

# **MAJOR-PROJECT REPORT**

**on**

## **DOCUMENT IMAGE CLASSIFICATION**

**Submitted in partial fulfilment of the requirement for the award of degree of**

## **BACHELOR OF TECHNOLOGY**

**In**

## **COMPUTER SCIENCE AND ENGINEERING**

**By**

**P.SAILUSHA 15AG1A0584**

**Under the esteemed guidance of**

**Mrs K. Jaya Bharathi**

**Professor**



**Department of Computer Science and Engineering**

**ACE Engineering College**

**NBA ACCREDITED B.TECH COURSES: EEE,ECE,CSE and ME**

**Accorded NAAC 3.20 CGPA**

**Ankushapur (V), Ghatkesar (M), Medchal.Dist.-501301**

**(Affiliated to Jawaharlal Nehru Technological University Hyderabad,T.S.)**

**2019**



**ACE**  
**Engineering College**  
(with a Difference in Excellence)

**NBA ACCREDITED B.TECH COURSES: EEE, ECE, CSE and ME**

**Accorded NAAC 3.20 CGPA**

**Ankushapur(V), Ghatkesar(M), Medchal Dist.-501 301**

**(Affiliated to Jawaharlal Nehru Technological University Hyderabad,T.S.)**

**Website: [www.aceec.in](http://www.aceec.in) Email: [info@aceec.ac.in](mailto:info@aceec.ac.in)**

## **Computer Science and Engineering**

### **CERTIFICATE**

This is to certify that the major project report entitled **“DOCUMENT IMAGE CLASSIFICATION”** that is being submitted by **P.SAILUSHA(15AG1A0584)** in partial fulfilment for the award of the Degree of **Bachelor of Technology in Computer Science and Engineering** affiliated to the **Jawaharlal Nehru Technological University, Hyderabad** during the Academic year 2018-2019 is a record of bonafied work carried out by him/her under my guidance and supervision.

The results embodied in this project report have not been submitted to any other University or Institute for the award of any Degree or Diploma.

#### **Internal Guide**

**Prof K. Jaya Bharathi**

**Professor**

#### **Head of the Department**

**Prof. K. Jaya Bharathi**

**Professor and Head**

**Dept of CSE**

**EXTERNAL EXAMINER**

## ACKNOWLEDGEMENT

I would like to express my gratitude to all the people behind the screen who have helped us transform an idea into a real-time application.

I would like to express my heart-felt gratitude to our parents without whom i would not have been privileged to achieve and fulfil our dreams. A special thanks to our Secretary, **Prof. Y. V. GOPALA KRISHNA**, for having founded such an esteemed institution. I am also grateful to our principal, **Dr. B. L. RAJU** for permitting me to carry out this project.

I profoundly thank **Prof. K. JAYA BHARATHI**, Head of the Department of Computer Science and Engineering, who has been an excellent guide and also a great source of inspiration to my work.

I extremely thank **Mr. D.KRISHNA**, Project coordinator, who helped me in all the way in fulfilling of all aspects in completion of my major-project.

I am very thankful to the internal guide by name **Prof. K. JAYA BHARATHI**, who has been an excellent and given continuous support for the completion of my work.

The satisfaction and euphoria that accompany the successful completion of the task would be great, but incomplete without the mention of the people who made it possible, whose constant guidance and encouragement crown all the efforts with success. In this context, i would like to thank all the other staff members, both teaching and non-teaching, who have extended their timely help and eased my task.

**P.SAILUSHA (15AG1A0584)**

## **DECLARATION**

I hereby declare that this project entitled “DOCUMENT IMAGE CLASSIFICATION” which is being submitted by us in partial fulfilment of the requirement for the award of the degree of B.Tech to the Jawaharlal Nehru Technological University Hyderabad, India either in part or full does not constitute any part of any project submitted by me or any other person to any University/Institute.

**P.Sailusha (15AG1A0584)**

# INDEX

Sl.No.	CONTENTS	Page No.
	<b>ABSTRACT</b>	<b>i</b>
	<b>LIST OF FIGURES</b>	<b>ii</b>
<b>1</b>	<b>Introduction</b>	<b>1-2</b>
	1.1. Aim	1
	1.2. Existing System	1
	1.3. Proposed System	2
<b>2</b>	<b>Literature Survey</b>	<b>3-4</b>
	2.1. Introduction	3
	2.2. Literature Review	4
<b>3</b>	<b>Requirement Analysis</b>	<b>5</b>
	3.1. Introduction	5
	3.2. Functional Requirements	5
	3.3. Non-Functional Requirements	6
	3.4. Software Requirements	7
	3.5. Hardware Requirements	7
	3.5. Algorithms	7
	3.5.1. Convolutional neural network	8
	3.5.2. Vgg16 with transfer learning	9
<b>4</b>	<b>Technology Used</b>	<b>10-11</b>
	4.1. Machine learning	10
	4.2. Deep Learning	10
	4.3. Python	11
	4.4. Jupyter notebook	11
<b>5</b>	<b>System Design</b>	<b>12-29</b>
	5.1. Software Design	12
	5.2. System Architecture	13
	5.3. Data Flow diagram	14
	5.4. Modules and their functionality	15
	5.5. Project Methodology	20

	5.6. UML Diagrams	21
	5.6.1 UML Concepts	21
	5.6.2 Building Blocks in UML	22-24
	5.6.3 Class Diagram	25
	5.6.4. Use Case Diagram	26
	5.6.5 Sequence Diagram	27
	5.6.6. Collaboration Diagram	28
	5.6.7. Component Diagram	29
6	<b>Implementation</b>	<b>30-36</b>
	6.1. Data loading and cleaning	30
	6.2. Data pre-processing	31
	6.3 Training and Testing models	32
	6.4 Evaluating models	34
7	<b>Testing</b>	<b>37-39</b>
	7.1.Types of Testing	37
	7.2. Unit Test	39
	7.3 Integration Testing	39
	7.4. Acceptance Testing	39
8	<b>Output Screens</b>	<b>40-50</b>
	8.1 Running the application	40
	8.2 Data loading and cleaning	42
	8.3 Data pre-processing	43
	8.4 Evaluating models	44
	8.5 Prediction	46
	8.6 Data preparation for keras vgg	47
	8.7 model for keras vgg	48
	8.8 Confusion matrix	49
	8.9 Classifcaton report	50
9	<b>Conclusion</b>	<b>51</b>
	9.1 Future Enhancements	51
10	<b>References</b>	<b>52</b>



# ABSTRACT

Documents can be classified into various classes based on their text contents and their structural properties. During a manual search for a particular document from a large collection of documents, knowledge about the type or structure of the document helps reduce the time necessary for the search.

Many document types have a distinct visual style. For example, “letter” documents are typically written in a standard format, which is recognizable even at scales where the text is unreadable. Motivated by this observation, this addresses the problem of document classification and retrieval, based on the visual structure and layout of document images. This project presents a new state-of-the-art for document image classification and retrieval, using features learned by deep convolutional neural networks (CNNs).

The current work explores the capacity in the realm of document analysis, and confirms that this representation strategy is superior to a variety of popular handcrafted alternatives. Extensive experiments show that features extracted from CNNs are robust to compression, CNNs trained on non-document images transfer well to document analysis tasks.



## **LIST OF FIGURES**

<b>Fig.No.</b>	<b>FIGURE NAMES</b>	<b>Page No.</b>
3.5.1	Convolution multiplication with kernel	8
3.5.2	Vgg16 Architecture	9
5.1	System Architecture	13
5.2	Data Flow Diagram	14
5.4.1	Dataset	16
5.4.2	Histogram Equalisation	17
5.4.3	Feature extraction through layers	18
5.4.4	VGG16 Architecture for my project	19
5.6.1	Class	22
5.6.2	Use case	22
5.6.3	Node	22
5.6.4	Class Diagram	25
5.6.5	Use case Diagram	26
5.6.6	Sequence Diagram	27
5.6.7	Collaboration Diagram	28

5.6.8	Component Diagram	29
-------	-------------------	----



# **CHAPTER – 1**

## **INTRODUCTION**

### **1.1. Aim :**

The purpose of this project is to develop a series of convolutional neural network and test various parameter settings to classify digitally scanned documents into 16 distinct classes. The data comes the RVL-CDIP Dataset (Ryerson Vision Lab Complex Document Information Processing). This dataset consists of 400,000 greyscale images of 16 classes of documents.

The images are presplit into 320,000 training images, 40,000 validation images, and 40,000 testing images. All image dimensions are 1000 pixels or less, though this will be scaled down for training efficiency. The 16 document classes include letter, form, email, handwritten, advertisement, scientific report, scientific publication, specification, file folder, news article, budget, invoice, presentation, questionnaire, resume, and memo.

### **1.2. Existing System:**

During a manual search for a particular document from a large collection of documents, knowledge about the type or structure of the document helps reduce the time necessary for the search. However, Document classifiers based on text contents use optical character recognition (OCR) techniques to extract the texts in the document image and thus are susceptible to OCR errors. Its time consuming and not useful if we only require the category and not the content inside the document.

Several studies on automatic document classification have been made in the recent past. An approach to automatic generation of a decision tree for logical labelling of business letters. A 3-step approach based on certain GTree for partitioning a document image into its logical objects. INFOCLAS, an early prototype system for indexing and classifying printed business letters. A machine learning based approach was proposed for automatic discovery of knowledge, that is, the relevant features for document classifiers.

### 1.3. Proposed System:

As a pre-processing stage, document image analysis can facilitate and improve OCR(optical character recognition). by providing information about each document's visual layout . Furthermore, document information that is lost in OCR, such as typeface, graphics, and layout, is often stored and indexed using images or image descriptors.

With increased usage of digital means of document exchange and the growth in volume of digitized documents, the requirement of an automatic system capable of understanding their logical structures is felt significantly for better management of these documents including their efficient archiving, retrieval, information mining and so on. However, the development of an automatic system for classification of arbitrary document images into their respective true categories is computationally a non-trivial task in contrast to the case of human beings The effectiveness of this system hinges on that of the structure classification techniques employed and hence speaks for the utility of structure learning as an important aspect of document understanding.

In structured documents, the layout of text and graphics often reflects important information about genre. Therefore, documents of a category often share region-specific features. Additionally, the project explores two different initialization strategies: the first initializes the weights of the CNNs randomly; the second “transfers” weights learned in another visual classification task

## CHAPTER – 2

### LITERATURE SURVEY

In the past twenty years of document image analysis, research has oscillated between region-based analysis and whole image analysis, and simultaneously, between handcrafted features and machine-learned ones. The power of region-based analysis of document images has been clearly demonstrated in the domain of rigidly structured documents, such as forms and business letters .

To some extent, the classification of perfectly rigid documents (e.g., forms) can be reduced to the problem of template matching, and less-rigid document types (e.g., letters) can similarly be classified by fitting the geometric configuration of the document's components to one of several template configurations, via geometric transformations. However, for documents with more flexible structures, as considered herein, template-based approaches are inapplicable.

An alternative strategy is to treat document images holistically, and search for discriminative “landmark” features that may appear anywhere in the document. This strategy is sometimes called a “bag of visual words” approach, since it describes images with a histogram over an order less vocabulary of features .For example, finding a salutation in a document (potentially through OCR) is a good cue that the document is a letter, regardless of that feature's exact spatial position .This approach has been successful in retrieving and classifying a broader range of documents than the template based approaches, although the approach is less discriminating in the domain of rigid-template documents.

#### **2.1 Literature Review:**

1. Ramanujan Kashi & Gordon Wilfong has presented a paper named “**Document Image Layout Comparison and Classification**” in 2000

This paper describes features and methods for document image comparison and classification at the spatial layout level. The methods are useful for visual similarity based document retrieval as well as fast algorithms for initial document type classification without OCR. A novel feature set called interval encoding is introduced to capture elements of spatial

layout. This feature set encodes region layout information in fixed-length vectors which can be used for fast page layout comparison.

