CUSTOM IMAGE CLASSIFIER

A PROJECT REPORT

submitted by

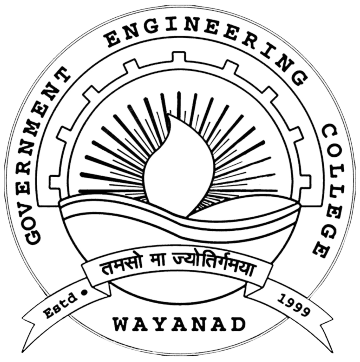
AMRUTHA K V (WYD16CS010)
JITHIN K M (WYD16CS029)
JITHIN T MATHEWS (WYD16CS030)
PRASILA P (WYD16CS045)

to

the APJ Abdul Kalam Technological University
in partial fulfilment of the requirements for the award of the Degree

of

Bachelor of Technology
in
Computer Science and Engineering



Department of Computer Science and Engineering

Government Engineering College, Wayanad
Thalappuzha – 670644

JULY, 2020

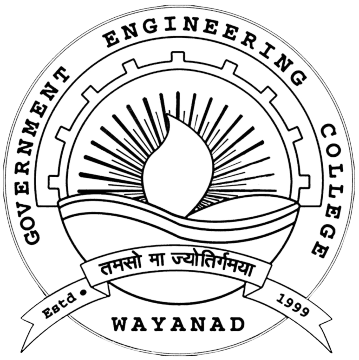
DECLARATION

I, on behalf of authors of the report: AMRUTHA K V , JITHIN K M , JITHIN T MATHEWS , PRASILA P , hereby declare that the project report "CUSTOM IMAGE CLASSIFIER " submitted for partial fulfilment of the requirements for the award of degree of Bachelor of Technology of the APJ Abdul Kalam Technological University , Kerala is a bonafide work done by us under supervision of Mr. NIKESH P (Assistant Professor, CSE Dept). This submission represents our ideas in our own words and where ideas or words of others have been included, I have adequately and accurately cited and referenced the original sources. I also declare that we have adhered to ethics of academic honesty and integrity and have not misrepresented or fabricated any data or idea or fact or source in my submission. I understand that any violation of the above will be a cause for disciplinary action by the institute and/or the University and can also invoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been obtained. This report has not been previously formed the basis for the award of any degree, diploma or similar title of any other University.

Place: Thalappuzha
Date: 20-07-2020

JITHIN T MATHEWS

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
GOVERNMENT ENGINEERING COLLEGE, WAYANAD
THALAPPUZHA – 670644



CERTIFICATE

This is to certify that the report entitled **'CUSTOM IMAGE CLASSIFIER '** submitted by **AMRUTHA K V (WYD16CS010) , JITHIN K M (WYD16CS029) , JITHIN T MATHEWS (WYD16CS030) , PRASILA P (WYD16CS045)** to the APJ Abdul Kalam Technological University in partial fulfilment of the requirements for the award of the Degree of Bachelor Technology in Computer Science and Engineering is a bonafide record of the project work carried out by them under our guidance and supervision. This report in any form has not been submitted to any other University or Institute for any purpose.

Internal Supervisor(s)

Department Seal

Project Coordinator(s) Head of the Department

ACKNOWLEDGEMENTS

First of all I would like to solicit my humble thanks to god almighty for being with me, guiding me the right way throughout my project work. I record my indebtedness to our principal Dr. V S Anitha, for her guidance and sustained encouragement for the successful completion of this project. I am highly grateful to Professor Nidheesh N, Head, Department of Computer Science, for his valuable suggestion. I also thank the project Coordinator Mr. Shabeer K P, Assistant Professor, Department of Computer Science and Engineering, Ms. Binatha C, Assistant Professor, Department of Computer Science and Engineering, Government Engineering College, Wayanad for their guidance throughout the course of this project. their positive approach had offered incessant help in all possible ways from the beginning. I take immense pleasure in expressing my humble note of gratitude to my project guide Mr. Nikesh P, Assistant Professor, Department of Computer Science and Engineering, Government Engineering College, Wayanad for her remarkable guidance in the course of completion of the project. I also extend my thanks to my faculty members and my friends for their moral support for successfully completing the project.

