, 128)	106624	['max_pooling2d_12[0][0]']
concatenate_8 (Concatenate)	(None, 6, 6, 1024)	0	['conv2d_51[0][0]', 'conv2d_53[0][0]', 'conv2d_55[0][0]', 'conv2d_56[0][0]']
global_average_pooling2d (GlobalAveragePooling2D)	(None, 1024)	0	['concatenate_8[0][0]']
dense (Dense)	(None, 4)	4100	['global_average_pooling2d[0][0]']

Total params: 5,978,676
Trainable params: 5,978,164
Non-trainable params: 512

Table 5: Layers in the new model architecture

The following training settings were also chosen

'categorical_ cross entropy loss, 'Adam' optimizer, 100 epochs, 32 batch size

The following curves were obtained after 3 hours 40 min of training on the above

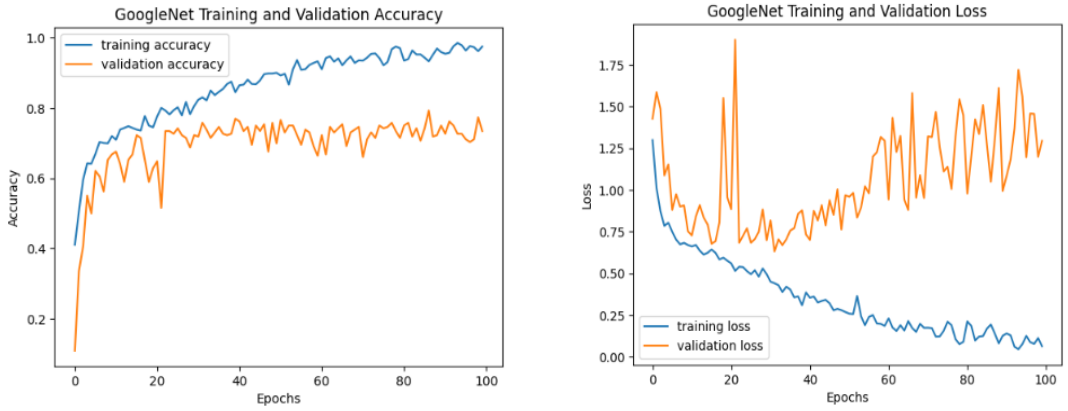

Figure 45: Training graphs of GoogleNet

The network reached its maximum training accuracy of 97.57% at the end of 94 epochs with a commit accuracy of 75 %. The training time for GoogleNet was estimated at 3 hours 40 minutes. The trained model was evaluated against the same mini-test set and achieved an accuracy of 71% and a Macro f1 score of 0.68 and recall of 0.66.