2. Yonatan Ami & Michael Fink presented a document named as “**uncovering shared structures in multiclass classification**” in 2014

This paper suggests a method for multiclass learning with many classes by simultaneously learning shared characteristics common to the classes, and predictors for the classes in terms of these characteristics.

3. Olga Russakovsky , Jia Deng & Hao Su presented a paper on “**Image Net Large Scale Visual Recognition Challenge**” in 2015

The Image Net Large Scale Visual Recognition Challenge is a benchmark in object category classification and detection on hundreds of object categories and millions of images. This paper describes the creation of this benchmark dataset and the advances in object recognition that have been possible as a result.

4 .ArindamDas, SaikatRoy ,UjjwalBhattacharya & SwapanK.Paru presented a paper on “**Document Image Classification with Intra-Domain Transfer Learning and Stacked Generalization of Deep Convolutional Neural Networks**” in 2018

In this article, a region-based Deep Convolutional Neural Network framework is presented for document structure learning. The contribution of this work involves efficient training of region based classifiers and effective ensemble for document image classification. A primary level of ‘inter-domain’ transfer learning is used by exporting weights from a pre-trained VGG16 architecture on the Image Net dataset to train a document classifier on whole document

## **CHAPTER – 3**

### **REQUIREMENT ANALYSIS**

#### **3.1 Introduction:**

Automatic classification of document images is an effective initial step of various Document Image Processing (DIP) tasks such as document retrieval, information extraction and text recognition, among others. The performance of a DIP system may be enhanced through efficient initial classification of an input document into a number of pre-determined categories. Automatic classification also has a significant role in indexing the documents of a Digital Library. It has been seen that any large volume of documents from different categories can be better organized provided these are first classified into several categories based on their structures.

#### **3.2 Functional requirements:**

With increased usage of digital means of document exchange and the growth in volume of digitized documents, the requirement of an automatic system capable of understanding their logical structures is felt significantly for better management of these documents including their efficient archiving, retrieval. The complexity of this task is increased due to inter-class similarity and intra-class variability issues in documents. For example, an advertisement may look like a news item or a form may be very dense with items arranged in multiple columns. The main idea of project is to classify documents of (Ryerson Vision Lab Complex Document Information Processing)dataset into 16 categories like form ,advertisement ,news article etc.

The challenge of document image analysis arises from the fact that within each document type, there exists a wide range of visual variability. . Intra-class variability renders spatial layout analysis difficult, and template-based matching impossible. Another issue is that documents of different categories often have substantial visual similarities. For instance, there exist advertisements that look like news articles, and questionnaires that look like forms, and so on. From the perspective of “visual styles”, some erroneous retrievals in such



circumstances may be justifiable, but in general the task of document image analysis is to classify and retrieve documents despite intra-class variability, and inter-class similarity.

### **3.3 Non Functional Requirements**

In requirements engineering, a non-functional requirement (NFR) is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviours. The plan for implementation of a non functional requirement is detailed in the system architecture because they are usually architecture significant requirements.

#### **3.3.1 Performance Requirement**

The performance of the model is the main aspect of this project, in order to meet the target the data and attributes must be specific and unambiguous. This requirement cannot be ignored at all cost. In order to assess the performance of the model we check the response time, scalability, the workload the project can take and the platform that the project is currently running on.

#### **3.3.2 Speed and Throughput Requirement**

Speed and throughput are also a factor as a client does not take a project which takes an input and consumes a lot of time to give the result so the developer has to focus and keep in mind about the speed of the response time of the model.

#### **3.3.3 Reliability Requirements**

The reliability requirement is typically part of a technical specifications document. They can be requirements that a company sets for its product and its own engineers or what it reports as its reliability to its customers. The essential elements of reliability requirement are attributes which are used, range of values and distribution of the values across the model.

#### **3.3.4 Scalability Requirement**

It refers to the system which can scale up when the work load becomes higher, the throughput, response time and performance must be similar and constant just like in the normal case scenario. The scalability required is often driven by the lifespan and the maturity of the system. In some cases there may be a problem with scalability specification is that it may not be economically viable to test the scalability, as it often requires additional hardware as the dataset increases.

### **3.4 Software Requirements:**

Programming Language : Python and required Libraries

Software Tools : Jupyter Notebook (or) ipython Notebook

### **3.5 Hardware Requirements:**

RAM : 8GB

Processor : i3 processor (or) equivalence

HardDisk:1TB

### **3.6 Algorithms**

#### **3.6.1 Convolutional neural network:**

I researched and developed the architecture of our convolutional neural network models. Convolutional neural networks are the primary neural architecture used for image analysis because they work with matrix inputs and have significantly fewer parameters that require training than other network designs. Where MLP would require each image to be flattened to a 1-dimensional input array with a length of the square of the image dimension, CNNs work with the image in its original matrix form. MLP puts weight and bias parameters on each node (pixel) and connects that node to every other node in the next layer all the way down the network. For a 100 x 100 image, this means that the MLP would have 10,000 input neurons. A network with one hidden layer of the same size as the input and 16 output nodes would require over 1.6 billion weight parameters. The CNN drastically shrinks the number of parameters by running small sets of weight values over the input values in the form of kernels.

#### **Input Layer**

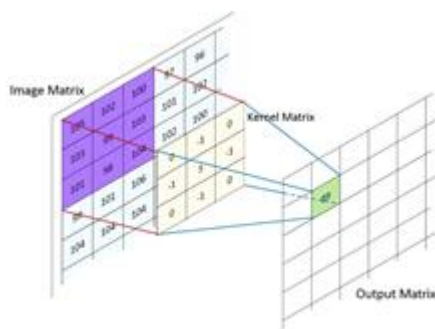
The input layer is a three-dimensional matrix with number of rows equal to the image height and number of columns equal to image width. For image analysis, the dimensions represent image height, width, and number of color channels, and the values represent pixel values

within the range [0, 255]. Gray scale images, as used in this project, have one color channel representing gray scale value.

### Convolutional Layers

The convolution layer consists of a set of learnable kernels, which are smaller matrices of randomly initialized weight values in a 3-dimensional matrix (number of color channels, height, width). In the feed forward process, the kernels slide (convolve) over each element of the input matrix and compute the sum of the element-wise multiplication between the kernel and the local image pixel values. This matrix multiplication results in a single value that maps to the feature map in the centre position of the section of image matrix under the kernel.

Each feature map is the resulting matrix from the computed values of a single kernel convolving over the whole of the input matrix. Aside from the number of kernels, convolutional layers have several adjustable parameters that control output size. Padding is a parameter that is used to maintain input size. As seen in the above figure, convolution results in a single value at the centre of the kernel, which means that the edges of the input are cut off and the output is  $n-1$  ( $n$  representing the height and width of the kernel) dimensions smaller than the input. To prevent this, layers of zeros or other values are added around outside of the input so that the first position of the kernel is centred on the upper-left pixel value. The number of padding layers is calculated by  $(n-1)/2$  with  $n$  representing kernel size. Stride is number of pixels that kernel jumps with each step of convolution. A stride of 1 means that the kernel moves one pixel row or column at a time and results in an output image that is the same size as the input image. Increasing stride to  $n$  decreases the output size  $n$ -fold (stride of 2 results in an output  $1/2$  the input size).



**Figure 3.5.1 Convolution multiplication with kernel**

### 3.6.2 VGG\_16 with transfer learning:

#### VGG-16

It is built by Visual Geometry Group . It consists of 16-layers with kernels of (3\*3) , Max pooling layers (used only 2\*2 size) and fully connected layers at end. It was originally trained on Image Net dataset.

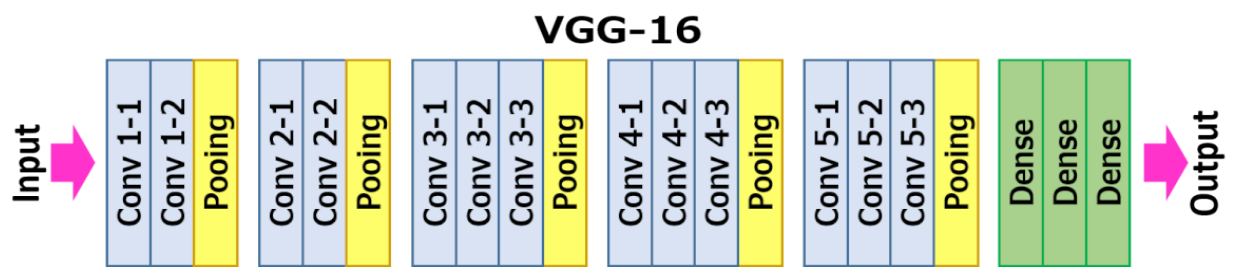


Figure 3.5.2 VGG16 architecture

#### 1. Transfer learning

Transfer learning is a popular method in computer vision because it allows us to **build accurate models in a timesaving way** (Rawat & Wang 2017). With transfer learning, instead of starting the learning process from scratch, you start from patterns that have been learned when solving a different problem. This way you leverage previous learnings and avoid starting from scratch. Take it as the deep learning version of Chartres' expression 'standing on the shoulder of giants'.

In computer vision, transfer learning is usually expressed through the use of **pre-trained models**. A pre-trained model is a model that was trained on a large benchmark dataset to solve a problem similar to the one that we want to solve. Accordingly, due to the computational cost of training such models, it is common practice to import and use models from published literature (e.g. VGG, Inception, MobileNet).

## CHAPTER-4

### TECHNOLOGIES USED

#### 4.1 Machine learning:

Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. **Machine learning focuses on the development of computer programs** that can access data and use it learn for themselves.

The process of learning begins with observations or data, such as examples, direct experience, or instruction, in order to look for patterns in data and make better decisions in the future based on the examples that we provide. **The primary aim is to allow the computers learn automatically** without human intervention or assistance and adjust actions accordingly.

#### 4.2 Deep Learning:

Deep learning is an aspect of artificial intelligence (AI) that is concerned with emulating the learning approach that human beings use to gain certain types of knowledge. At its simplest, deep learning can be thought of as a way to automate predictive analytics.

While traditional machine learning algorithms are linear, deep learning algorithms are stacked in a hierarchy of increasing complexity and abstraction. To understand deep learning, imagine a toddler whose first word is dog. The toddler learns what a dog is (and is not) by pointing to objects and saying the word dog. The parent says, "Yes, that is a dog," or, "No, that is not a dog." As the toddler continues to point to objects, he becomes more aware of the features that all dogs possess. What the toddler does, without knowing it, is clarify a complex abstraction (the concept of dog) by building a hierarchy in which each level of abstraction is created with knowledge that was gained from the preceding layer of the hierarchy.

### 4.3 Python:

Python is an interpreter, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace. It provides constructs that enable clear programming on both small and large scales. Van Rossum led the language community until stepping down as leader in July 2018. Python interpreters are available for many operating systems.

### 4.4 Jupyter Notebook:

Project Jupyter is a nonprofit organization created to "develop open-source software, open-standards, and services for interactive computing across dozens of programming languages". Spun-off from IPython in 2014 by Fernando Pérez, Project Jupyter supports execution environments in several dozen languages. Project Jupyter's name is a reference to the three core programming languages supported by Jupyter, which are Julia, Python and R. Project Jupyter has developed and supported the interactive computing products Jupyter Notebook, Jupyter Hub, and Jupyter Lab, the next-generation version of Jupyter Notebook.

Jupyter Notebook (formerly IPython Notebooks) is a web-based interactive computational environment for creating Jupyter notebooks documents. It is a JSON document, following a versioned schema, and containing an ordered list of input/output cells which can contain code, text (using Markdown), mathematics, plots and rich media, usually ending with the ".ipynb" extension.