ABSTRACT

Users are sharing a vast amount of data through apps every day. The large volume of digital data is being used by companies to deliver better and smarter services to the people accessing it. Image recognition refers to technologies that identify places, logos, people, objects, buildings, and several other variables in images. Image recognition is a part of computer vision and a process to identify and detect an object or an attribute in a digital video or image. However, the vast amount of data can be recognized and classified only with the use of generally trained CNN. There is no specific purpose trained CNN for recognizing data with custom requirements. This can be solved by creating a custom image classifier with transfer learning to identify specific classes of objects using Nvidia jetson nano and Pytorch. A Convolutional neural network is used to recognize the images accurately. It is an algorithm used for analyzing images and assigning importance to specific bits. The last few layers of CNN are retrained to recognize, identify, and classify images more deeply.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

Image Recognition and identification is a classic machine learning problem. It is a very challenging task to detect an object or to recognize an image from a digital image or a video. Image Recognition has applications in the various field of computer vision, some of which include facial recognition, biometric systems many more. Deep Learning algorithms have achieved great progress in the field of computer vision. Convolutional neural networks (CNN) are deep learning algorithms that have high accuracy and can train large data sets with millions of parameters, in form of 2D images as input and combine it with filters to produce the desired outputs. In our project, CNN models are built to evaluate its performance on image recognition and detection data sets. The algorithm is implemented using Nvidia jetson nano and PyTorch.

1.1 CURRENT SYSTEM

With the introduction of the Convolutional neural network (CNN), there has been a wide range of image classifiers that perform near-perfect image classification and recognition of objects that the classification model is trained for. But at certain times fails to do so on improper images and on images or objects that the neural network is not trained for. Since the current system was not trained for more precise classification it does not gives a precise identification of object or image. For example, if we provide a plant's picture it just identifies it as a plant but won't give details like which plant it is. One of the most important questions raised among people was " how can we recognize an object that the neural network is not trained for? " In other words how to make your image classifier and

run it with hardware acceleration.

1.2 PROPOSED SYSTEM

Considering most of the drawbacks of the current system we propose to build a custom image classifier that will recognize something specific, for example identifying an automatic pet door for a specific breed of dog, or a plant species for sorting, or any other exiting applications you can think about.

ResNet CNN is used to train the model using python based on custom image requirements. And then it is implemented on an Nvidia jetson nano go board which is a RISC AI Module that can be used for image classification. The system will provide real-time stream to the laptop along with classification output and confidence score. Thus the proposed system was able to identify more classes than the current system could do and was more precise too.

1.3 FEASIBILITY

In terms of the feasibility study, first of all, the possibility of developing such an image classifier to a fully functional level needs great effort. Our project 'Custom Image Classifier' is comparatively feasible since it required only Nvidia jetson nano which was costly hardware other than the laptop.

1.3.1 Technical Feasibility

The proposed system will require Miniconda 3.0 supported by python 3.6 to create a new environment so that the changes done for the construction of the project model do not disturb the existing environment, it also requires a basic understanding of TensorFlow, Keras modeling, PyTorch, NumPy, etc. ResNet-18 model (convolutional neural network that is 18 layers deep) pre-trained network whose last few layers are retrained is used to train the model here. The transfer learning (re-training of the last few layers of convolutional neural network) in the classifier is done with the help of Pytorch and Nvidia jetson nano board.

The technical cost other than getting Nvidia Jetson nano board for developing Custom Image Classifier is less because all the technical support and software used are open source and also used the college server for the project at most times so the server cost is also less.

1.3.2 Economical & Financial Feasibility

The project is built around an AI module called Nvidia jetson nano which is a relatively new board used in AI. Other than the dataset requirements, the board is the only part of the system that requires financial funding. The board costs around 9.5k INR if shipped or higher if bought locally.

1.3.3 Schedule Feasibility

The system consists of tasks such as Environment setup, Dataset collection, training, Doing sanity checks, Model implementation, Re-training, and Re-modelling, etc. All

these are kept up to date with the project schedule. Each of the team members is defined to establish the project ideas and combine them in a better and effective way and schedule them to stick to the exact plan of the project and to avoid maximum confusing facts.

1.3.4 Resource Feasibility

Like any other machine learning project, this project is also high on resource dependency, for training we need a good chunk of data that we can work on. Fortunately, many of the datasets can be collected from sites like Kaggle, Getdata, etc. The proposed system is also used for transfer learning where pre-trained data are used in making the system relatively easier to implement.

1.4 PROCESS MODEL

The figure below shows an abstract view of the system training cycle. This has to be done until the output is of acceptable performance. A key part of the process is the construction of a model with peak performance. The system has an innate environment for the training of the model as well as an interface to implement the model with the Nvidia Jetson board and provide accurate output to the user.

Initial process starts with the collection of data. The data in the training set are then fed as input into the training script to train the model. Once training is completed a PyTorch model is obtained. The accuracy of the model is verified by sanity checks by providing data in the validation set. This validation set will be used for testing the model accuracy after each epoch/training rounds. This model then has to be fed into the Nvidia jetson nano development kit. Since the Nvidia jetson nano accepts only ONNX (Open Neural

Network Exchange) model, the Pytorch model is converted to the ONNX model and fed into the development kid. The ONNX model is the final model for classification and identification. Data in the test set is finally used by the user to test the accuracy of the constructed model.

