International Journal of Multidisciplinary Engineering in Current Research - IJMEC

Volume 9, Issue 9, September-2024, http://ijmec.com/, ISSN: 2456-4265

MEDICAL RECOGNIZANCE FOR THE VISUALLY IMPAIRED USING CONVOLUTIONAL NEURAL NETWORKS

Naziya[*1]**, Dr. Md. Ateeq Ur Rahman** [*2]**, Dr. Jothikumar. R**[*3]

[*1] PG Scholar - CSE, Dept. of Computer Science Engineering,
Shadan College of Engineering & Technology
naziyajaffer007@gmail.com

[*2] Professor, Dept. of Computer Science Engineering, Shadan College of Engineering & Technology
mail_to_ateeq@yahoo.com.

[*3] Professor, Dept. of Computer Science Engineering, Shadan College of Engineering & Technology
drjothikumarr@gmail.com.

Abstract: *Providing blind persons with adequate access to their surroundings is a pressing issue that has to be addressed. There are a number of challenges that modern assistive technology poses, which restrict the activities that persons who are visually impaired are able to participate in on a daily basis. These specific aids do not live up to the expectations of the customers, and owing to the exorbitant price tags, they are no longer within the financial grasp of some segments of the population. As part of this project, we provide a solution that makes use of smart glasses to assist visually impaired persons in identifying drugs. Our goal is to address some of the limitations that are associated with the visual aids that are now available. It is possible to read the name of the medication by using the smart glass technology, which operates by sending acoustic impulses via the headphones. Defining the name of the medicine is accomplished by the smart glass system via the use of a convolutional neural network.*

Index Terms—*Convolutional neural network, LeNet, Alex Net, Image processing techniques and python.*

1. INTRODUCTION

VARIOUS statistics shows an increase of visually impaired people are in higher. Worldwide around 285 million people are estimated to be visually impaired as reported by WHO. The development of the assistive technology is not affordable to the all the section of the society and cost effective devices are provided with single or the limited function. This inadequacy of the assistive aids has limited the accommodation of visually impaired in schools and other jobs making their life more dispirited. Wearable devices for helping blind people are found to be the most effective because they require minimum use of hands.

1.1. SCOPE

The project is used for the Blind people to recognize the medicine

1.2.PURPOSE OF PROJECT

Giving blind people with great accessibility to their environment is of great demand. People with visual impairments experience a lot of problem in using the modern assistive device that limits their daily basic activities

1.3.KEYWORDS

Convolutional neural network, LeNet, Alex Net

2. OBJECTIVE OF THE PROJECT

A. Existing System:

The existing system was with only the bar code scanner sound censor mechanism That was not up to mark
And the prediction was not accurate

Disadvantages:

1. It can't identify every product
2. Hard to Code the system

B. Proposed system:

The objects present in an image or in a video sequence are detected and identified using the Object recognition techniques like Convolutional neural network, LeNet, Alex Net

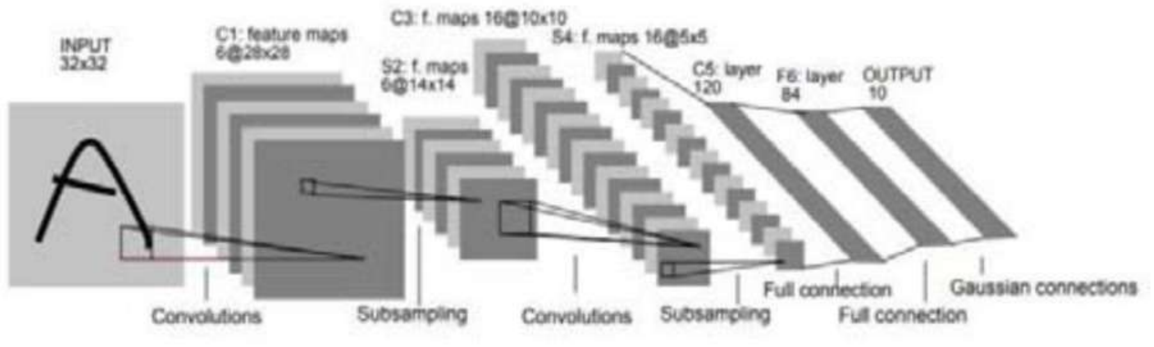
Advantages:

In the proposed system Wee have using Deep learning leaning algorithm namely CNN
Building CNN was quite simple and Fast
The algorithm gives Accurate predictions

C. Modules Description

We have used almost all of the same libraries which are used in normal ML/DL problems like pandas, numpy, matplotlib, sklearn, Computer Vision,Keras Stream, Detection etc.

3.Project Architecture



4. Literature survey

Real Time Image Processing on Single Board Computer

This paper focus on real time image processing. Industrial image processing is a very important branch of scientific image processing and it is because of the rapid development of computer industry production and computer intelligence, as well as the corresponding developments in computer-aided image analysis . Image edge detection on single board computer is discussed in this paper. The application (GUI) Graphical User Interface was designed using Qt and Linux gcc Integrated Development Environment (IDE) for implementing image processing algorithm using Open Source Computer Vision Library (OpeCV). This developed software integrated in mobiles by the cross compilation of Qt and the OpeCV software for Linux Operating system.

Learning to localize objects with structured output regression

Sliding window classifiers are among the most successful and widely applied techniques for object localization. However, training is typically done in a way that is not specific to the localization task. First a binary classifier is trained using a sample of positive and negative examples, and this classifier is subsequently applied to multiple regions within test images. We propose instead to treat object localization in a principled way by posing it as a problem of predicting structured data: we model the problem not as binary classification, but as the prediction of the bounding box of objects located in images. The use of a joint-kernel framework allows us to formulate the training procedure as a generalization of an SVM, which can be solved efficiently. We further improve computational efficiency by using a branch-and-bound strategy for localization during both training and testing. Experimental evaluation on the PASCAL VOC and TU Darmstadt datasets show that the structured training procedure improves performance over binary training as well as the best previously published scores.