## CHAPTER –5

### SYSTEM DESIGN

System Design is transition from a user oriented document to programmers or database personnel. The design is a solution, how to approach to the creation of a new system . This is composed of several steps. It provides the understanding and procedural details necessary for implementing the system recommended in the feasibility study .

Designing goes through logical and physical stages of development , logical design reviews the present physical system , prepare input and output specification , details of implementation plan and prepare a logical design walkthrough.

The database table are designed by analyzing functions involved in the system and format of the fields is also designed . The fields in the database table should define their role in the system . The unnecessary fields should be avoided because it affects the storage areas of the system. Then in the input and output screen design, the design should be made user friendly. The menu should be precise and compact.

#### 5.1 Software Design

In designing the software following principles are followed

1. **Modularity and partitioning:** Software is designed such that, each system should consists hierarchy of modules and serve to partition into separate function.
2. **Coupling:** Modules should have little dependence on other modules of a system.
3. **Cohesion:** Modules should carry out in a single processing function.
4. **Shared use:** Avoid duplication by allowing a single module called by other that need the function it provides.

## 5.2 System Architecture:

A system architecture or systems architecture is the conceptual model that defines the structure, behaviour, and more views of a system. An architecture description is a formal description and representation of a system, organized in a way that supports reasoning about the structures and behaviours of the system.

A system architecture can consist of system components and the sub-systems developed, that will work together to implement the overall system. There have been efforts to formalize languages to describe system architecture, collectively these are called architecture description languages (ADLs).

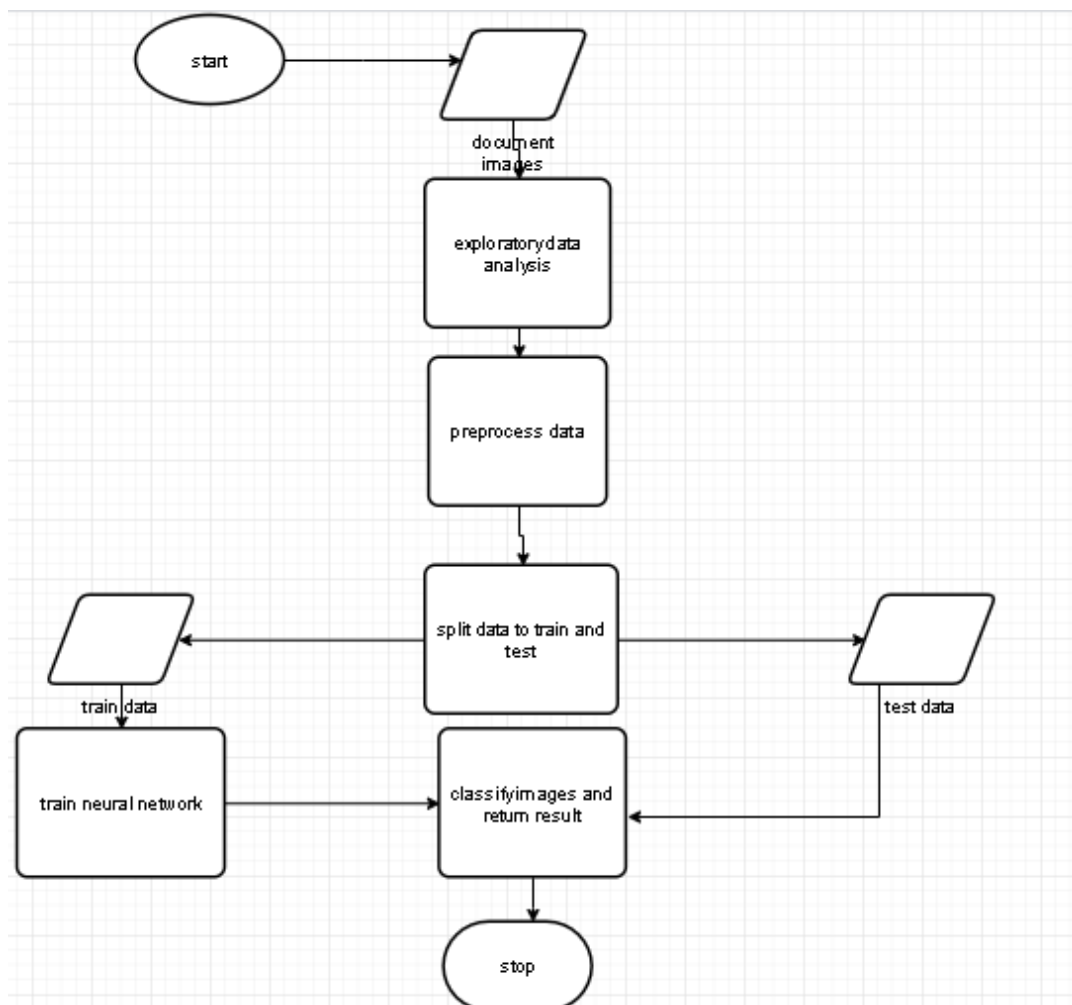


Figure-5.1 System Architecture



### 5.3 Data Flow Diagram:

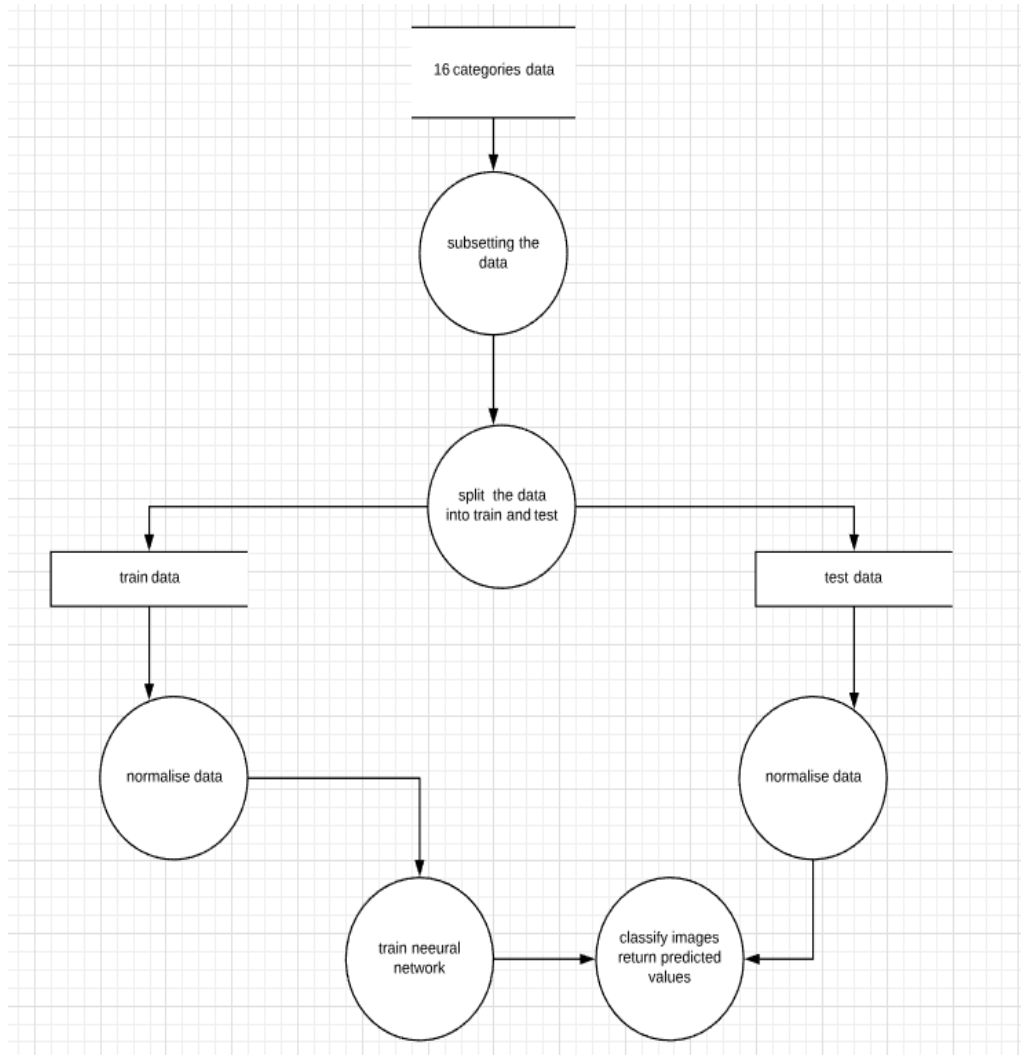


Figure-5.2 Data flow diagram

## 5.4 Modules and their Functionality:

We can define the machine learning workflow in 3 stages.

1. Gathering data & Data Cleaning
2. Data pre-processing
3. Researching the model that will be best for the type of data
4. Training and testing the model
5. Evaluation

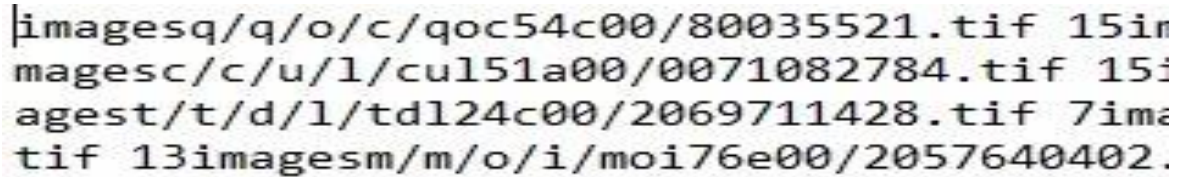
The machine learning model is nothing but a piece of code; an engineer or data scientist makes it smart through training with data. So, if you give garbage to the model, you will get garbage in return, i.e. the trained model will provide false or wrong predictions.

### 5.4.1 Gathering Data and Data Cleaning:

The process of gathering data depends on the type of project we desire to make, if we want to make an ML project that uses real-time data, then we can build an IoT system that using different sensors data. The data set can be collected from various sources such as a file, database, sensor and many other such sources but the collected data cannot be used directly for performing the analysis process as there might be a lot of missing data, extremely large values, unorganized text data or noisy data.

My initial individual task consisted of considerable data cleaning. Images were stored in a format of each image residing in its own folder with multiple layers of parent folders with nonsensical names, all stored in the images directory. Image labels came in the form of a text file of all image file paths from the image directory and the numeric label with a space between the path and label with no space between the label and the next image path. The label text file was split on the image directory name and again on the space between path and label to create a nested list with each sub-list representing the image path and label. Next, a new image directory was created as well as a reference dictionary mapping numeric label to document type. New subdirectories were made for training, testing, and validation, with document type folders for each dataset folder. The image sub-path was then appended to the end of the new parent directory and moved to the individual document type folder via

mapping the numeric label to the reference dictionary. Once sorted, the training, testing, and validation datasets set was randomly subsetting to create workable, class uniform datasets.



```
imagesq/q/o/c/qoc54c00/80035521.tif 15  
imagesc/c/u/l/cul51a00/0071082784.tif 15  
agest/t/d/l/tdl24c00/2069711428.tif 7  
imagesm/m/o/i/moi76e00/2057640402.tif 13
```

figure-5.4.1 dataset

### 5.4.2 Data pre-processing

Data pre-processing is one of the most important steps in machine learning. It is the most important step that helps in building machine learning models more accurately. In machine learning, there is an 80/20 rule. Every data scientist should spend 80% time for data pre-processing and 20% time to actually perform the analysis.