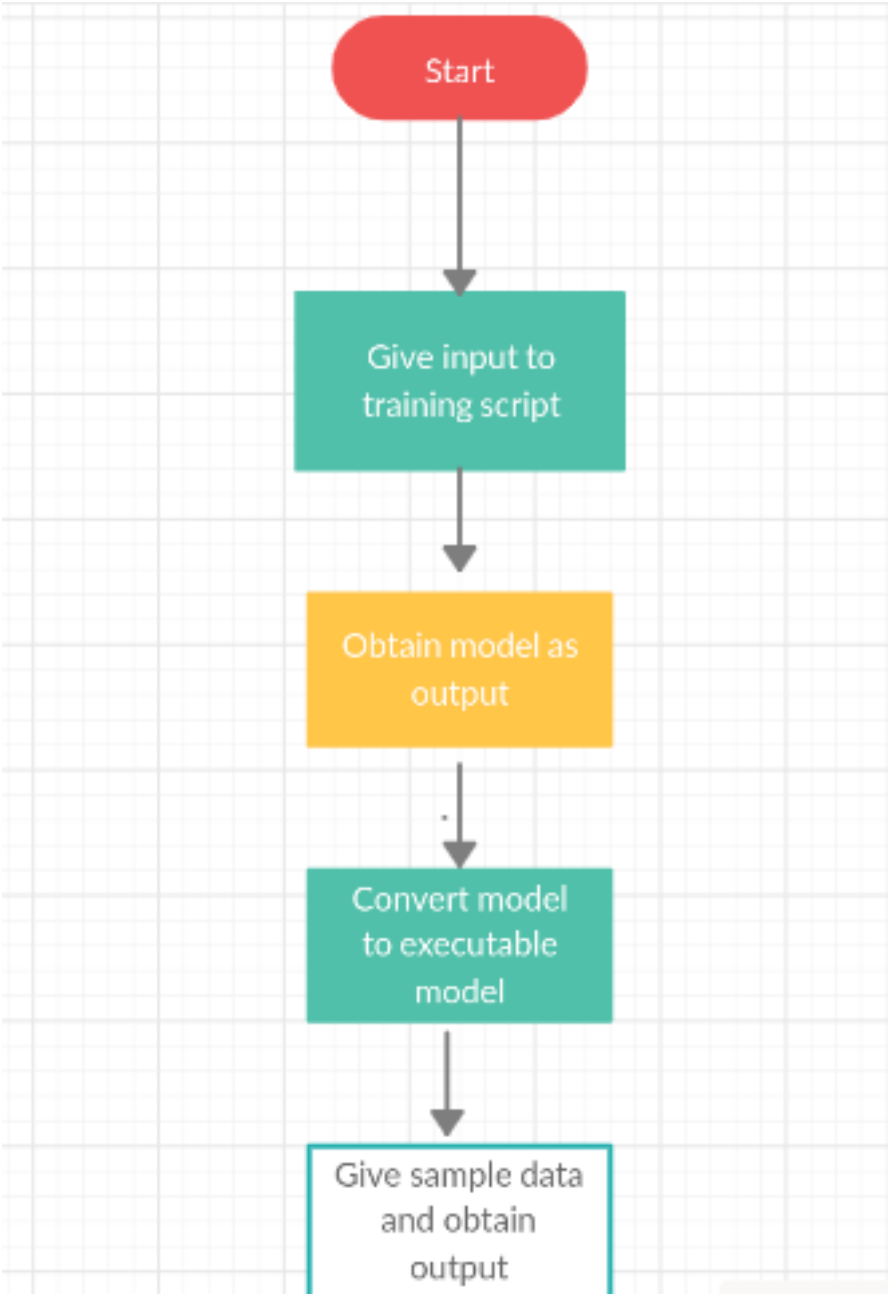


Figure 1.1: Process Model

CHAPTER 2
LITERATURE SURVEY

2.1 INTRODUCTION

In recent years there have been great strides in building classifiers for image detection and recognition on various data sets using various machine learning algorithms. Deep learning, in particular, has shown improvement in the accuracy of various data sets. We have gone through some of the works on image classification and studied their features, properties, and implementation details. The different aspects we have gone through are image recognition using machine learning and histogram.