The following image represents the confusion matrix during the test phase

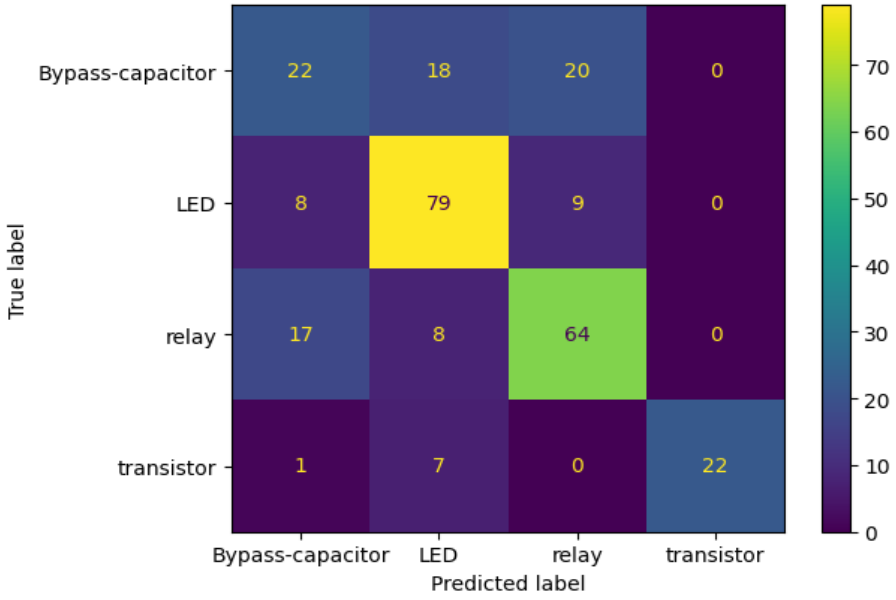

Figure 46: Confusion matrix of GoogleNet on mini-testes

IV.5.1.4 VGG-16 performance

VGG-16 was trained for 25 epochs in transfer learning before setting the other half of the layers. By unlocking these layers, the associated weights were learned and changed based on the specific dataset's properties. The architecture acquired after the layer change process is shown in the table below.

block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv3 (Conv2D)	(None, 14, 14, 512)	2359808
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0
global_average_pooling2d (GlobalAveragePooling2D)	(None, 512)	0
dense (Dense)	(None, 256)	131328
dense_1 (Dense)	(None, 4)	1028

Total params: 14,847,044
Trainable params: 132,356
Non-trainable params: 14,714,688

Table 6: Layers in the new model architecture

The following training settings were also chosen categorical_ cross entropy loss, 'Adam' optimizer, 25 epochs, 32 batch size The following curves were obtained after 1 hours of training on the above parameters.

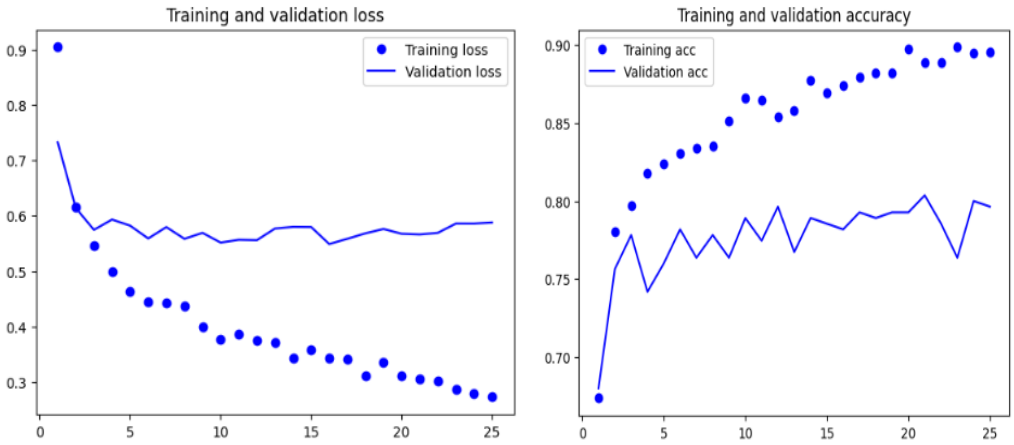

Figure 47: graphs of pre-trained VGG-16

- The network reached its maximum training accuracy of 89.48% at the end of 24 epochs with a commit accuracy of 80%. The training time for VGG16 was estimated at 1 hours. The trained model was evaluated against the same mini-test set and achieved an accuracy of 81% and a Macro f1 score of 0.81 and recall of 0.81.