- **What is data pre-processing?**

Data pre-processing is a process of cleaning the raw data i.e. the data is collected in the real world and is converted to a clean data set. In other words, whenever the data is gathered from different sources it is collected in a raw format and this data isn't feasible for the analysis. Therefore, certain steps are executed to convert the data into a small clean data set, this part of the process is called as data pre-processing.

- **Why do we need it?**

As we know that data pre-processing is a process of cleaning the raw data into clean data, so that can be used to train the model. So, we definitely need data pre-processing to achieve good results from the applied model in machine learning and deep learning projects

I then researched image pre-processing techniques that would be appropriate for the images and performed exploratory data analysis. I determined that histogram equalization would be ideal for the dataset. Histogram equalization is an image processing method that attempts to linearize the histogram of the cumulative sum of pixel values. This brings out areas of lower local contrast by equally distributing high intensity values. As can be seen in the above figure,

histogram equalization casts more pixel values towards the lower end of the grayscale spectrum (black) and increases overall images contrast. I then explored images for the distribution of dimensions and determined that all images had a height of 1000 pixels and almost all images had a width between 750 and 800 pixels, meaning that relative aspect ratio between images would be mostly preserved in resizing. Finally, I looked into how features were affected by resizing by checking images at half, quarter, eighth, and sixteenth sizes. As can be seen below, images lose distinguishable features at eighth size, therefore keeping images at 224x224 is the minimum size for preserving enough clear features for classification.

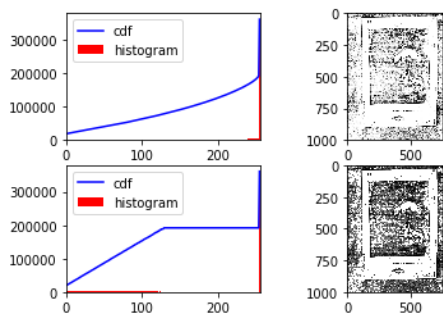


Figure 5.4.2 Histogram Equalization with CDF distribution

- **Histogram Equalization**

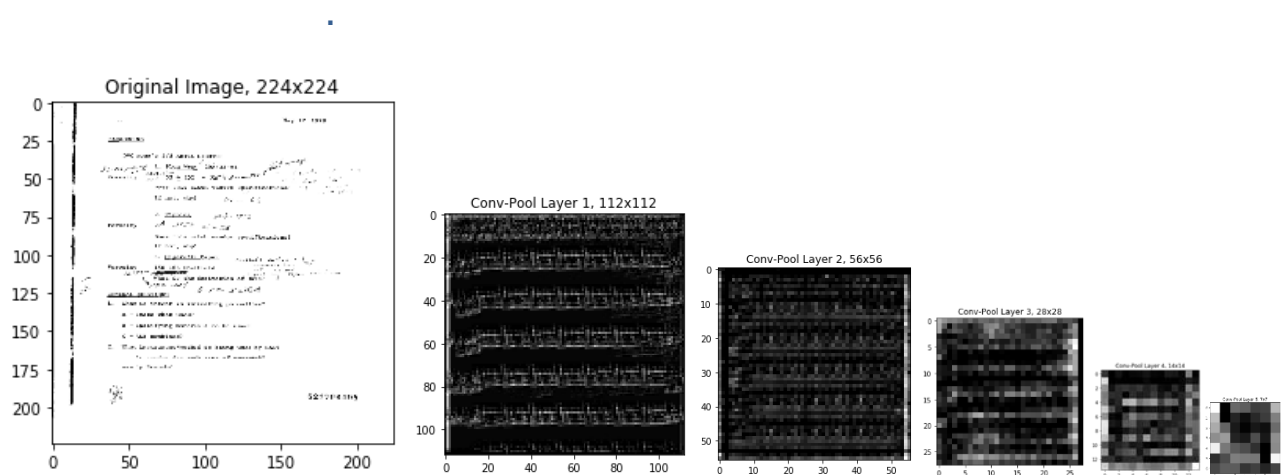
I next researched pre-processing methods and determined that histogram equalization would likely have a positive impact on image recognition. Histogram equalization attempts to linearize the histogram of the cumulative sum of pixel values as much as possible by applying a non-linear transformation to each pixel of the input image. This brings out areas of lower local contrast by equally distributing high intensity values. The equation for the transformed pixel value is the cumulative sum of the frequency of pixel values up to that level. A transformation is then applied to this new value to make the cumulative distribution linear.

### 5.4.3 Researching that model that is suitable for image data

- **Convolutional Neural Networks:**

Convolutional neural networks are used for this project as they take matrices as inputs and use a relatively small number of trainable parameters to learn features. As images are simply matrices of a large number of pixel values, CNN's are ideal for this task. As a group, we developed a workable dataset, explored CNN architecture and tested different deep learning frameworks, including PyTorch and Keras.

In this project, we explored using CNN's in both PyTorch and Keras to examine if the architecture will produce an effective classifier for document images.



**Fig 5.4.3 Feature extraction through layers**

I conducted all my model training in PyTorch. I first devised a custom wrapper around PyTorch's Dataset class to load the data. This method first involved creating a reference csv for each training and testing datasets that contained the full image path and the numeric label. This csv was then passed through the custom data loader and each image was read using the path with OpenCV. The image and label were passed through PyTorch's DataLoader class and iterated over to create batches for training.

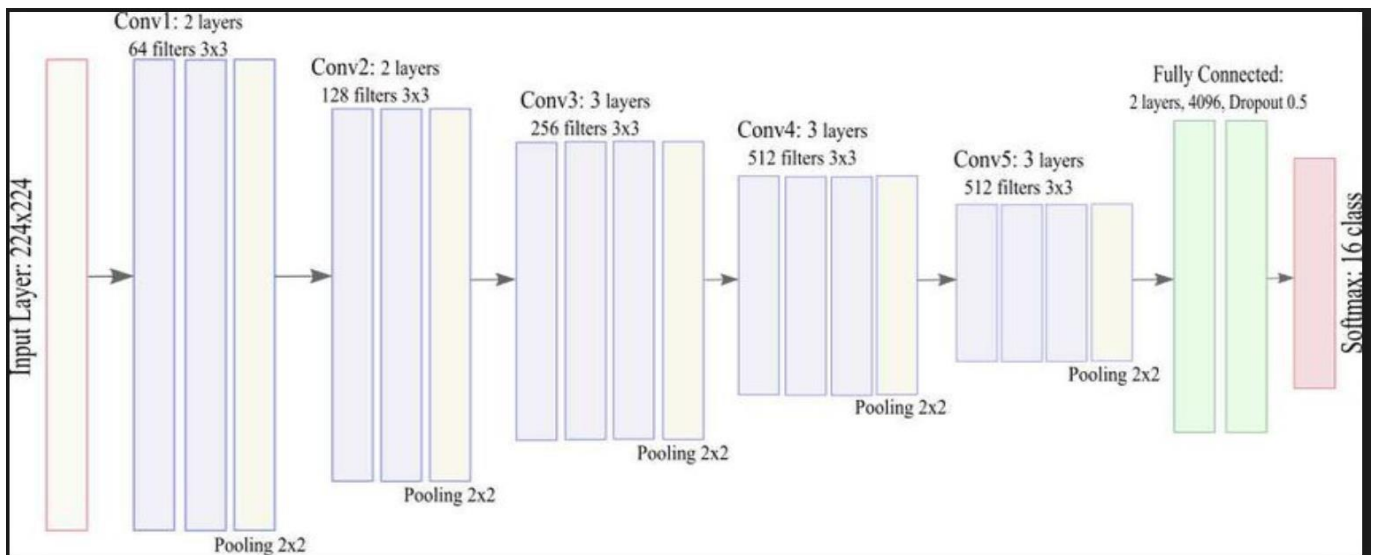
After loading the image data, I built the CNN model with varying parameters. Each layer sequence consisted of a convolution layer, batch normalization, ReLU activation, and a max pooling layer. The convolution layer required the specification of input channels (first layer is

1 for grayscale input, rest are number of feature maps of previous layer), output channels (number of feature maps), kernel size, padding size ( $(\text{kernel\_size}-1)/2$  to maintain image size through convolution), and stride. After the convolutional layers, a fully connected layer of input size image size \* number of feature maps ( $7 \times 7 \times 32 = 1568$  for most implementations of the model) and output size selected number of neurons was added. Dropout was implemented after this layer, followed by a fully connected layer of output size of 16 (number of classes) and a softmax activation for class probability. Models were tested with varying image sizes, pre-processing techniques, batch sizes, learning rates, and kernel sizes.

- **Vgg16 with transfer learning:**

VGG-16 is a convolutional neural network trained on more than a million images from Image Net database. The network is 16 layers deep and can classify images into 1000 categories. The default input shape for VGG-16 is  $224 \times 224$

**Figure-5.4.4 Architecture of VGG-16 for my project**



VGG-16 is imported from `keras.applications.vgg16`. Keras Applications are deep learning models that are based on pre-trained weights.

Format:

```
VGG16(include_top = True, weights = 'None', input_tensor = None, input_shape
= None, pooling = None, classes=1000)
```

## 5.5 Project methodology:

For the project, we have chosen Spiral Model of Software Engineering, since this model is a combination of iterative development with incremental release approach. The training of the model, testing, validation and verification of the model performance is done in an iterative manner, with the performances and working prowess saved and carried over to next version in an incremental fashion.

This is the reason we chose to implement the project in a spiral fashion, because we focus both on iterative development and incremental deployment for better performance, credibility, accountability and higher performance to loss ratios for the project.

Spiral model is technically best suitable to this due to the machine learning base chosen for the project. New content is added, the model learns from every instance of execution and also updates itself to perform better and better. This requires a model that allows us to keep the development focused on iterative development of the project while also focusing on incremental progression maintenance. This helps in developing an all rounded application, without having to worry about development paradigms and shifts, with the development going in progressive succession. This also allows to keep track of risks, providing an efficient way of analysing, managing, monitoring and mitigating risks to avoid losses and improve working of the system. Spiral model provides best possible way of implementing the said technology requirements, with perfect balance between development environment and control over risks.

## **5.6 UML Diagrams**

### **5.6.1 UML Concepts:**

The Unified Modeling Language (UML) is a standard language for writing software blue prints.

The UML is a language for

1. Visualizing
2. Specifying
3. Constructing
4. Documenting the artifacts of a software intensive system.

The UML is a language which provides vocabulary and the rules for combining words in that vocabulary for the purpose of communication. A modeling language is a language whose vocabulary and the rules focus on the conceptual and physical representation of a system. Modeling yields an understanding of a system.

### **5.6.2. Building Blocks of the UML:**

The vocabulary of the UML encompasses three kinds of building blocks:

1. Things
2. Relationships
3. Diagrams



## 1. Things in the UML:

Things are the abstractions that are first-class citizens in a model; relationships tie these things together; diagrams group interesting collection of things

Structural things are the nouns of UML models. The structural things used in the project design are: First, a class is a description of a set of objects that share the same attributes, operations, relationships and semantics

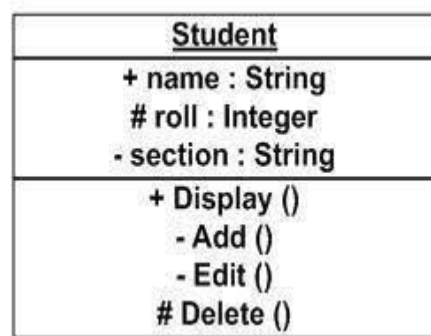


Figure 5.6.1 Class

Second, a use case is a description of set of sequence of actions.

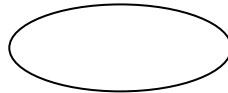


Fig 5.6.2 Use Case

Third node is a physical element that exists at runtime and represents a computational resource, generally having at least some memory and often processing capability.