Body part detectors trained using 3D human pose annotations

We address the classic problems of detection, segmentation and pose estimation of people in images with a novel definition of a part, a poselet. We postulate two criteria . It should be easy to find a poselet given an input image it should be easy to localize the 3D configuration of the person conditioned on the detection of a poselet. To permit this we have built a new dataset, H3D, of annotations of humans in 2D photographs with 3D joint information, inferred using anthropometric constraints. This enables us to implement a data-driven search procedure for finding poselets that are tightly clustered in both 3D joint configuration space as well as 2D image appearance. The algorithm discovers poselets that correspond to frontal and profile faces, pedestrians, head and shoulder views, among others.

Automatically Structured Neural Networks for Handwritten Character and Word Recognition

Highly structured neural networks like the Time-Delay Neural Network (TDNN) can achieve very high recognition accuracies in real world applications like on-line handwritten character and speech recognition systems. Achieving the best possible performance greatly depends on the optimization of all structural parameters for the given task and amount of training data. We propose an Automatic Structure Optimization (ASO) algorithm that avoids time-consuming manual optimization and apply it to Multi State Time-Delay Neural Networks (MSTDNNs), a recent extension of the TDNN. We show that MSTDNNs are a very powerful approach to on-line handwritten character and word recognition and that the ASO algorithm can automatically structure this type of architecture efficiently in a single training run.

Multi-column deep neural networks for image classification

Traditional methods of computer vision and machine learning cannot match human performance on tasks such as the recognition of handwritten digits or traffic signs. Our biologically plausible, wide and deep artificial neural network architectures can. Small (often minimal) receptive fields of convolutional winner-take-all neurons yield large network depth, resulting in roughly as many sparsely connected neural layers as found in mammals between retina and visual cortex. Only winner neurons are trained. Several deep neural columns become experts on

DOI: https://doi-ds.org/doilink/09.2024-43831256

inputs preprocessed in different ways; their predictions are averaged. Graphics cards allow for fast training. On the very competitive MNIST handwriting benchmark, our method is the first to achieve near-human performance. On a traffic sign recognition benchmark it outperforms humans by a factor of two. We also improve the state-of-the-art on a plethora of common image classification benchmarks.

Development of an augmented reality vehicle for driver performance evaluation

Observing drivers' behaviors by reproducing traffic accidents and conflict situations is important for developing advanced driver assistance systems. For this purpose, driving simulators are frequently used to evaluate the effectiveness of driver assistance systems during product development. However, motion (simulator) sickness can be a serious practical problem with driving simulators. Therefore, an instrumented vehicle, the JARI-ARV (Japan Automobile Research Institute-Augmented Reality Vehicle), was developed to reproduce realistic traffic accident and conflict scenarios without endangering the driver. In this study, we examined level of control in the following cases: a right turn and encounter with a pedestrian by comparing the JARI-ARV with a standard (unaltered) same model vehicle. Results of the experiment indicated that drivers tend to react to virtual traffic participants in the same way as driving a standard vehicle. The study indicates that the JARI-ARV can play a useful role in human factors research.

5. SYSTEM DESIGN

a.UML DIAGRAMS

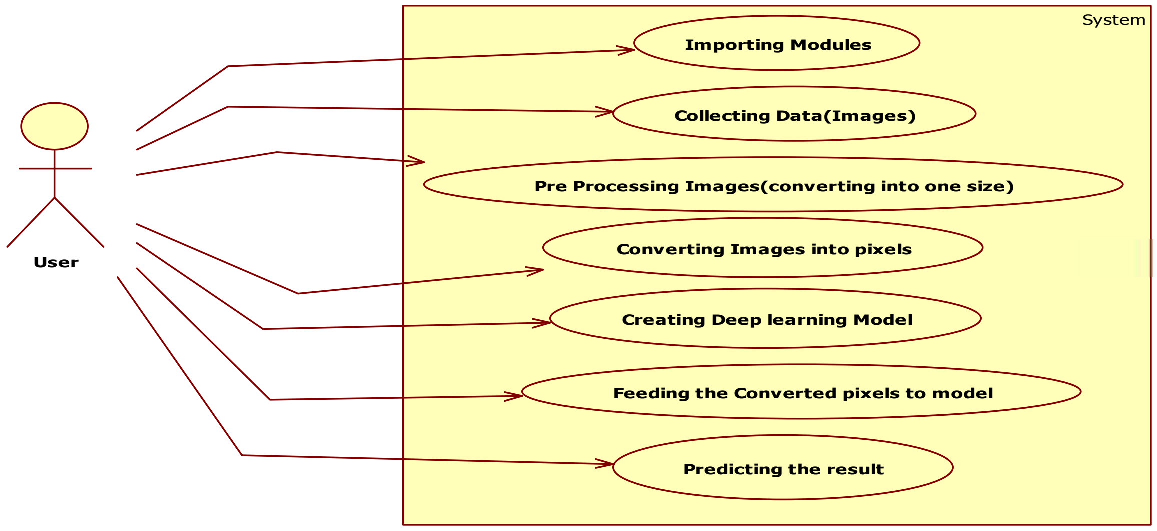
UML DIAGRAMS

The Unified Modeling Language (UML) is used to specify, visualize, modify, construct and document the artifacts of an object-oriented software intensive system under development. UML offers a standard way to visualize a system's architectural blueprints, including elements such as:

- actors
- business processes
- (logical) components
- activities
- programming language statements
- database schemas, and
- Reusable software components.