2.2 RAPID OBJECT DETECTION USING A BOOSTED CASCADE OF SIMPLE FEATURES

This paper by P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," describes a machine learning approach for visual object detection which is capable of processing images extremely rapidly and achieving high detection rates. This work is distinguished by three key contributions. The first is the introduction of a new image representation called the "integral image" which allows the features used by the detector to be computed very quickly. The second is a learning algorithm, based on AdaBoost, which selects a small number of critical visual features from a larger set and yields extremely efficient classifiers. The third contribution is a method for combining increasingly more complex classifiers in a "cascade" which allows background regions of

the image to be quickly discarded while spending more computation on promising object-like regions. The cascade can be viewed as an object-specific focus-of-attention mechanism which unlike previous approaches provides statistical guarantees that discarded regions are unlikely to contain the object of interest. In the domain of face detection, the system yields detection rates comparable to the best previous systems. Used in real-time applications, the detector runs at 15 frames per second without resorting to image differencing or skin color detection. This was totally limited to the facial detection of human beings. Another thing was it can't handle large data set and the whole process was time-consuming too.

2.3 HISTOGRAMS OF ORIENTED GRADIENTS FOR HUMAN DETECTION

Navneet.Dalal and Bill. Triggs in paper "Histograms of Oriented Gradients for Human Detection" studied the question of feature sets for robust visual object recognition, adopting linear SVM based human detection as a test case. After reviewing existing edge and gradient-based descriptors, they showed experimentally that grids of Histograms of Oriented Gradient (HOG) descriptors significantly outperform existing feature sets for human detection. The paper says about the influence of each stage of the computation on performance, concluding that fine-scale gradients, fine orientation binning, relatively coarse spatial binning, and high-quality local contrast normalization in overlapping descriptor blocks are all important for good results. The new approach gives near-perfect separation on the original MIT pedestrian database, so they introduced a more challenging dataset containing over 1800 annotated human images with a large range of pose variations and backgrounds. This was limited to identify human detection. It uses only a

continuous data set and even does not provide specific classification.

2.4 SUMMARY

By analyzing and studying about two different existing product we understood that both had advantages as well as disadvantages, so we are supposed to develop a system that works better than the existing system and cover almost all the demerits of the existing system.

CHAPTER 3
REQUIREMENT ANALYSIS

3.1 INTRODUCTION

The purpose of this document is to find the expectations of the users for the application that is to be built or modified. It involves all the tasks that are conducted to identify the needs of different users. Therefore requirements analysis means analyzing, document, validate, and manage software or system requirements.

3.2 METHOD OF REQUIREMENT ELICITATION

Requirements elicitation practices include interviews, questionnaires, user observation, workshops, brainstorming, use cases, role-playing, and prototyping. Before requirements can be analyzed, modeled, or specified they must be gathered through an elicitation process. The similar type of applications are present, but they have less accuracy and some difficulties. In the proposed system, considering solutions to the disadvantages of previously existing systems.

3.2.1 System Study

Discussion among colleagues and faculty about the requirements. Surfing the internet for the current system's requirements that can also be included. Understand and study the current system's features as well as shortcomings. Reach to a conclusion of what to be included and avoided in our system.

3.2.2 Interaction

Prepared a series of questions related to the current system and the need for the new system. Find out the limitations of the current system and comes up with the features to be included in our product and also their feasibility. Understand the point of view of users rather than as designers and make further adjustments.

3.2.3 Group Discussion

Through group discussions with the colleagues and faculty, the need for a new system was confirmed, and had a basic idea of what requirement to be added. Do brainstorming to understand the design and implementation of the system. New ideas and innovative solutions came mostly by group discussion.

3.3 USER REQUIREMENTS

How can I recognize an object that the neural network is not trained for? Most of the time a user doesn't need to recognize some generic objects, like cats and dogs and airplanes. You want to recognize something specific, for example, a breed of the dog for that automatic pet door, or a plant species for sorting, or any other exiting applications you can think about. The user will request the specific classification requirement and that will be given by our system.

3.4 PROJECT REQUIREMENTS

The requirements for the project are to identify the goals of the project. The way to develop, the methods to follow, the technologies in co-operating to the image classification,

the resources to be collected modules to be identified.

- Software Requirements

    1. Python 3.6 or 2.7
    2. Tensorflow,keras,numpy
    3. Pytorch
    4. Minconda as a python environment manager
    5. ResNet-18 model CNN
    6. Data sets

- Hardware Requirements

    1. Nvidia jetson nano board
    2. PC
    3. Waveshare imx219-77 (camera used for the development kit.)

3.5 SUMMARY

By analysing these methods of requirement elicitation, user and project requirements we could determine the needs and conditions for the project. The project can be completed with some requirements which are easily available.

CHAPTER 4
DESIGN AND IMPLEMENTATION

4.1 INTRODUCTION

The purpose of this document is to delineate the design of the project titled 'Custom Image Classification'. The developer team responsible for the implementation of the project and those who are to maintain the software in the coming future. Design and implementation is an early phase of the project where a project's key features, structure, criteria for success and major outcomes are all planned out.