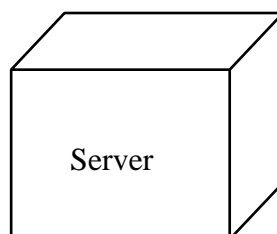


Fig 5.6.3 Node

Behavioral things are the dynamic parts of UML models. They are as follows.

**1. Interaction:**

An interaction is a behavior that comprises a set of messages exchanged among a set of objects within a particular context to accomplish a specific purpose. An interaction involves a number of other elements, including messages, action sequences (the behavior invoked by a message, and links the connection between objects).

**2. Association:**

An association is a structural relationship that describes a set links, a link being a connection among objects. Aggregation is a special kind of association, representing a structural relationship between a whole and its parts.

## **2. Relationships in the UML:**

There are four kinds of relationships in the UML:

1. Dependency
2. Association
3. Generalization
4. Realization

A dependency is a semantic relationship between two things in which a change to one thing may affect the semantics of the other thing (the dependent thing).

A generalization is a specialization/generalization relationship in which objects of the specialized element (the child) are substitutable for objects of the generalized element (the parent).

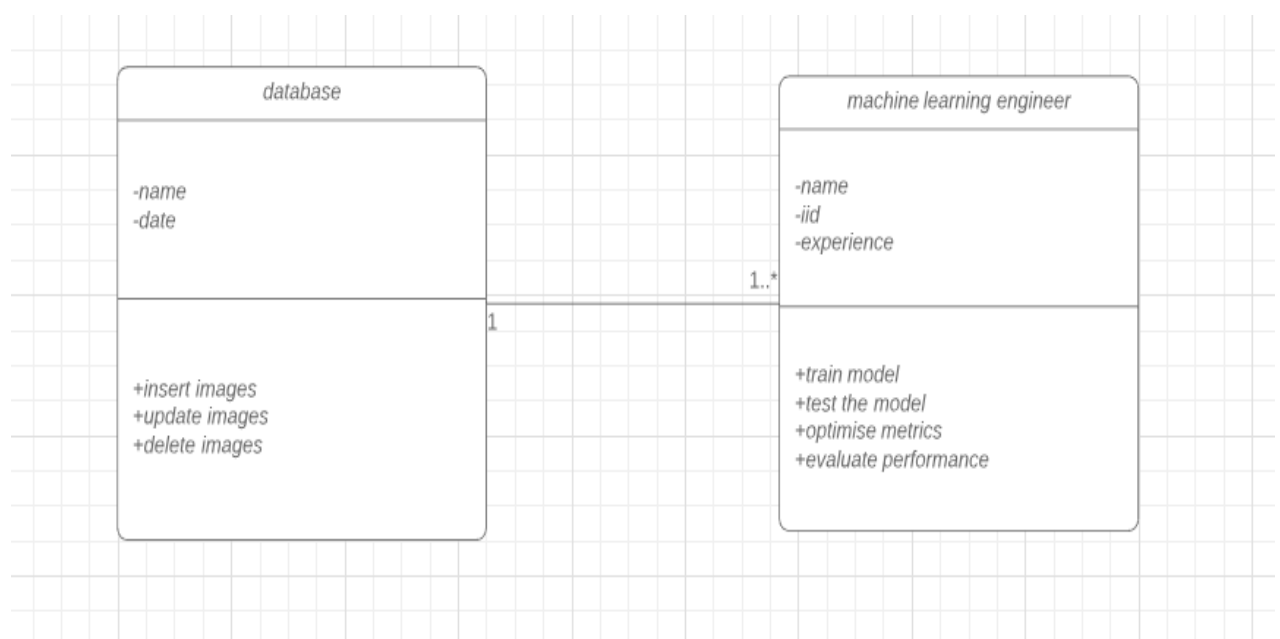
An Association is a relationship between classifiers which is used to show that instances of classifiers could be either linked to each other or combined logically or physically into some aggregation

A realization is a semantic relationship between classifiers, where in one classifier specifies a contract that another classifier guarantees to carry out.

### 5.6.3 Class Diagram

The class diagram is the main building block of object oriented modelling. It is used for both conceptual modelling and detailed modelling. Class diagrams can also be used for data modelling.

The class diagram consists of three classes namely data engineer, machine learning engineer and database. The task of data engineer is to collect the data, clean it by removing null and missing values and preprocess data by visualising them. Machine learning engineer is given the preprocessed data he creates pipeline and give it to model. Testing the model on new unseen points and optimising single metric until achieving the baseline standard score of that metric. data engineer depends on database of the organization which provides data.



**Fig 5.6.4 Class Diagram**

### 5.6.4 Use Case Diagram

Use case diagrams have accessibility upon the overall system with specific to the data insertion, deletion, updating and queries. They are the highest authorities within the system which have maximum control upon the entire database. A use case diagram at its simplest is a representation of a user's interaction with the system and depicting the specifications of a use case.

A use case diagram is a graph factors set of use cases enclosed by a system boundary, communication associations between the actors and users and generalization among use cases. The use case model defines the outside (actors) and inside (use case) of the systems behavior.

Use case diagram is quite simple in nature and depicts two types of elements one representing the business roles and the other representing the business processes.

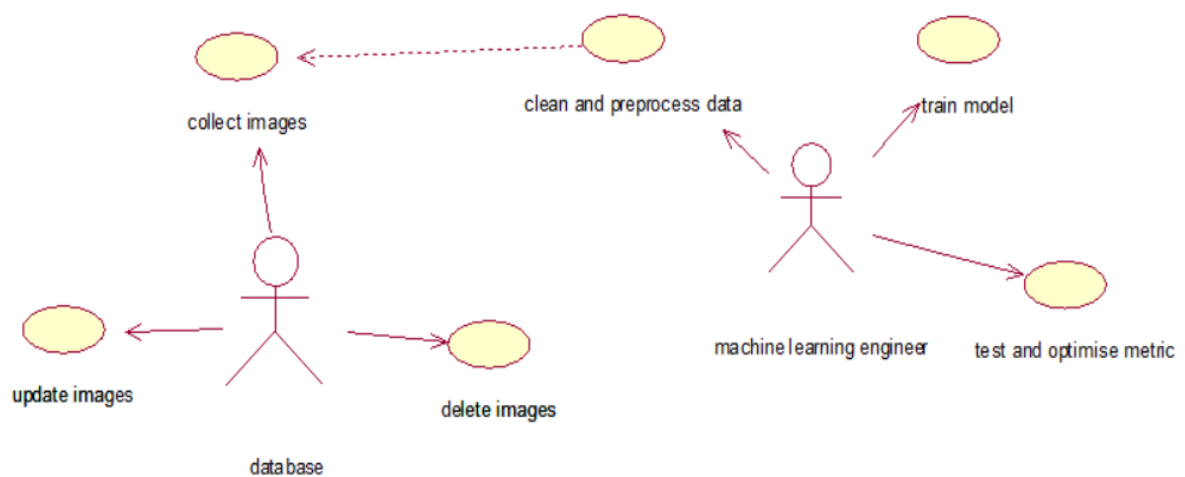


Figure 5.6.5 Use case Diagram

### 5.7.5 Sequence Diagram

A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called event diagrams or event scenarios.

A sequence diagram shows, as parallel vertical lines (lifelines), different processes or objects that live simultaneously, and as horizontal arrows, the messages exchanged between them, in the order in which they occur.

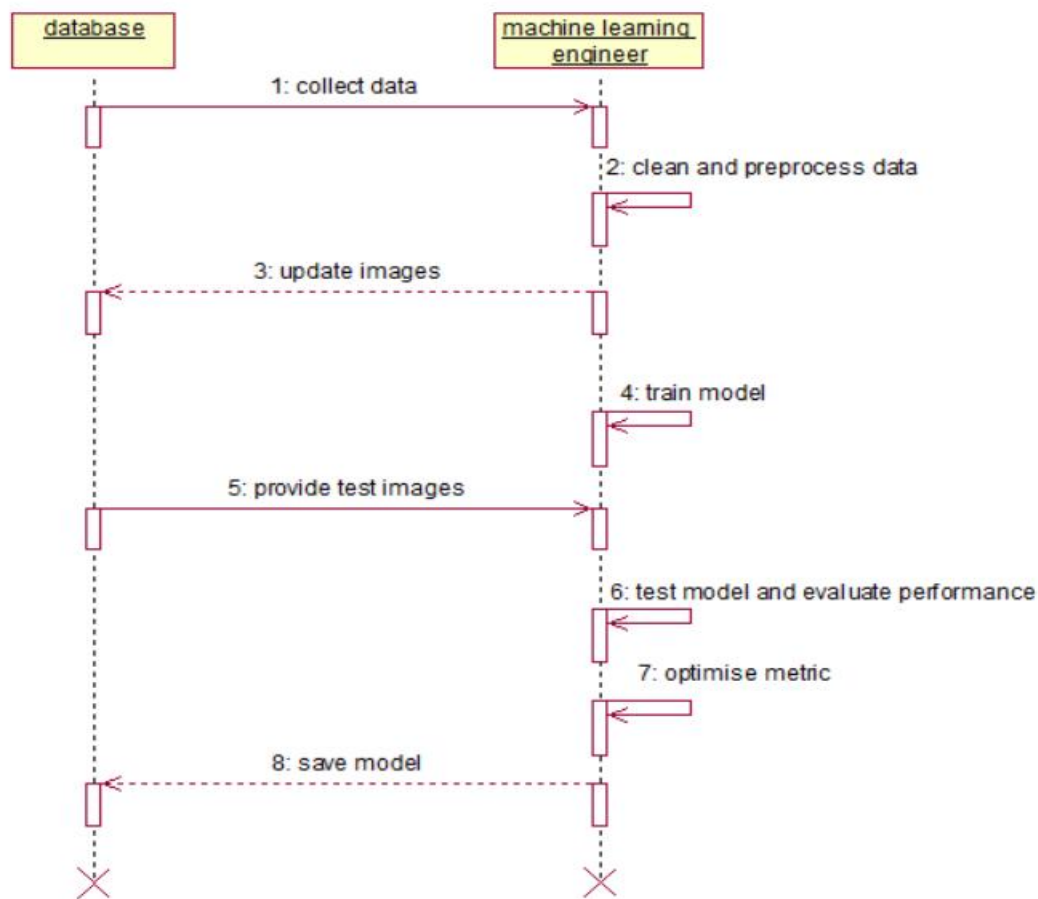


Figure 5.6.6 Sequence Diagram

### 5.7.6 Collaboration Diagram:

A collaboration diagram is a type of visual presentation that shows how various software objects interact with each other within an overall IT architecture and how users can benefit from this collaboration. A collaboration diagram often comes in the form of a visual chart that resembles a flow chart. It can show, at a glance, how a single piece of software complements other parts of a greater system

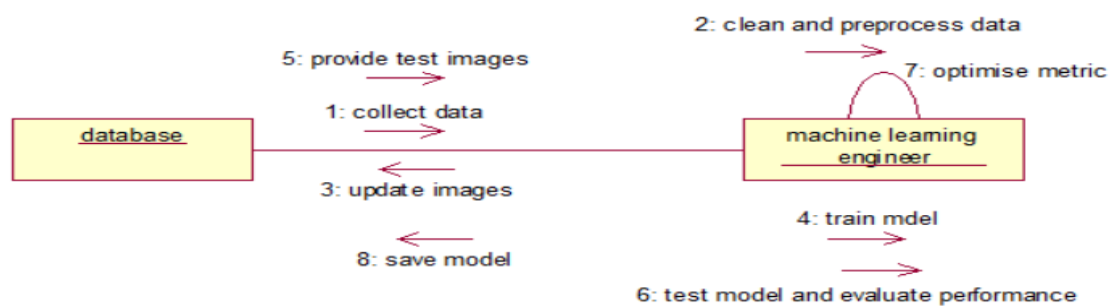


Figure 5.6.7 Collaboration Diagram

### 5.7.7 Component Diagram:

Component diagrams are used to visualize the organization and relationships among components in a system. These diagrams are also used to make executable systems. Component diagram is a special kind of diagram in UML. The purpose is also different from all other diagrams discussed so far. It does not describe the functionality of the system but it describes the components used to make those functionalities.