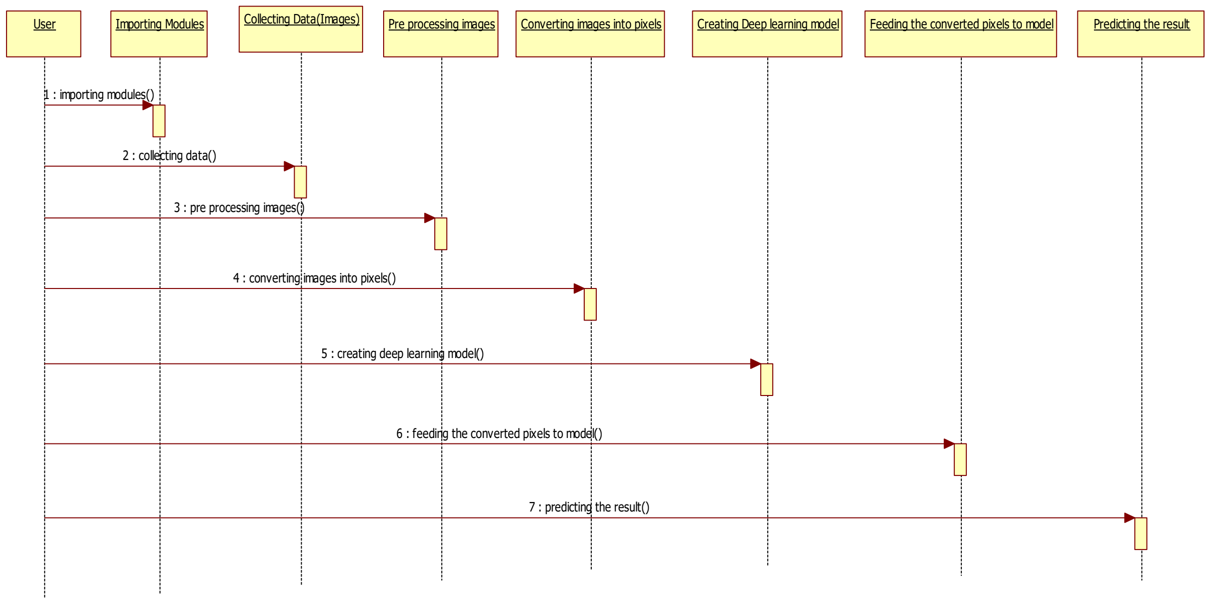
UML combines best techniques from data modeling (entity relationship diagrams), business modeling (work flows), object modeling, and component modeling. It can be used with all processes, throughout the software development life cycle, and across different implementation technologies. UML has synthesized the notations of the Booch method, the Object-modeling technique (OMT) and Object-oriented software engineering (OOSE) by fusing them into a single, common and widely usable modeling language. UML aims to be a standard modeling language which can model concurrent and distributed systems.

Use Case Diagram:

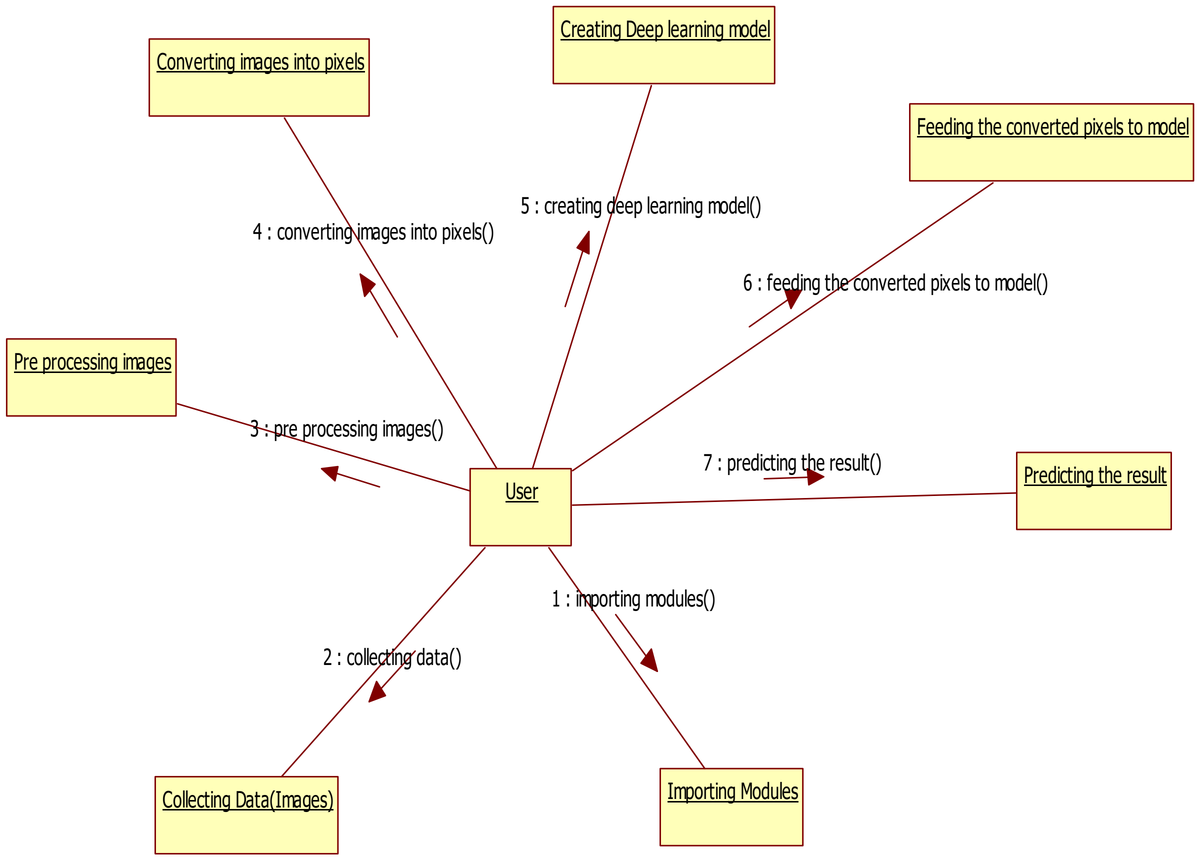

System
User
Importing Modules
Collecting Data(Images)
Pre Processing Images(converting into one size)
Converting Images into pixels
Creating Deep learning Model
Feeding the Converted pixels to model
Predicting the result

Sequence diagram:

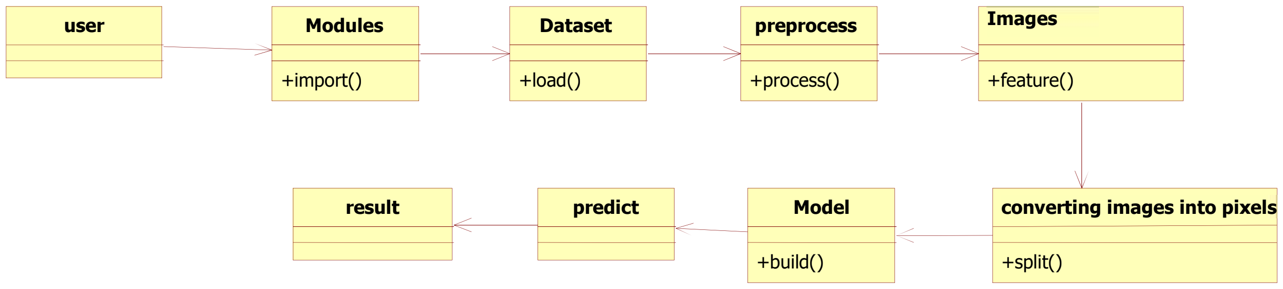
Sequence Diagrams Represent the objects participating the interaction horizontally and time vertically. A Use Case is a kind of behavioral classifier that represents a declaration of an offered behavior. Each use case specifies some behavior, possibly including variants that the subject can perform in collaboration with one or more actors. Use cases define the offered behavior of the subject without reference to its internal structure. These behaviors, involving interactions between the actor and the subject, may result in changes to the state of the subject and communications with its environment. A use case can include possible variations of its basic behavior, including exceptional behavior and error handling.


User
Importing Modules
Collecting Data(Images)
Pre processing images
Converting images into pixels
Creating Deep learning model
Feeding the converted pixels to model
Predicting the result
1 : importing modules()
2 : collecting data()
3 : pre processing images()
4 : converting images into pixels()
5 : creating deep learning model()
6 : feeding the converted pixels to model()
7 : predicting the result()

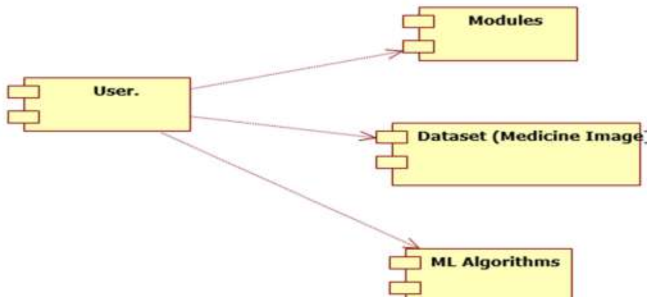
Collaboration Diagram:


Converting images into pixels
Creating Deep learning model
Feeding the converted pixels to model
5 : creating deep learning model()
4 : converting images into pixels()
6 : feeding the converted pixels to model()
Pre processing images
3 : pre processing images()
User
7 : predicting the result()
Predicting the result
1 : importing modules()
2 : collecting data()
Collecting Data(Images)
Importing Modules

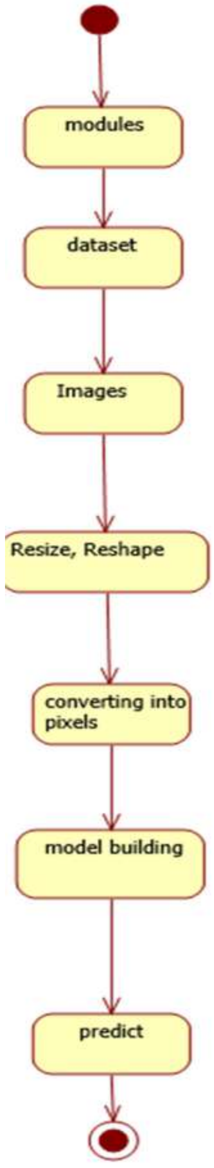
Class Diagram:



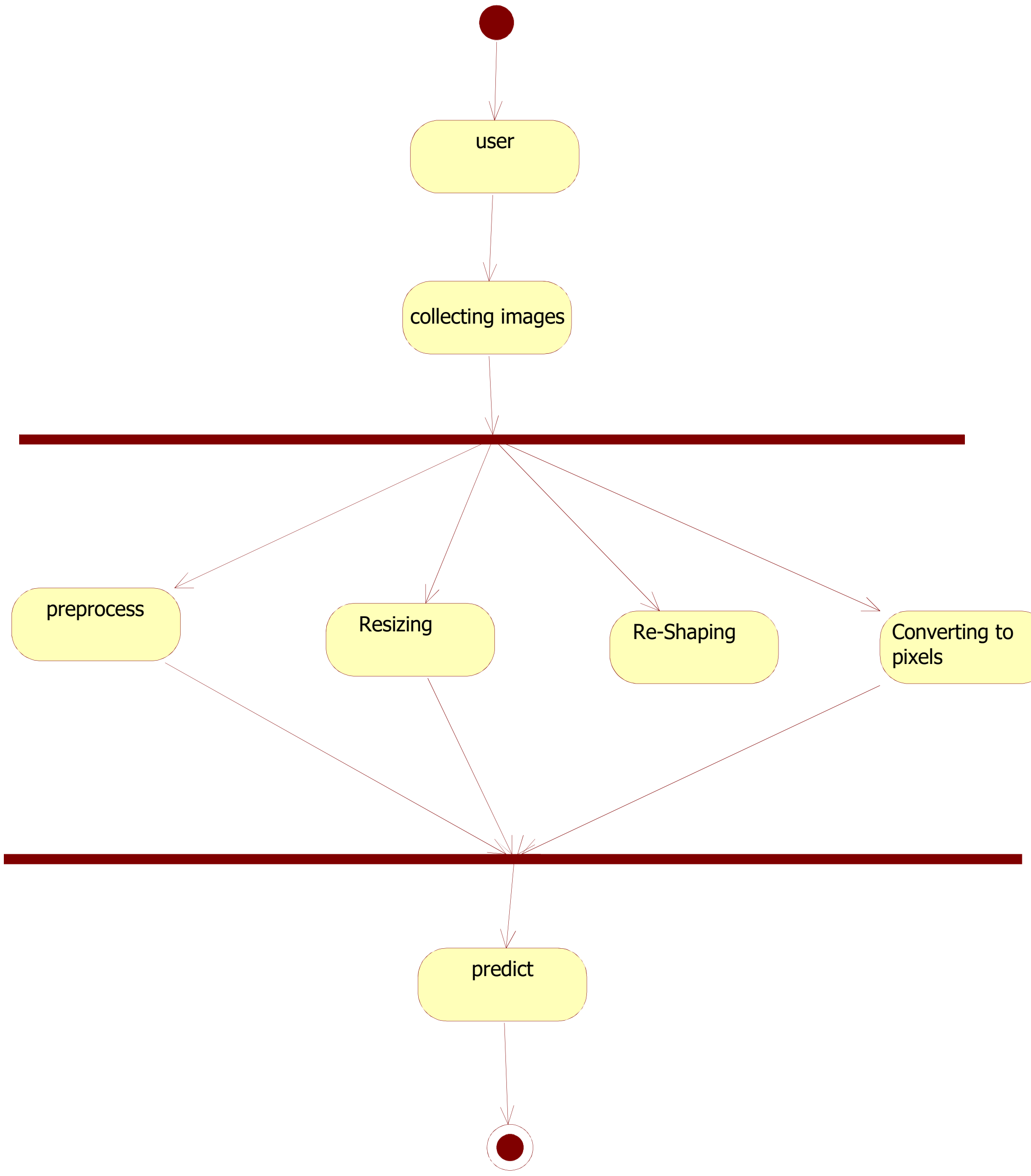
Component Diagram



State Diagram:



Activity Diagram



6. TECHNOLOGY DESCRIPTION AND IMPLEMENTATION

IntroductionToPython Framework : Introduction to Django This book is about Django, a Web development framework that saves you time and makes Web development a joy. Using Django, you can build and maintain high-quality Web applications with minimal fuss. At its best, Web development is an exciting, creative act; at its worst, it can be a repetitive, frustrating nuisance. Django lets you focus on the fun stuff — the crux of your Web application — while easing the pain of the repetitive bits. In doing so, it provides high-level abstractions of common Web development patterns, shortcuts for frequent programming tasks, and clear conventions for how to solve problems. At the same time, Django tries to stay out of your way, letting you work outside the scope of the framework as needed. The goal of this book is to make you a Django expert. The focus is twofold. First, we explain, in depth, what Django does and how to build Web applications with it. Second, we discuss higher-level concepts where appropriate, answering the question “How can I apply these tools effectively in my own projects?” By reading this book, you'll learn the skills needed to develop powerful Web sites quickly, with code that is clean and easy to maintain.

What Is a Web Framework?

Django is a prominent member of a new generation of Web frameworks. So what exactly does that term mean? To answer that question, let's consider the design of a Web application written using the Common Gateway Interface (CGI) standard, a popular way to write Web applications circa 1998. In those days, when you wrote a CGI application, you did everything yourself — the equivalent of baking a cake from scratch. For example, here's a simple CGI script, written in Python, that displays the ten most recently published books from a database:

This code is straightforward. First, it prints a “Content-Type” line, followed by a blank line, as required by CGI. It prints some introductory HTML, connects to a database and executes a query that retrieves the latest ten books. Looping over those books, it generates an HTML unordered list. Finally, it prints the closing HTML and closes the database connection.

With a one-off dynamic page such as this one, the write-it-from-scratch approach isn't necessarily bad. For one thing, this code is simple to comprehend — even a novice developer can read these 16 lines of Python and understand all it does, from start to finish. There's nothing else to learn; no other code to read. It's also simple to deploy: just save this code in a file called latestbooks.cgi, upload that file to a Web server, and visit that page with a browser. But as a Web application grows beyond the trivial, this approach breaks down, and you face a number of problems:

DOI: https://doi-ds.org/doilink/09.2024-43831256

Test Cases:

SNO	Test case Title	Pre-requisites	Action	Expected Result	Test Result(pass/fail)
Test case 1	Software requirements	Python version 3.6.5	python --version (Checking the version)	Python 3.6.5 present in your system	Pass
Test case 2	Idle requirements	Jupyter notebook	CMD--(jupyter notebook)	jupyter file should run on the local host	Pass
Test case 3	packages need	pandas, numpy,sklearn,matplotlib,seaborn	ls(list of packages)	all packages should import	Pass
Test case 4	Set the path for image directory	Path was correct	Check the classes	Images are visisble	Pass
Test case 5	Prediction	Model should be trained with training dataset	Use trained model on test Images	It will display the predicted results	Pass

7. INPUT AND OUTPUT DESIGN

The following some are the projects inputs and outputs.

Inputs:

- Importing the all required packages like numpy, pandas, matplotlib, scikit – learn and required machine learning algorithms packages.
- Setting the dimensions of visualization graph.
- Downloading and importing the dataset and convert to data frame.

Outputs:

- Preprocessing the importing data frame for imputing nulls with the related information.
- All are displaying cleaned outputs.
- After applying machine learning algorithms it will give good results and visualization plots.

INPUT DESIGN

Input design is a part of overall system design. The main objective during the input design is as given below:

- To produce a cost-effective method of input.
- To achieve the highest possible level of accuracy.
- To ensure that the input is acceptable and understood by the user.