The following image represents the confusion matrix during the test phase

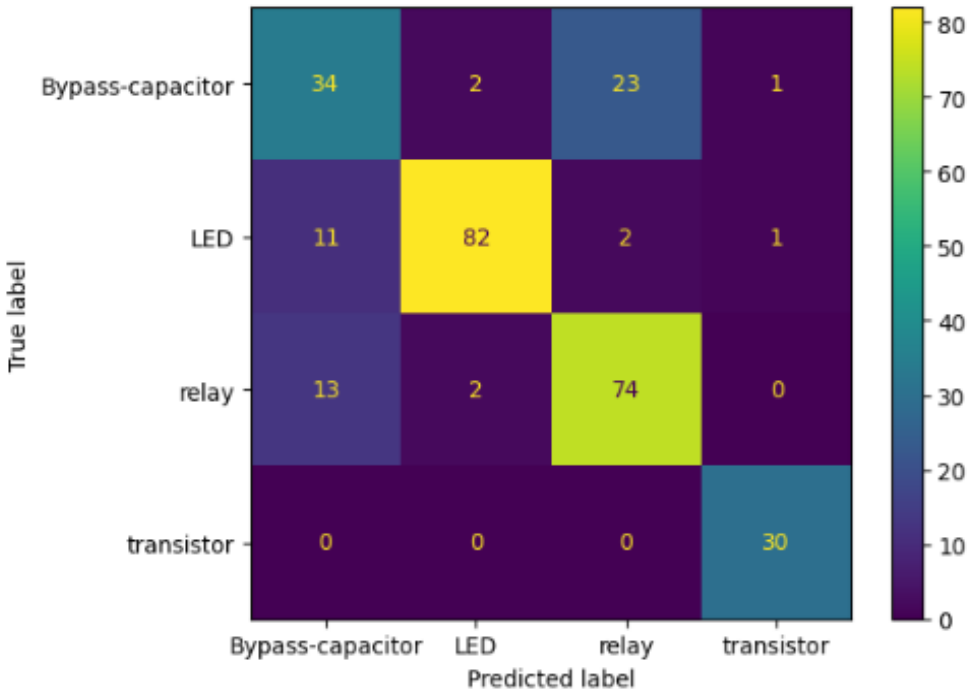

Figure 48: Confusion matrix of VGG-16 on mini-testes

Ⅳ.5.2 Results Comparison

The results obtained for Macro-Accuracy, Macro Recall and Macro F1-Score are summarized for all networks so that we can compare with a more accurate picture. The results are listed in the following table.

Table 7: Precision, recall and F1-Scores of different networks on mini-testset

	Macro-precision	Macro Recall	Macro F1-Score
LENET	67	66	66
ALEXNET	74	73	74
GOOGLENET	71	66	68
VGG16	81	81	81