The purpose of the component diagram can be summarized as:

1. Visualize the components of a system.
2. Construct executables by using forward and reverse engineering.
3. Describe the organization and relationships of the components.

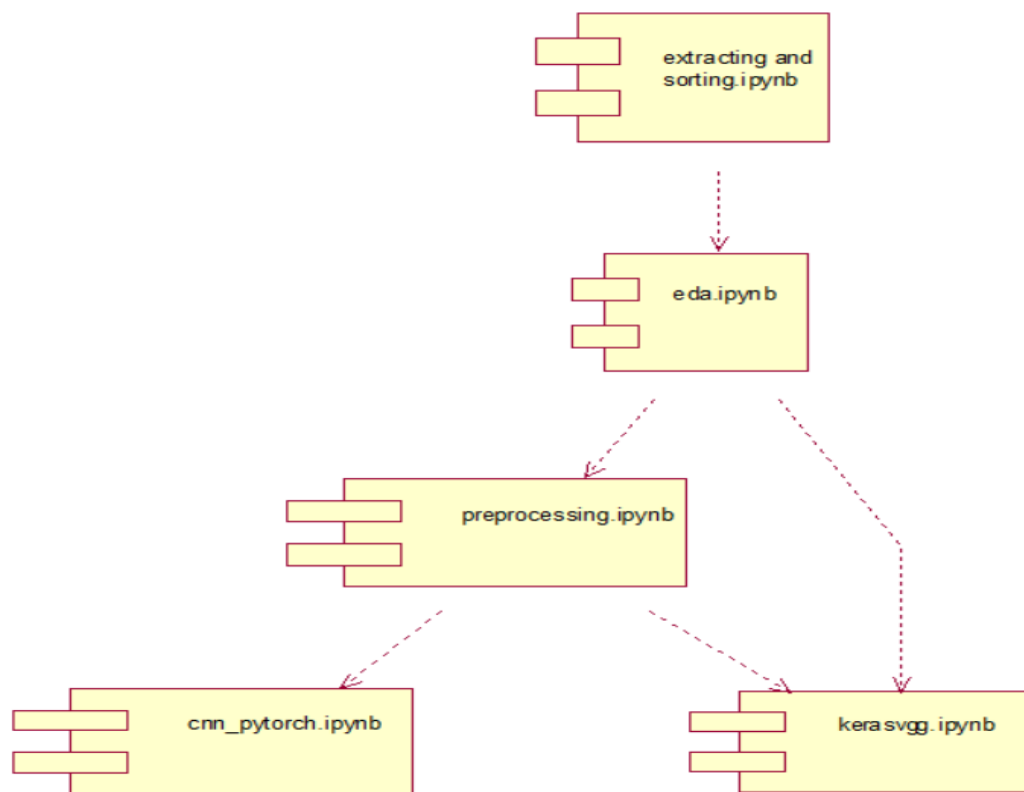


Figure -5.6.8 Component Diagram



## CHAPTER-6

### IMPLEMENTATION

#### 6.1 Data loading and cleaning:

##### Sample code:

```
#loading basic packages

import pandas as pd
import numpy as np

#converting .txt file to .csv file
import csv

txtfile = r"./labels/train.txt"
csvfile = r"train.csv"

with open(txtfile, 'r') as infile, open(csvfile, 'w') as outfile:

    stripped = (line.strip() for line in infile)
    lines = (line.split(",") for line in stripped if line)
    writer = csv.writer(outfile)
    writer.writerows(lines)

#Loading data
df=pd.read_csv("./train.csv",header=None)

#naming columns in data
df.columns=['image']

df = pd.DataFrame(df['image'].str.split(' ',1).tolist(), columns = ['image','class'])

#checking if it is done or not
df.head()

df.to_csv('train.csv')

df['image']=df['image'].astype(str)
df['class']=df['class'].astype(int)
```

```

#trial
for i in range(len(df['image'])):
    my_file = Path("./images/"+df['image'][i])
print(my_file)

#Creating 15 folders for 15 classes of images in desired location
import os

class_name={'0':'letter','1':'form','2':'email','3':'handwritten','4':'advertisement','5':'scientific
report','6':'scientific publication', '7':' specification', '8': 'filefolder' ,9': 'news article' ,10':
'budget', '11': 'invoice' ,12':'presentation','13':'questionnaire','14':'resume','15':'memo'}

path = './'

i = 0
for j in class_name.keys():
    os.rename(path+'/' +j, path+'/' +class_name[j])
    i=i+1

```

## 6.2 Data Preprocessing:

### Sample code:

for image in examples:

```
    directory = os.path.dirname(image)
```

```
    doctype = directory.rsplit("\\", 1)[-1]
```

```
    img = cv2.imread(image, 0)
```

```
    equ = cv2.equalizeHist(img)
```

```
    res = np.hstack((img,equ))
```

```
    plt.subplot(111)
```

```
    plt.imshow(res, cmap='Greys_r')
```

```
plt.title('{}'.format(doctype))
```

```
plt.xticks([])
```

```
plt.yticks([])
```

```
plt.show()
```

```
# -----
```

### 6.3 Training and testing model

```
# CNN Model (2 conv layer)
```

```
class CNN(nn.Module):
```

```
    def __init__(self):
```

```
        super(CNN, self).__init__()
```

```
        self.layer1 = nn.Sequential(
```

```
            #1 input for grayscale, # of feature maps,
```

```
            nn.Conv2d(1, 32, kernel_size=9, padding=4, stride=1),
```

```
            nn.BatchNorm2d(32),
```

```
            nn.ReLU(),
```

```
            nn.MaxPool2d(2))
```

```
        self.layer2 = nn.Sequential(
```

```
            nn.Conv2d(32, 32, kernel_size=7, padding=3),
```

```
            nn.BatchNorm2d(32),
```

```
            nn.ReLU(),
```

```
            nn.MaxPool2d(2))
```

```
        self.layer3 = nn.Sequential(
```

```
            nn.Conv2d(32, 32, kernel_size=5, padding=2),
```

```
            nn.BatchNorm2d(32),
```

```
            nn.ReLU(),
```

```
            nn.MaxPool2d(2))
```

```
        self.layer4 = nn.Sequential(
```

```
            nn.Conv2d(32, 32, kernel_size=5, padding=2),
```

```
            nn.BatchNorm2d(32),
```

```
            nn.ReLU(),
```

```
            nn.MaxPool2d(2))
```

```
        self.layer5 = nn.Sequential(
```

```
            nn.Conv2d(32, 32, kernel_size=3, padding=1),
```

```
            nn.BatchNorm2d(32),
```

```
            nn.ReLU(),
```

```
            nn.MaxPool2d(2))
```

```
        self.fc1 = nn.Linear(7 * 7 * 32, 128)
```

```

        self.drop1 = nn.Dropout(0.2)
        self.fc2 = nn.Linear(128, 16)

    def forward(self, x):
        out = self.layer1(x.float())
        out = self.layer2(out)
        out = self.layer3(out)
        out = self.layer4(out)
        out = self.layer5(out)
        out = out.view(out.size(0), -1)
        out = self.fc1(out)
        out = self.drop1(out)
        out = self.fc2(out)
        return out

# -----
cnn = CNN()
cnn.cuda()
# -----
# Loss and Optimizer
criterion = nn.CrossEntropyLoss()
optimizer = torch.optim.Adam(cnn.parameters(), lr=learning_rate)
# -----
# Train the Model

losses = []

start_time = time.time()
for epoch in range(num_epochs):
    print ("Starting Epoch {}".format(epoch + 1))
    train_iter = iter(train)
    i = 0
    for images, labels in train_iter:

        images = Variable(images).cuda()
        labels = Variable(labels).cuda()

        # Forward + Backward + Optimize
        optimizer.zero_grad()
        outputs = cnn(images)
        loss = criterion(outputs, labels)

```

```

        loss.backward()
        optimizer.step()
        i += 1

    if (i) % 50 == 0:
        print('Epoch {}/{}, Iter {}/{}, Loss: {}'.format(epoch + 1, num_epochs, i,
            (train_loader.data_len / batch_size), loss.item()))

    losses.append(loss.item())

print("Epoch Done")

print("--- %s seconds ---" % (time.time() - start_time))
# -----

```

## 6.4 Evaluate model

# Test the Model

cnn.eval() # Change model to 'eval' mode (BN uses moving mean/var).

correct = 0

total = 0

im\_labels = []

im\_preds = []

for i, (images, labels) in enumerate(test\_iter):

images = Variable(images).cuda()

outputs = cnn(images)

\_, predicted = torch.max(outputs.data, 1)

total += labels.size(0)

correct += (predicted.cpu() == labels).sum()

if (i) % 10 == 0:

print('Epoch {}/{}, Iter {}/{}, Loss: {}'.format(epoch + 1, num\_epochs, i,
 (test\_loader.data\_len / batch\_size), loss.item()))

im\_labels.append(labels.cpu().numpy())

im\_preds.append(predicted.cpu().numpy())

# -----

print("Test Accuracy of the model on the test images: {}".format(100 \* correct / total))

# -----

**Sample code for vgg16:**

```
vgg_conv=VGG16(include_top=False,weights='imagenet',input_shape=(img_row
s, img_cols, 3))
print(vgg_conv.summary())

# Freeze the layers except the last 4 layers
for layer in vgg_conv.layers[:-4]:
    layer.trainable = False
#Check the trainable status of the individual layers
for layer in vgg_conv.layers:
    print(layer, layer.trainable)

from keras import models
from keras import layers
from keras import optimizers

# Create the model
model = models.Sequential()

# Add the vgg convolutional base model
model.add(vgg_conv)

# Add new layers
model.add(layers.Flatten(input_shape=train_x.shape[1:]))
model.add(layers.Dense(100, activation='relu'))
model.add(layers.Dropout(0.7))
model.add(layers.Dense(16, activation='softmax'))

# Show a summary of the model. Check the number of trainable parameters
model.summary()

vgg_conv=VGG16(include_top=False,weights='imagenet',input_shape=(img_row
s, img_cols, 3))
print(vgg_conv.summary())

model = models.Sequential()

# Add the vgg convolutional base model
model.add(vgg_conv)

# Add new layers
model.add(layers.Flatten(input_shape=vgg_conv.output_shape[1:]))
model.add(layers.Dropout(0.5))
```

```
model.add(layers.Dense(100, activation='relu',kernel_initializer='he_normal'))
model.add(layers.Dropout(0.5))
model.add(layers.Dense(100, activation='relu',kernel_initializer='he_normal'))
model.add(layers.Dropout(0.5))
model.add(layers.Dense(16,
activation='softmax',kernel_initializer='glorot_uniform'))
#model= Model(inputs=vgg_conv.input, output=model(model.output))
# Show a summary of the model. Check the number of trainable parameters
model.summary()

model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.Adam(lr=1e-4, beta_1=0.9, beta_2=0.999,
epsilon=None, decay=0.0, amsgrad=False),
              metrics=['accuracy'])
history=model.fit(train_x, train_y,
batch_size=batch_size,
              epochs=epochs,
              verbose=1,
              validation_data=(eval_x, eval_y))
score = model.evaluate(test_x, test_y, verbose=0)
print("Test loss:", score[0])
print("Test accuracy:", score[1])
```

## **UNIT-7**

### **TESTING**

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the

Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

#### **7.1 Types of testing**

##### **7.1.1 Unit testing:**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application. It is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

##### **7.1.2 Integration testing:**

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfactory, as shown by successful unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.