4.2 ARCHITECTURAL DESCRIPTION

The architecture comprises of the static and dynamic aspects of the system. It gives a view of the entire system highlighting the important features and ignoring unnecessary details. In our project, we use NVIDIA Jetson Nano as our development kit.

NVIDIA Jetson Nano is a small, powerful computer that lets us run multiple neural networks in parallel for applications like image classification, object detection. Jetson Nano is a GPU-enabled edge computing platform for AI and deep learning applications. The GPU-powered platform is capable of training models and deploying online learning models and is most suited for deploying pre-trained AI models for real-time high-performance inference. The neural network model supported by NVIDIA Jetson Nano is ONNX (Open Neural Network Exchange) model.ONNX supports interoperability between frameworks, which means we can train a model in one of the many popular machine learning frameworks like PyTorch which we have used in our project, and convert it into ONNX format

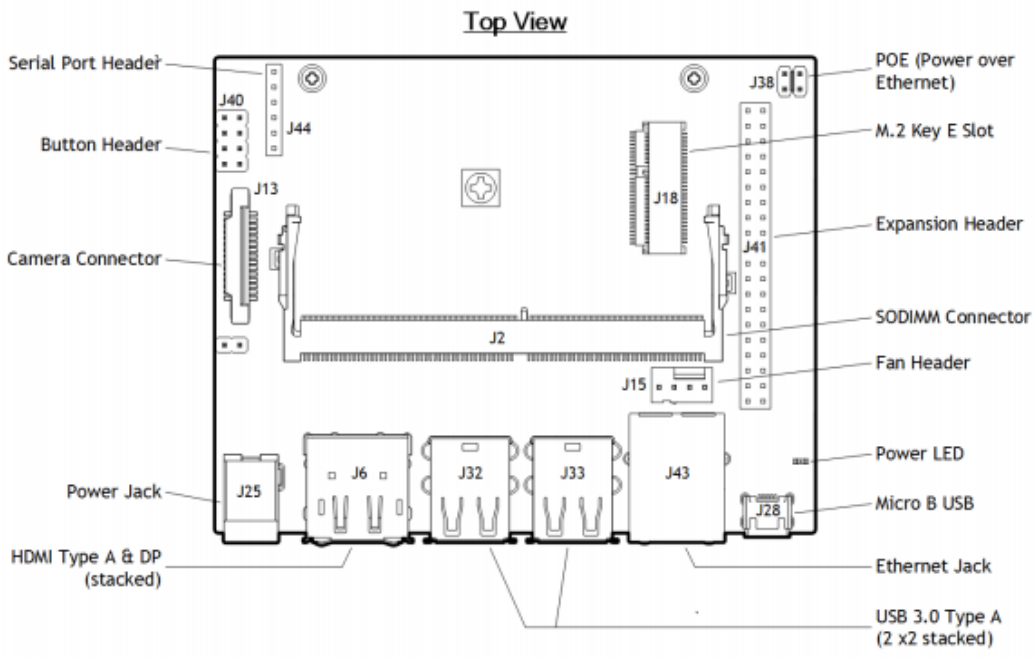


Figure 4.1: Board Architecture

and consume the ONNX model. The NVIDIA Jetson Nano module features a 1.43 GHz ARM Cortex-A57 quad-core processor and 128-core NVIDIA Maxwell graphics with speeds up to 921 MHz. Jetson Nano delivers 472 GFLOPs for running modern AI algorithms fast.

- BOARD SPECIFICATION

  1. GPU: 128-core NVIDIA Maxwell architecture-based GPU
  2. CPU: Quad-core ARMR©A57
  3. Video: 4K @ 30 fps (H.264/H.265) / 4K @ 60 fps (H.264/H.265)encode and decode
  4. Camera: MIPI CSI-2 DPHY lanes, 12x (Module) and 1x (DeveloperKit)
  5. Memory: 4 GB 64-bit LPDDR4; 25.6 gigabytes/second

6. Connectivity: Gigabit Ethernet
7. OS Support: Linux for TegraR
8. Module Size: 70mm x 45mm
9. Developer Kit Size: 100mm x 80mm

4.2.1 Decomposition Description

The System is divided into 3 modules based on the functionality of the system. The Modules are:-

- **Environment setup** - This phase consists of setting up a remote environment for the system to do training. This will prevent the system operations from not to change the operating system's environment variable and bring about critical errors. The system will install a python environment and create the environment with packages and libraries that will be needed for the training of the classification model.
- **Training** - This Module would be called after the collection of required training data. The training script will scan the input training data and associate each input data to a class that will be used to decide which classes will be new data which was not in the training set be. The purpose of this module would be to acquire the data for the generation of the model and save it for implementation. The developer would need to assign proper epoch values for the model to be accurate.
- **Real-Time Classification** - This module will convert the saved model from the training set into a model that can be run on the Nvidia Jetson nano board. The

user interface will provide a live stream from the board along with the output of the object being read.