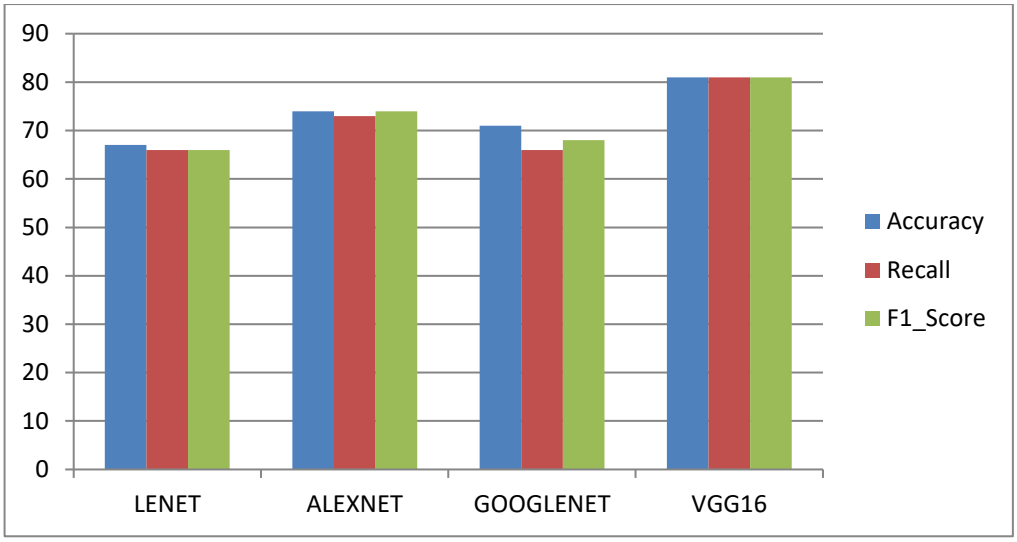

Figure 49.Performance indicators obtained with CNN model

By comparing the results as shown in figure 49 it is safe to say that the pre-trained VGG16 model got the highest perfomance compared the other models such as LeNet, AlexNet and GoogleNet In terms of accuracy by 14% 7% and 10% respectively and in terms of recall it got 15% , 8% and 15% compraed to LeNet ,AlexNet and GoogleNet respectively and by 15% . 7% and 13% in terms of f1-score compraed to LeNet ,AlexNet and GoogleNet respectively.

As shown in figure 50 its clear that the transistor was the easiest components to classified by the models followed by the led , relay, and the bypass capacitor mostly because it's the class with the lowest number of components compared to other classes this means the accuracy of our tested models might decrease in larger amount of data however the accuracy of transistor and led are quite similar in vgg16 thanks to it being pre-trained model unlike other models whom trained from scratch.

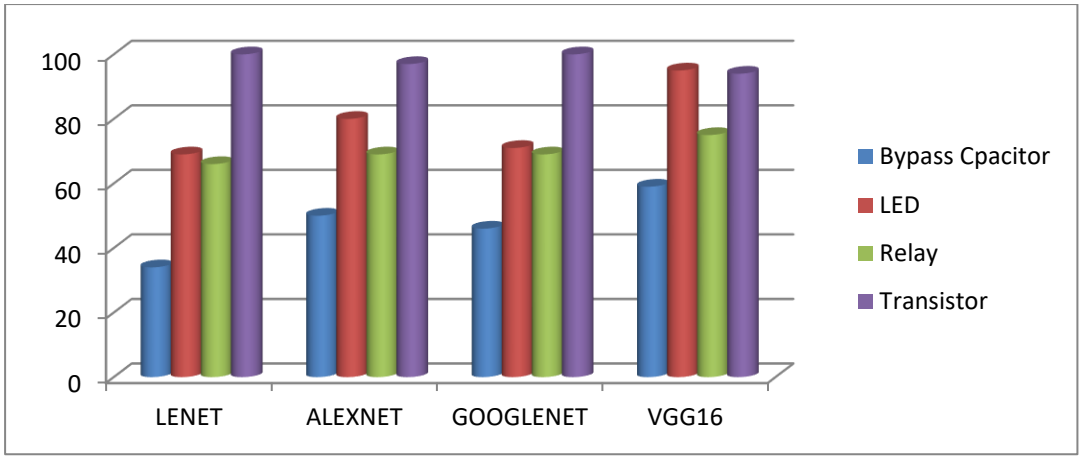

Figure 50 : Accuracy(%) for each class by CNN model

V. CONCLUSION GENERAL

In this thesis , various deep learning-based classification approaches were utilized to classify electronic components. We compared the forecasting performances of these approaches and. Instead of modifying the data, we also focused on identifying the appropriate fine-tuning strategies.

The study observed that the implementation of CNN-based architectures for classification yielded effective results, with high accuracy rates. Specifically, the following accuracy values were obtained for the classification of electronic components: AlexNet achieved 74% accuracy, GoogleNet achieved 71% accuracy, LeNet achieved 67% accuracy, and pretrained VGG-16 achieved 81% accuracy on small data set of 1378 images this means that the pre-trained VGG-16 was the best model out of all with higher performance in shorter time thanks to transfer learning .

V.1 Future studies

In future studies, we plan to explore improved data augmentation methods, such as mirroring, and utilize larger datasets in conjunction with several pretrained models. We also aim to calculate the training duration for larger datasets and make efforts to enhance the training process to achieve high performance in a shorter time.

Overall, the study highlights the effectiveness of deep learning-based approaches, particularly CNN-based architectures, in classifying electronic components. The researchers emphasize the need for further research to, incorporate larger datasets, and optimize training processes for improved performance.