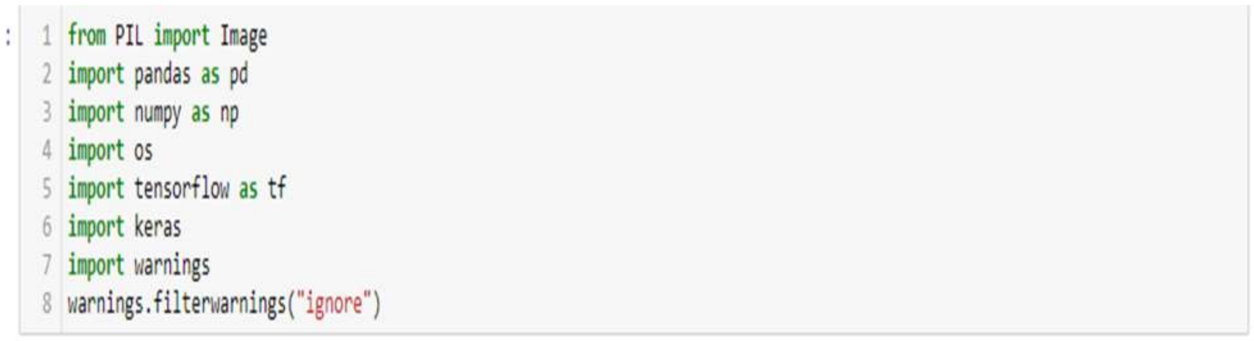
OUTPUT DESIGN

Outputs from computer systems are required primarily to communicate the results of processing to users. They are also used to provide a permanent copy of the results for later consultation. The various types of outputs in general are:

- External Outputs, whose destination is outside the organization,
- Internal Outputs whose destination is within organization and they are the
- User's main interface with the computer.
- Operational outputs whose use is purely within the computer department.
- Interface outputs, which involve the user in communicating directly with

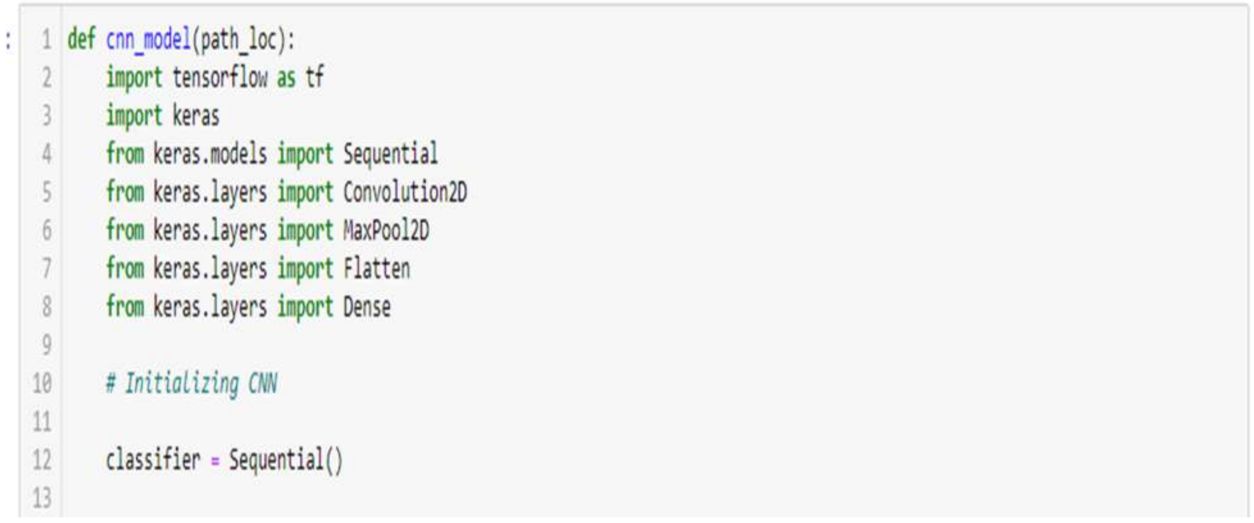
The outputs were needed to be generated as a hard copy and as well as queries to be viewed on the screen. Keeping in view these outputs, the format for the output is taken from the outputs, which are currently being obtained after manual processing. The standard printer is to be used as output media for hard copies.

8. OUTPUT SCREENS

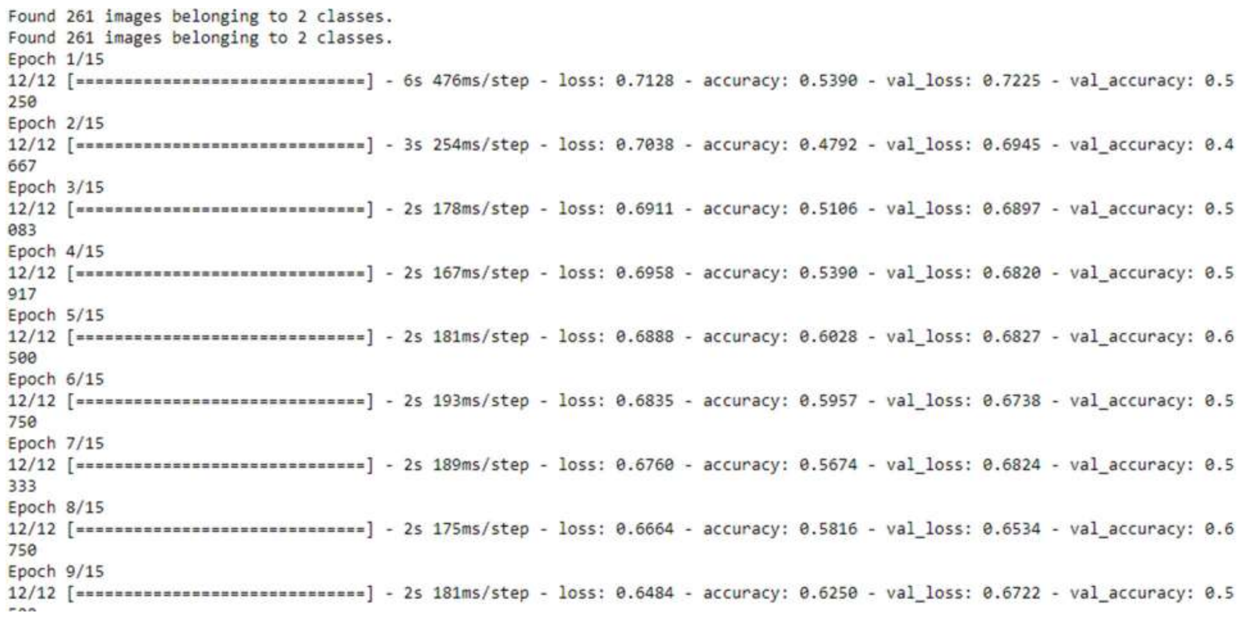


```
from PIL import Image
import pandas as pd
import numpy as np
import os
import tensorflow as tf
import keras
import warnings
warnings.filterwarnings("ignore")
```