4.2.2 User Interface Design

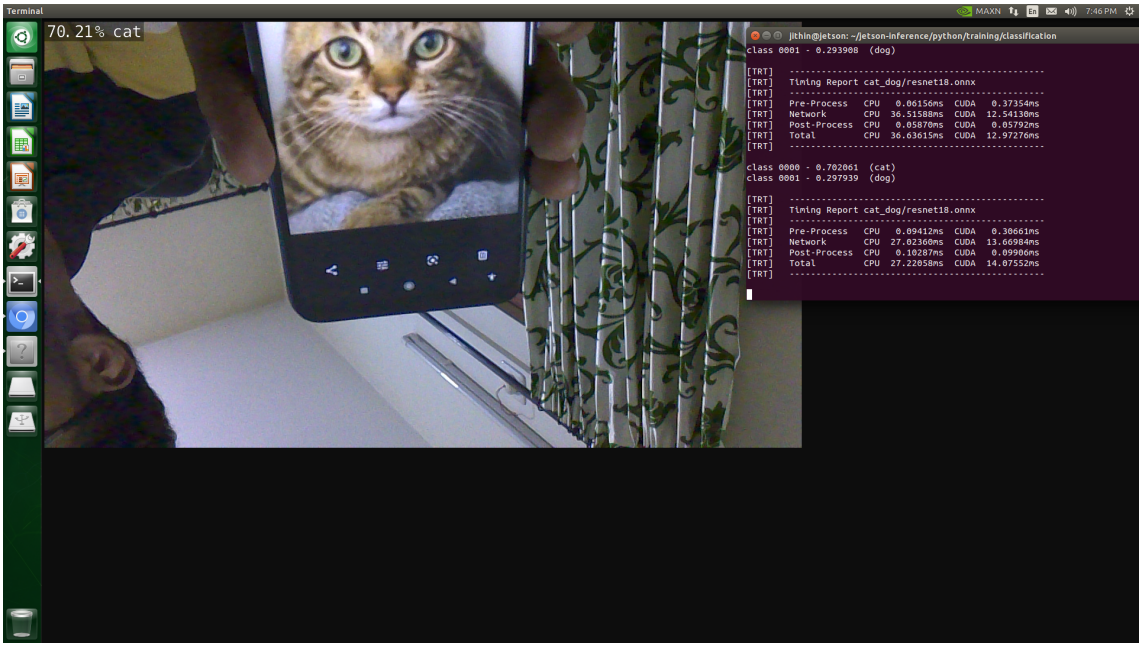


Figure 4.2: User Interface Design

The UI will be kept minimal. In fact, a minimal interface can actually contain quite a lot of complexity. The key to minimalism isn't making the interface simple by itself. It's about making it as simple as it needs to be. In other words, it means stripping back unnecessary elements, and only keeping what's strictly necessary for the functioning of the website or app. When the model is executed, the User will be shown a live stream from the camera of the development board and the predicted class of the object in front of the camera. A terminal window also will be shown which depicts the classification computations of the class.