V.2 References

[1] Yuanyuan Xu , 1,2 Genke Yang , 1,3 Jiliang Luo,2 and Jianan He4 " An Electronic Component Recognition Algorithm Based on Deep Learning with a Faster SqueezeNet ". october 2020

[2] S.-S. Du, Z.-D. Shan, Z.-C. Huang, H.-Q. Liu, and S.-L. Liu, "*the algorithmic of components auto-classification and system development of electronic components," Development & Innovation of Machinery & Electrical Products, vol. 6, pp. 133–135, 2008

[3] D. Lefkaditis, and G. Tsirigotis, "Morphological feature selection and neural classification for electronic components," Journal of Engineering Science and Technology Review, vol. 2, no. 1, pp. 151–156, 2009.

[4] Fast Classification in Incrementally Growing Spaces (Pattern Recognition and Image Analysis)

[5] Ravi Kumar Y.B.; Ravi Kumar C.N. "Local binary pattern: An improved LBP to extract non uniform LBP patterns with Gabor filter to increase the rate of face similarity" 02 January 2017

[6] M. Moetesum and S. W. Younus, M. A. Warsi and I. Siddiqi " Segmentation and recognition of electronic components in hand-drawn circuit diagrams " 13 April 2018

[7] IpekAtik" Classification of Electronic Components Based on Convolutional Neural Network Architecture " 23 March 2022

[8] Yuanyuan Xu , Genke Yang , Jiliang Luo, and Jianan He "An Electronic Component Recognition Algorithm Based on Deep Learning with a Faster SqueezeNet" 28 October 2020

[9] Rui Huang, Jinan Gu, Xiaohong Sun, Yongtao Hou and Saad Uddin "A Rapid Recognition Method for Electronic Components Based on the Improved YOLO-V3 Network " 25 July 2019

[10] Xiaohong Sun, Jinan Gu and Rui Huang " A modified SSD Method for Electronic recognition using deep learning algorithm" November 28, 2019

REFRENCES

[11] Tianhong Panand Mian Khuram Ahsan" Hand-drawn electronic component Components Fast Recognition" , © 2019 Published by Elsevi

[12] Irfan Aziz " Deep Learning: An Overview of Convolutional Neural Network" April 2020

[13] Jonathan Johnson "What's a Deep Neural Network? Deep Nets Explained" July 27, 2020

[13] Adrian RosebrockConvolutional Neural Networks (CNNs) and layrestybes on May 14, 2021

[14] sensetime, «https://www.sensetime.com/en».

[15] F. Vázquez, «Deep Learning made easy with Deep Cognition,» Dec 21, 2017

[16] M. K. a. K. Johnson, «Feature Engineering and Selection,» 2019-06-21

[17] A. Gondhalekar, «Data Augmentation — Is it really necessary,» Mar 24, 2020.

[18] X. C. M. N. a. W. Q. Y. Dongqing Shen, «In 2018 4th International Conference on Control,Automation and Robotics (ICCAR).,» Dongqing Shen, Xin Chen, Minh Nguyen, and Wei Qi Yan., IEEE, apr 2018.

[19] D. L. ahmedsaied, «Fire Dataset ,Fire and Smoke Dataset,» https://www.kaggle.com/datasets/phylake1337/fire-dataset ,https://www.kaggle.com/datasets/dataclusterlabs/fire-and-smoke-dataset.

[20] China Electronic Technology Standardization Institute. GB/T 17564.4-2001, IEC Standard Data Element Types and Related Classification Models for Electrical Components-Part 4: IEC Standard Data Element Types, Component Classifications, and Reference Sets of Items, China Standard Press, Beijing, China, 2001.

[21] Standardization Institute of Ministry of Electronic Industry, China Electronic Statistics Society. SJ/T 11144-1997, Electronic Product Classification and Code, vol. 7, Ministry of Electronics Industry, PRC, Beijing, China, 1999.