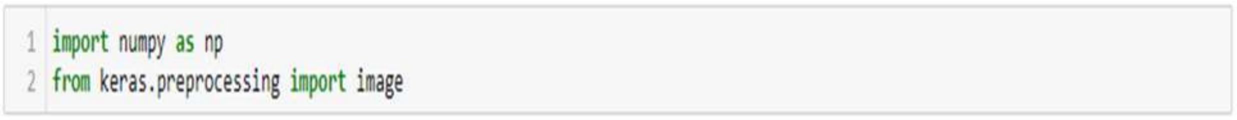
```
import keras
```



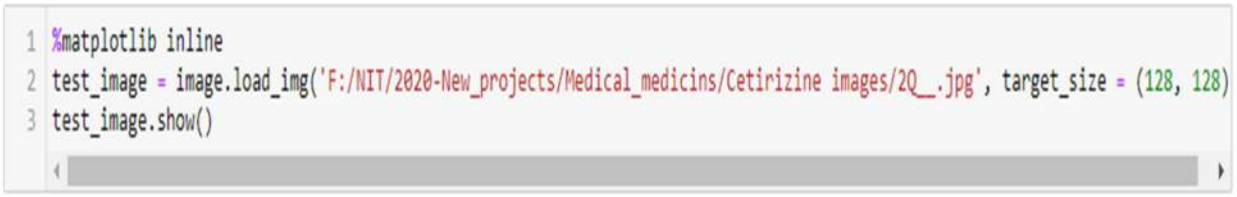
```
def cnn_model(path_loc):
    import tensorflow as tf
    import keras
    from keras.models import Sequential
    from keras.layers import Convolution2D
    from keras.layers import MaxPool2D
    from keras.layers import Flatten
    from keras.layers import Dense

    # Initializing CNN

    classifier = Sequential()

```



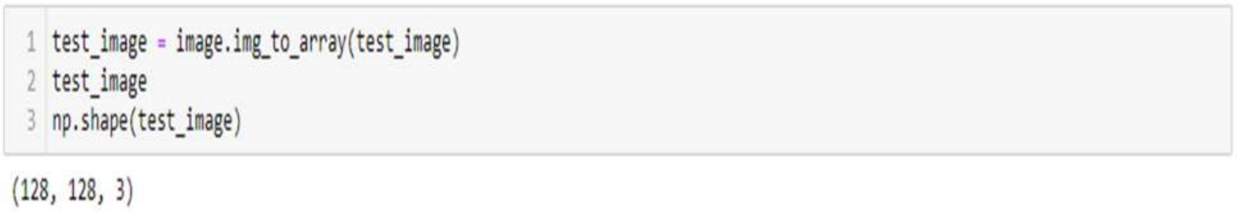
```
Found 261 images belonging to 2 classes.
Found 261 images belonging to 2 classes.
Epoch 1/15
12/12 [==============================] - 6s 476ms/step - loss: 0.7128 - accuracy: 0.5390 - val_loss: 0.7225 - val_accuracy: 0.5
250
Epoch 2/15
12/12 [==============================] - 3s 254ms/step - loss: 0.7038 - accuracy: 0.4792 - val_loss: 0.6945 - val_accuracy: 0.4
667
Epoch 3/15
12/12 [==============================] - 2s 178ms/step - loss: 0.6911 - accuracy: 0.5106 - val_loss: 0.6897 - val_accuracy: 0.5
083
Epoch 4/15
12/12 [==============================] - 2s 167ms/step - loss: 0.6958 - accuracy: 0.5390 - val_loss: 0.6820 - val_accuracy: 0.5
917
Epoch 5/15
12/12 [==============================] - 2s 181ms/step - loss: 0.6888 - accuracy: 0.6028 - val_loss: 0.6827 - val_accuracy: 0.6
500
Epoch 6/15
12/12 [==============================] - 2s 193ms/step - loss: 0.6835 - accuracy: 0.5957 - val_loss: 0.6738 - val_accuracy: 0.5
750
Epoch 7/15
12/12 [==============================] - 2s 189ms/step - loss: 0.6760 - accuracy: 0.5674 - val_loss: 0.6824 - val_accuracy: 0.5
333
Epoch 8/15
12/12 [==============================] - 2s 175ms/step - loss: 0.6664 - accuracy: 0.5816 - val_loss: 0.6534 - val_accuracy: 0.6
750
Epoch 9/15
12/12 [==============================] - 2s 181ms/step - loss: 0.6484 - accuracy: 0.6250 - val_loss: 0.6722 - val_accuracy: 0.5
```



```python
import numpy as np
from keras.preprocessing import image
```

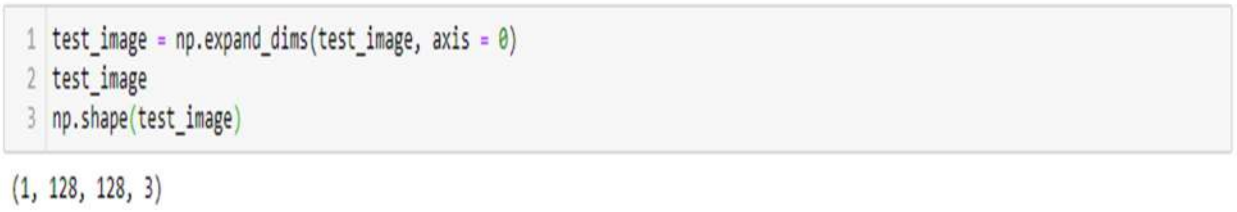


```python
%matplotlib inline
test_image = image.load_img('F:/NIT/2020-New_projects/Medical_medicins/Cetirizine images/2Q__.jpg', target_size = (128, 128)
test_image.show()
```