4.2.3 Dependency Description

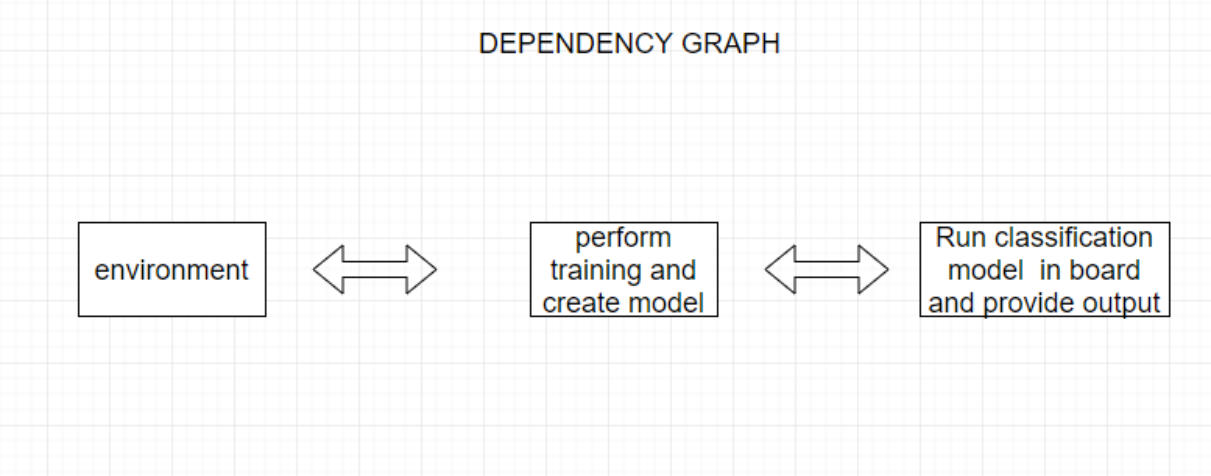


Figure 4.3: Dependency Graph

Consists of three modules which are independent of each other. The environment ensures that the training module variables and other settings do not reflect on the main system settings. It also prevents outside variables and settings from changing the training parameters. The training module uses the input data to make our classification model within the environment provided. The classification model will run the model created by the training model and provide output to the user.

4.3 TEST CASE DESIGN

4.3.1 Requirement based design

If the image is to be provided as input to the model via terminal, the image will have to be of sufficient resolution. Smaller images can be identified but the level of accuracy would be lower. We suggest a minimum resolution of 640 x 480 pixels. And if the image is to be provided as input to the model via a camera of the development board, any test cases can

be used as long as the user keeps the image in a way that is clearly visible to the system.

4.4 SUMMARY

The purpose of this chapter is to delineate the design of our project in different design viewpoints. We have successfully designed and implemented the project to achieve the desired project goals.

CHAPTER 5
CODING

5.1 INTRODUCTION

This section contains code for some key modules in the project such as model reshaping and model training. Once we have an idea of what our project's result will be, it's time to develop code. This is the most important planning stage of all. Code is done by breaking the program into mini-programs or more specifically, functions. There are many online platforms available for helping our coding section like GitHub.

5.1.1 Reshaping the pre-trained network

```
import torch

import torch.nn


# reshape the model for N classes

#

def reshape_model(model, arch, num_classes):

"""Reshape a model's output layers for the given number of classes"""


# reshape output layers for the dataset

if arch.startswith("resnet"):

model.fc = torch.nn.Linear(model.fc.in_features, num_classes)

print("=> reshaped ResNet fully-connected layer with: " + str(model.fc))
```

```
else:
print("classifier reshaping not supported for " + args.arch)
print("model will retain default of 1000 output classes")
return model
```

The above code reshapes the pre-trained model according to the number of training classes and output needed for our experiment. That is if the pre-trained model comes with 512 input and 1000 output layers and if we are about to do cat-dog model with only two example classes, the model output and input layers are set to 2.

5.1.2 Training the model

```
def train_model(model, dataloaders, criterion, optimizer, num_epochs=25
, is_inception=False):
    since = time.time()

    val_acc_history = []

    best_model_wts = copy.deepcopy(model.state_dict())
    best_acc = 0.0

    for epoch in range(num_epochs):
```

```
print('Epoch {}/{}'.format(epoch, num_epochs - 1))
print('-' * 10)

for phase in ['train', 'val']:
    if phase == 'train':
        model.train()  # Set model to training mode
    else:
        model.eval()   # Set model to evaluate mode

    running_loss = 0.0
    running_corrects = 0

    for inputs, labels in dataloaders[phase]:
        inputs = inputs.to(device)
        labels = labels.to(device)

        optimizer.zero_grad()

        with torch.set_grad_enabled(phase == 'train'):

            # if is_inception and phase == 'train':

            #     outputs, aux_outputs = model(inputs)
```

```
            ##   loss2 = criterion(aux_outputs, labels)
              #  loss = loss1 + 0.4*loss2
            #else:
                outputs = model(inputs)
                loss = criterion(outputs, labels)

            _, preds = torch.max(outputs, 1)

            if phase == 'train':
                loss.backward()
                optimizer.step()

        running_loss += loss.item() * inputs.size(0)
        running_corrects += torch.sum(preds == labels.data)

    epoch_loss = running_loss / len(dataloaders[phase].dataset)
    epoch_acc = running_corrects.double()
    / len(dataloaders[phase].dataset)

    print('{} Loss: {:.4f} Acc: {:.4f}'.format(phase,

    epoch_loss, epoch_acc))
```

```
            if phase == 'val' and epoch_acc > best_acc:
                best_acc = epoch_acc
                best_model_wts = copy.deepcopy(model.state_dict())
            if phase == 'val':
                val_acc_history.append(epoch_acc)

        print()

    time_elapsed = time.time() - since
    print('Training complete in {:.0f}m {:.0f}s'.format(time_elapsed // 60,
    time_elapsed % 60))
    print('Best val Acc: {:4f}'.format(best_acc))

    # load best model weights
    model.load_state_dict(best_model_wts)
    return model, val_acc_history
```

The train-model function handles the training and validation of a given model. As input, it takes a PyTorch model, a dictionary of data loaders, a loss function, an optimizer, a specified number of epochs to train and validate for, and a Boolean flag for when the model is an Inception model. The is-inception flag is used to accommodate the Inception v3 model, as that architecture uses an auxiliary output and the overall model loss respects

both the auxiliary output and the final output, however, this part is taken out because the ResNet model is the one being used. The function trains for the specified number of epochs and after each epoch runs a full validation step. It also keeps track of the best performing model (in terms of validation accuracy), and at the end of training returns the best performing model. After each epoch, the training and validation accuracies are printed.