[22] B. Wang, "Base and current situation of data standardization for electronic components & devices," Electronic Component & Device Applications, vol. 11, pp. 30–32, 2010.

REFRENCES

[23] X. Chen, J.-d. Yu, X.-a. Chen, and Y. Zhai, “Classification of electronic components based on convolution neural network,” Wireless Communication Technology, vol. 27, no. 2, pp. 7–12, 2018

[24] Y. Zhu and W. X. Zheng, “Multiple lyapunov functions analysis approach for discrete-time-switched piecewise-affine systems under dwell-time constraints,” IEEE Transactions on Automatic Control, vol. 65, no. 5, pp. 2177–2184, 2020.

[25] J. Na, B. Jing, G. Gao, Y. Huang, and C. Zhang, “Unknown system dynamics estimator for motion control of nonlinear robotic systems,” IEEE Transactions on Industrial Electronics, vol. 67, no. 5, pp. 3850–3859, 2020.

[26] Y. Zhu and W. Zheng, “Observer-based control for cyberphysical systems with periodic DoS attacks via a cyclic switching strategy,” IEEE Transactions on Automatic Control, vol. 65, no. 8, pp. 3714–3721, 2020.

[27] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” in Proceedings of the 3rd International Conference on Learning Representations (ICLR), pp. 1068–1082, San Diego, CA, USA, May 2015.

[28] Yuan, J.; Hou, X.; Xiao, Y.; Cao, D.; Guan, W.; Nie, L. Multi-criteria active deep learning for image classification. Knowl.-Based Syst. 2019, 172, 86–94. [CrossRef]

[29] Cetinic, E.; Lipic, T.; Grgic, S. Fine-tuning Convolutional Neural Networks for fine art classification. Expert Syst. Appl. 2018, 114, 107–118. [CrossRef]

[30] dos Santos, M.M.; da S. Filho, A.G.; dos Santos, W.P. Deep convolutional extreme learning machines: Filters combination and error model validation. Neurocomputing 2019, 329, 359–369. [CrossRef]

[31] Liang, Z. Automatic Image Recognition of Rapid Malaria Emergency Diagnosis: A Deep Neural Network Approach. Master’s Thesis, York University, Toronto, ON, Canada, 2017. Available online: https://yorkspace.library.yorku.ca/xmlui/handle/10315/ 34319 (accessed on 10 January 2022)

REFRENCES

[32] Yann LeCun, Léon Bottou, YoshuaBengio, and Patrick Haffner LeNet Architecture: A Complete Guide Proceedings of the IEEE (1998)

[33] #013 B CNN AlexNet 10.11.2018

[34] Prabhu "CNN Architectures — LeNet, AlexNet, VGG, GoogLeNet and ResNet" Mar 15, 2018

[35] Joseph Nelson " How to Train a VGG-16 Image Classification Model on Your Own Dataset" MAY 25, 2020

[36] «Deep Learning Based Real-Time Body Condition Score Classification System,» KerimKürşatÇevik , Akdeniz University, November 2020IEEE Access 8:213950- 213957.

[37]https://www.bing.com/ck/a?!&&p=fff3cc3bc4603a01JmltdHM9MTY4NTQwNDgwMCZpZ3VpZD0xYTVkNDU0My0zNDA4LTYzZWYtMDhjMi01N2ZmMzU2MzYyOTYmaW5zaWQ9NTIwMg&ptn=3&hsh=3&fclid=1a5d4543-3408-63ef-08c2-57ff35636296&psq=DenseNet+Architecture&u=a1aHR0cHM6Ly9pcS5vcGVuZ2VudXMub3JnL2FyY2hpdGVjdHVyZS1vZi1kZW5zZW5ldDEyMS8&ntb=1

[38] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

[39] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. IEEE Transactions on Neural Networks, 5(2):157–166, 1994.

[40] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In AISTATS, 2010

[41] J. Wu, «AI, Machine Learning, Deep Learning Explained Simply,» Jul 1, 2019.