### 7.1.3 Functional Test :

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centred on the following items:

- Valid Input : identified classes of valid input must be accepted.
- Invalid Input : identified classes of invalid input must be rejected.
- Functions : identified functions must be exercised.
- Output : identified classes of application outputs must be exercised.
- Systems/Procedures : interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

### 7.1.4 System Test:

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

### 7.1.5 White Box Testing:

White Box Testing is a testing in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is used to test areas that cannot be reached from a black box level.

### 7.1.6 Black Box Testing:

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works..

## 7.2 Unit Testing:

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

### **Test strategy and approach**

Field testing will be performed manually and functional tests will be written in detail.

### **Test objectives**

- All field entries must work properly.
- The entry screen, messages and responses must not be delayed.

### **Features to be tested**

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

## 7.3 Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

## 7.4 Acceptance Testing

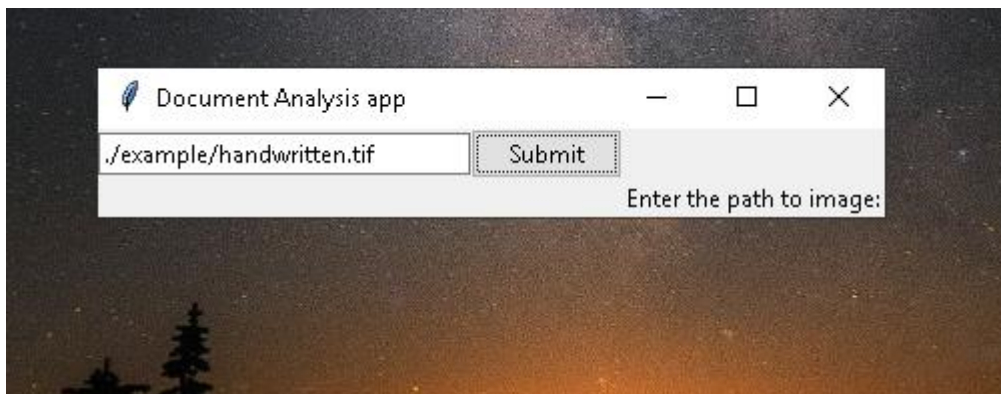
User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

## UNIT-8

### OUTPUT SCREENS

#### 8.1 Running the application:

When you run the Example.py file in command prompt we will be getting a pop box in which we can enter the path of the image .



Output Screen 8.1 Document analysis application

The output will be displayed as follows so we can view the label for that particular input image



Output Screen 8.2 Output label for the given input

## 8.2 Data loading and cleaning

```

In [1]: #Loading basic packages
import pandas as pd
import numpy as np

In [4]: #Repaeting the process to test.txt
#converting .txt file to .csv file
import csv
txtfile = r"./labels/train.txt"
csvfile = r"train.csv"
with open(txtfile, 'r') as infile, open(csvfile, 'w') as outfile:
    stripped = (line.strip() for line in infile)
    lines = (line.split(",") for line in stripped if line)
    writer = csv.writer(outfile)
    writer.writerows(lines)

In [5]: #Loading data
df=pd.read_csv("./train.csv",header=None)

In [6]: #naming columns in data
df.columns=['image']

In [7]: #Splitting the column into 2 columns image and class using space ' ' as delimiter
df = pd.DataFrame(df['image'].str.split(' ',1).tolist(),
                  columns = ['image','class'])

In [8]: #checking if it is done or not
df.head()

Out[8]:

```

	image	class
0	imagesq/q/o/c/qoc54c00/80035521.tif	15
1	imagese/e/w/c/ewc23d00/513280028.tif	1
2	imagesw/w/b/t/wbt26e00/2053453161.tif	7
3	imagesm/m/k/m/mkm05e00/2040792992_2040792994.tif	10
4	imageso/o/e/x/oex80d00/522787731+7732.tif	3

```

In [9]: df.to_csv('train.csv')

```

Output screen-8.3 Data loading and cleaning

## 8.3 Data Preprocessing

```
In [10]: for image in examples:
          directory = os.path.dirname(image)
          doctype = directory.rsplit('\\', 1)[-1]
          img = cv2.imread(image, 0)

          equ = cv2.equalizeHist(img)
          res = np.hstack((img,equ))

          plt.subplot(111)
          plt.imshow(res, cmap='Greys_r')
          plt.title('{} '.format(doctype))
          plt.xticks([])
          plt.yticks([])

          plt.show()

#np.histogram(1d array of img values, 256 bins between range 0 and 255, inclusive), returns histogram values and bin edges
```



Output Screen-8.4 Data Preprocessing

## 8.4 Evaluation of model

```
[ ] cnn.eval() # Change model to 'eval' mode (BN uses moving mean/var).
    correct = 0
    total = 0
    im_labels = []
    im_preds = []
    epoch=20
    for i, (images1, labels1) in enumerate(test_iter1):
        images1 = Variable(images1).cuda()
        outputs = cnn(images1)
        _, predicted = torch.max(outputs.data, 1)
        total += labels1.size(0)
        correct += (predicted.cpu() == labels1.cpu()).sum()

        if (i) % 10 == 0:
            print('Epoch {}/ {}, Iter {}/ {}'.format(epoch + 1, num_epochs, i, (test_loader1.data_len / batch_size)))

        im_labels.append(labels1.cpu().numpy())
        im_preds.append(predicted.cpu().numpy())

    # -----
    print('test Accuracy of the model on the test images: {}'.format(100 * correct / total))
```

```
Epoch 21/30, Iter 0/187.5
Epoch 21/30, Iter 10/187.5
Epoch 21/30, Iter 20/187.5
Epoch 21/30, Iter 30/187.5
Epoch 21/30, Iter 40/187.5
Epoch 21/30, Iter 50/187.5
Epoch 21/30, Iter 60/187.5
Epoch 21/30, Iter 70/187.5
Epoch 21/30, Iter 80/187.5
Epoch 21/30, Iter 90/187.5
Epoch 21/30, Iter 100/187.5
Epoch 21/30, Iter 110/187.5
Epoch 21/30, Iter 120/187.5
Epoch 21/30, Iter 130/187.5
Epoch 21/30, Iter 140/187.5
Epoch 21/30, Iter 150/187.5
Epoch 21/30, Iter 160/187.5
Epoch 21/30, Iter 170/187.5
Epoch 21/30, Iter 180/187.5
test Accuracy of the model on the test images: 71
```

### Output Screen 8.5 Evaluation of model

```
[26] cnn.eval() # Change model to 'eval' mode (BN uses moving mean/var).
      correct2 = 0
      total2 = 0
      epoch=0
      im_labels2 = []
      im_preds2 = []

      for i, (images, labels) in enumerate(val_iter):
          images = Variable(images).cuda()
          outputs = cnn(images)
          _, predicted = torch.max(outputs.data, 1)
          total2 += labels.size(0)
          correct2 += (predicted.cpu() == labels).sum()

          if (i) % 10 == 0:
              print('Epoch {}/ {}, Iter {}/ {}'.format(epoch + 1, num_epochs, i, (val_loader.data_len / batch_size)))

          im_labels2.append(labels.cpu().numpy())
          im_preds2.append(predicted.cpu().numpy())

      # -----
      print('val Accuracy of the model on the test images: {}'.format(100 * correct2 / total2))
```

```
Epoch 1/30, Iter 0/187.5
Epoch 1/30, Iter 10/187.5
Epoch 1/30, Iter 20/187.5
Epoch 1/30, Iter 30/187.5
Epoch 1/30, Iter 40/187.5
Epoch 1/30, Iter 50/187.5
Epoch 1/30, Iter 60/187.5
Epoch 1/30, Iter 70/187.5
Epoch 1/30, Iter 80/187.5
Epoch 1/30, Iter 90/187.5
Epoch 1/30, Iter 100/187.5
Epoch 1/30, Iter 110/187.5
Epoch 1/30, Iter 120/187.5
Epoch 1/30, Iter 130/187.5
Epoch 1/30, Iter 140/187.5
Epoch 1/30, Iter 150/187.5
Epoch 1/30, Iter 160/187.5
Epoch 1/30, Iter 170/187.5
Epoch 1/30, Iter 180/187.5
val Accuracy of the model on the test images: 70
```

### Output Screen 8.6 validation of model



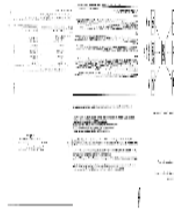
## 8.5 Prediction

```
[ ] _, preds = torch.max(outputs.data, 1)
    predicted_labels = [preds[j] for j in range(images.size()[0])]
```

```
[ ] print("Prediction:")
    show_databatch(images.data.cpu(), predicted_labels)
```

☞ Prediction:

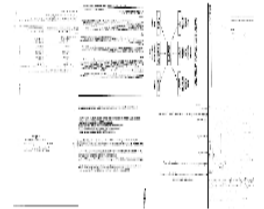
['advertisement', 'specification', 'news article', 'resume', 'resume', 'scientific publication', 'letter', 'news article', 'advertisement', 'advertisement', 'form', 'letter', 'budget', 'memo', 'scientific report', 'email',



```
[ ] print("Ground truth:")
    show_databatch(images.data.cpu(), labels.data.cpu())
```

☞ Ground truth:

['advertisement', 'specification', 'invoice', 'questionnaire', 'resume', 'news article', 'letter', 'news article', 'advertisement', 'advertisement', 'form', 'letter', 'budget', 'memo', 'scientific report', 'email', 'form',



### Output Screen 8.7 prediction

## 8.6 Data preparation for keras vgg

```
In [70]: inv=os.listdir('./rv1-cdip/subsets/tr1/invoice')
#uninfected=os.listdir('')

for a in inv:
    try:
        image=cv2.imread("./rv1-cdip/subsets/tr1/invoice/"+a)
        data2.append(np.array(image))
        labels.append(11)
    except AttribubuteError:
        print("")
```

```
In [71]: tr=np.array(data2)
tr.shape
```

```
Out[71]: (12000, 224, 224, 3)
```

```
In [72]: pre=os.listdir('./rv1-cdip/subsets/tr1/presentation')
#uninfected=os.listdir('')

for a in pre:
    try:
        image=cv2.imread("./rv1-cdip/subsets/tr1/presentation/"+a)
        data2.append(np.array(image))
        labels.append(12)
    except AttribubuteError:
        print("")
```

```
In [73]: tr=np.array(data2)
tr.shape
```

```
Out[73]: (13000, 224, 224, 3)
```

Activate V  
Go to Setting:

### Output Screen 8.8 Data preparation for keras vgg

## 8.7 Accuracy in keras vgg16

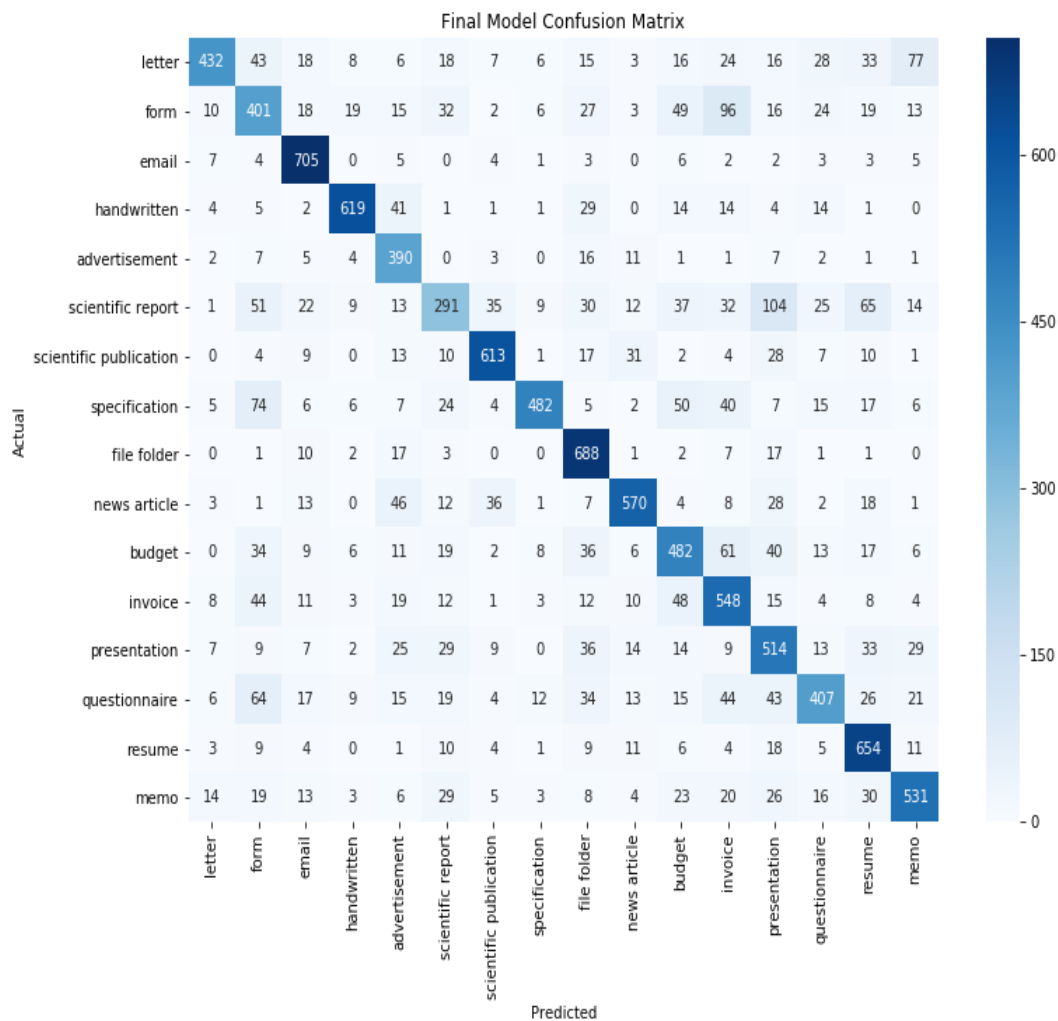
```

Epoch 30/50
9600/9600 [=====] - 122s 13ms/step - loss: 0.0796 - acc: 0.9819 - val_loss: 1.7976 - val_acc: 0.7275
Epoch 31/50
9600/9600 [=====] - 121s 13ms/step - loss: 0.0674 - acc: 0.9852 - val_loss: 1.7716 - val_acc: 0.7319
Epoch 32/50
9600/9600 [=====] - 121s 13ms/step - loss: 0.0698 - acc: 0.9832 - val_loss: 1.9298 - val_acc: 0.7206
Epoch 33/50
9600/9600 [=====] - 121s 13ms/step - loss: 0.0898 - acc: 0.9770 - val_loss: 1.8617 - val_acc: 0.7344
Epoch 34/50
9600/9600 [=====] - 121s 13ms/step - loss: 0.0840 - acc: 0.9785 - val_loss: 1.8427 - val_acc: 0.7219
Epoch 35/50
9600/9600 [=====] - 121s 13ms/step - loss: 0.1139 - acc: 0.9740 - val_loss: 1.8521 - val_acc: 0.7175
Epoch 36/50
9600/9600 [=====] - 121s 13ms/step - loss: 0.1720 - acc: 0.9595 - val_loss: 1.8399 - val_acc: 0.7191
Epoch 37/50
9600/9600 [=====] - 121s 13ms/step - loss: 0.0958 - acc: 0.9771 - val_loss: 1.9123 - val_acc: 0.7203
Epoch 38/50
9600/9600 [=====] - 121s 13ms/step - loss: 0.1221 - acc: 0.9715 - val_loss: 1.9260 - val_acc: 0.7253
Epoch 39/50
9600/9600 [=====] - 121s 13ms/step - loss: 0.0818 - acc: 0.9803 - val_loss: 1.7006 - val_acc: 0.7234
Epoch 40/50
9600/9600 [=====] - 121s 13ms/step - loss: 0.0741 - acc: 0.9817 - val_loss: 1.8019 - val_acc: 0.7100
Epoch 41/50
9600/9600 [=====] - 121s 13ms/step - loss: 0.0798 - acc: 0.9801 - val_loss: 1.9302 - val_acc: 0.7222
Epoch 42/50
9600/9600 [=====] - 121s 13ms/step - loss: 0.0751 - acc: 0.9821 - val_loss: 1.9086 - val_acc: 0.7288
Epoch 43/50
9600/9600 [=====] - 121s 13ms/step - loss: 0.0490 - acc: 0.9893 - val_loss: 2.2207 - val_acc: 0.7247
Epoch 44/50
9600/9600 [=====] - 121s 13ms/step - loss: 0.0778 - acc: 0.9820 - val_loss: 2.0869 - val_acc: 0.7131
Epoch 45/50
9600/9600 [=====] - 121s 13ms/step - loss: 0.1003 - acc: 0.9777 - val_loss: 1.8177 - val_acc: 0.7212
Epoch 46/50
9600/9600 [=====] - 121s 13ms/step - loss: 0.0657 - acc: 0.9832 - val_loss: 2.0480 - val_acc: 0.7194
Epoch 47/50
9600/9600 [=====] - 121s 13ms/step - loss: 0.0622 - acc: 0.9847 - val_loss: 1.9572 - val_acc: 0.7125
Epoch 48/50
9600/9600 [=====] - 121s 13ms/step - loss: 0.0762 - acc: 0.9823 - val_loss: 2.1156 - val_acc: 0.7222
Epoch 49/50
9600/9600 [=====] - 121s 13ms/step - loss: 0.0535 - acc: 0.9880 - val_loss: 2.1211 - val_acc: 0.7231
Epoch 50/50
9600/9600 [=====] - 121s 13ms/step - loss: 0.0509 - acc: 0.9889 - val_loss: 2.1686 - val_acc: 0.7256
Test loss: 2.0029831072688102
Test accuracy: 0.7284375

```

## Output Screen 8.8 accuracy in keras vgg16

## 8.8 Confusion matrix



Output Screen 8.9 Confusion matrix

## 8.9 Classification Report

	Precision	recall	f1-score	support
<b>letter</b>	0.81	0.64	0.72	750
<b>form</b>	0.57	0.6	0.58	750
<b>email</b>	0.86	0.96	0.91	750
<b>handwritten</b>	0.97	0.71	0.82	750
<b>advertisement</b>	0.7	0.83	0.76	750
<b>scientific report</b>	0.51	0.43	0.47	750
<b>scientific publication</b>	0.76	0.86	0.81	750
<b>specification</b>	0.71	0.81	0.76	750
<b>file folder</b>	0.83	0.86	0.85	750
<b>news article</b>	0.76	0.78	0.77	750
<b>budget</b>	0.67	0.6	0.63	750
<b>invoice</b>	0.61	0.74	0.67	750
<b>presentation</b>	0.57	0.68	0.62	750
<b>questionnaire</b>	0.59	0.62	0.6	750
<b>resume</b>	0.88	0.84	0.86	750
<b>memo</b>	0.88	0.61	0.72	750
<b>micro average</b>	0.72	0.72	0.72	11700
<b>macro average</b>	0.73	0.72	0.72	11700
<b>weighted average</b>	0.73	0.72	0.72	11700

*Table-1 Classification Report for Final Model*

## CHAPTER-9

### CONCLUSION

In general, the convolutional neural network proved to be an effective classifier of documents but had varied results on different classes. Some classes, including emails, resumes, file folders, and scientific publications predicted very well, with F1-scores of 0.91, 0.86, 0.85, and 0.81, respectively, while others, including scientific reports, forms, questionnaires, and presentations, predicted relatively poorly, with F1-scores of 0.47, 0.58, 0.60, and 0.62, respectively.

Document images are more difficult to identify than natural images. Whereas most natural images attempt to locate a specific object within the overall image, documents require the analysis of multiple elements of each image for classification. Elements in document headers, footers, and bodies all factor into correct classification. In addition, documents contain small features that require input sizes to be kept relatively large, requiring more parameters to be trained and a longer training time.

High intraclass variance also exists in all of the class and makes identification even more difficult. As can be seen below in the two worst performing classes (scientific report and form), documents vary considerably within each class, though variance is greater in some classes over others. Classes with more strict document structures, such as emails and resumes, predicted better than classes with varying structure.

#### **9.1 Future improvements**

Several improvements can be made to this project. Classes could be merged to make fewer, more homogeneous classes to reduce the size of the dataset need for training and reduce intraclass variance. Regions of each image could be explored in separate models and combined using another classifier such as an SVM. Lastly, a top-3 or top-5 classifier could be developed rather than a top-1 classifier to provide more clarity into how the CNN model is misclassifying some of the worse predicting classes.

## CHAPTER-10

### REFERENCES

- 1.Amir Jafari. “Amir-Jafari/Deep-Learning.” GitHub, 18 Aug. 2018, [github.com/amir-jafari/Deep-Learning/tree/master/Pytorch\\_/6-Conv\\_Mnist](https://github.com/amir-jafari/Deep-Learning/tree/master/Pytorch_/6-Conv_Mnist).
- 2.“API Reference¶.” 1.4. Support Vector Machines - Scikit-Learn 0.19.2 Documentation, [scikit-learn.org/stable/modules/classes.html#module-sklearn.metrics](https://scikit-learn.org/stable/modules/classes.html#module-sklearn.metrics).
- 3.Brits, Denny. “Understanding Convolutional Neural Networks for NLP.” WildML, 10 Jan. 2016.
- 4.Budhiraja, Amar. “Learning Less to Learn Better-Dropout in (Deep) Machine Learning.” Medium.com,Medium, 15 Dec. 2016, [medium.com/@amarbudhiraja/https-medium-com-amarbudhiraja-learning-less-to-learn-better-dropout-in-deep-machine-learning-74334da4bfc5](https://medium.com/@amarbudhiraja/https-medium-com-amarbudhiraja-learning-less-to-learn-better-dropout-in-deep-machine-learning-74334da4bfc5).
- 5.CascadeClassification-OpenCVDDocumentation, [docs.opencv.org/3.1.0/d5/daf/tutorial\\_py\\_histogram\\_equalization.html](https://docs.opencv.org/3.1.0/d5/daf/tutorial_py_histogram_equalization.html).
- 6.COSTE, Arthur. “Project 1 : Histograms.” Summation of Twitches and Tetanization, Sept.2012, [www.sci.utah.edu/~acoste/uou/Image/project1/Arthur\\_COSTE\\_Project\\_1\\_report.html](http://www.sci.utah.edu/~acoste/uou/Image/project1/Arthur_COSTE_Project_1_report.html).
- 7.CS231n Convolutional Neural Networks for Visual Recognition, [cs231n.github.io/convolutional-networks/](https://cs231n.github.io/convolutional-networks/).
- 8.“PyTorch Documentation.” PyTorch, [pytorch.org/docs/stable/index.html](https://pytorch.org/docs/stable/index.html).
- 9.Tensmeyer, Chris, and Tony Martinez. "Analysis of Convolutional Neural Networks for Document Image Classification.”
- 10.”The RVL-CDIP Data Set”, [www.cs.cmu.edu/~aharley/rvl-cdip/](http://www.cs.cmu.edu/~aharley/rvl-cdip/)