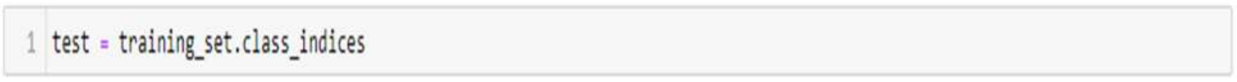
```python
test_image = image.img_to_array(test_image)
test_image
np.shape(test_image)
```

```
(128, 128, 3)
```

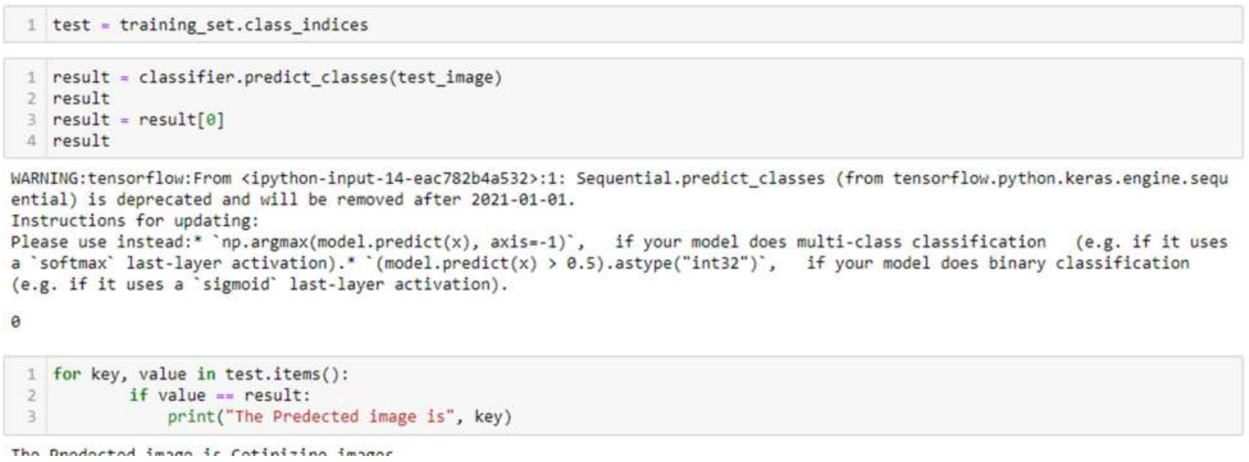


```python
test_image = np.expand_dims(test_image, axis = 0)
test_image
np.shape(test_image)
```

```
(1, 128, 128, 3)
```



```python
test = training_set.class_indices
```



```
test = training_set.class_indices
```

```
result = classifier.predict_classes(test_image)
result
result = result[0]
result
```

```
WARNING:tensorflow:From <ipython-input-14-eac782b4a532>:1: Sequential.predict_classes (from tensorflow.python.keras.engine.sequ
ential) is deprecated and will be removed after 2021-01-01.
Instructions for updating:
Please use instead:* `np.argmax(model.predict(x), axis=-1)`,   if your model does multi-class classification   (e.g. if it uses
a `softmax` last-layer activation).* `(model.predict(x) > 0.5).astype("int32")`,   if your model does binary classification
(e.g. if it uses a `sigmoid` last-layer activation).

0
```

```
for key, value in test.items():
        if value == result:
            print("The Predected image is", key)
```

```
The Predected image is Cetirizine images
```

9. FUTURE ENHANCEMENTS

Construct the model that is more accurate by using the most advanced algorithms. The user experience may be improved by developing a user interface that is more interesting to use. From the moment the model is put into operation, access to the remotely.

10. CONCLUSION

The ability to recognize items in the visual environment and to extract words from pictures are both essential but challenging visual activities. However, because to the fact that it is so complicated and the constraints of the available computer resources, it continues to be an outstanding problem. Through the use of the deep learning concept, a CNN architecture that was designed to determine the name of a drug was investigated and assessed by utilizing a dataset that included 6290 photographs. There is the capability for the Raspberry Pi that is embedded in the smart glass to include the algorithm that has been explained. Through the use of the camera that is placed in the glass, the drug is photographed, and the name of the medication is read out to those who are blind or visually impaired.

BIBLOGRAPHY

For software installation:

https://www.anaconda.com/download/

https://www.python.org/downloads/release/python-360/

References:

[1] Aditya Wagh, Kantian Dhoti , Karmic Ingle, “Real Time Image Processing on Single Board Computer”, Imperial Journal of Interdisciplinary Research, vol.2, Issue-7, 2016.

[2] M. B. Blaschko and C. H. Lampert,“Learning to localize objects with structured output regression”, Computer Vision– ECCV, pp. 2–15. Springer, 2008.

[3] L. Bourdev, J. Malik. Poselets,“Body part detectors trained using 3D human pose annotations”,proceedings in International Conference on Computer Vision (ICCV), 2009.

[4] U. Bodenhausen, S. Manke, “Automatically Structured Neural Networks for Handwritten

DOI: https://doi-ds.org/doilink/09.2024-43831256

Character and Word Recognition", pp. 956–961,1993.

[5] D. Dakopoulos, N. G. Bourbakis, "Wearable obstacle avoidance electronic travel aids for blind: A survey," IEEE Trans. Systems, Man, Cybern., vol. 40, no. 1, pp. 25-35, 2010.

[6] Dan, UeliMeier,JurgenSchmidhuber, "Multi-column deep neural networks for image classification." IEEE Conference on Computer Vision and Pattern Recognition, NY: Institute of Electrical and Electronics Engineers (IEEE), 2012