5.2 SUMMARY

The chapter included the major portions of the code we have done in our project. It is an important portion of the production of outputs.

CHAPTER 6
PERFORMANCE EVALUATION

6.1 INTRODUCTION

Performance evaluation is the process of evaluating how effectively a product is fulfilling its responsibilities and contributing to the accomplishment of the user's goals.

6.2 EVALUATION SCENARIO

With the scenario evaluation, you can compare scenarios against previously defined or categorized plan goals, objectives, and risks.

6.2.1 Scenario Based Testing

- User comes across an unfamiliar plant while taking a daily stroll.

  The user will likely take a leaf, a branch of the plant, or even take a picture of the plant and bring it to the system. The system having been trained on several classes of plants is likely to identify the plant correctly. Even if the plant was not in the training data set, the system would use the leaf patterns like data it learned during training to give a prediction that is close to the correct one or may identify it as a plant that belongs to the same family as the plant found by the user.

- A city council would like to separate plastic and organic waste in their trash cans.

  The system could be attached to the top of the garbage can and connected to a motor that controls the opening of the can. The system is attached to two cans, one for plastic and the other for organic. The plastic can will open only if it detects the waste like plastic and the organic can would only open if it detects organic waste.

Hence two cans would contain separate wastes of organic and plastic.

- The user uses a pet door.

  Pet doors are mini doors engraved on regular house doors that allow their pets to enter and leave the house. Now if the area is surrounded by stray animals, they could also enter their house. If the system is used to correctly identify the house owner's pet, the door will only open for that pet, so stray animals won't enter their houses.

6.2.2 Load testing

The system can only identify single input if the input is provided via a terminal and not through the camera of the system. However, if the input is through the camera board, the system would identify any objects as long as the image can be focused on by the camera. That is if two or more images are shown in front of the camera the system will try to identify the image that is in or closer to the focusing area of the camera first.

6.2.3 Install/Uninstall testing

Installation and uninstallation are fairly simple for the custom classifier system as it is practically just another PC/desktop. Both of these can be done by the user without any technical help. The user only needs to connect i/o devices like monitor, keyboard, etc if they plan to use it as a desktop, or simply to a monitor if being used otherwise. The system runs Ubuntu, a well known OS and the system can be shut down if not in use. Uninstallation of the system is easy as unplugging the power source and disconnecting any i/o devices connected to it.

Maintenance or upgrades is as easy as copying a file from a pen-drive or downloading the upgraded model provided through the internet as the only part that changes are the trained model.

6.3 RESULTS AND DISCUSSION

6.3.1 Dataset

A total of 5000 data set was collected for the cat-dog model, 10000 for different ten species of plants for plant model and 2000 for a custom model. Image Batch assistant was the tool used to collect data. It is ab Ad-on that can be installed on your browsers. Useful for downloading all the images found on a web page as a batch file rather than downloading them individually. As for the plant and cat-dog data set, they were collected as a subset from these sources: PLANTS - PLANTCLEF DATA SET 2017-

https://www.imageclef.org/lifeclef/2017/plant

CAT DOG - subset of ILSCRV12 -

https://drive.google.com/file/d/1LsxHT9HX5gM2wMVqPUfILgrqVlGtqX1o/view.

6.3.2 Experimental setup

The collected data need to be set up before it can be fed as input for training. The data need to be split into three separate sets, 70% for training and 20% for validation, and 10% for testing.

- Training set-This will contain the majority of the data set and will be used for training.

- validation set- This was used to estimate the accuracy and how well our model had been trained after each epoch/training rounds.

- Test set- This will be used by the user to test the performance of the constructed model.

To avoid noise and perform pre-processing Feature extraction technique was performed, where we try to highlight the important parts of the image. However, doing that on a data set that contains more than 10,000images manually was thought to be infeasible. Deep learning was the second option we attempted but these could only be used to separate human-like faces efficiently. Therefore it could not be done on the data set of pens, plants.

For further accuracy, Hyperparameter learning was tried out. Here we manually modify the layers in the neural network and make slight variations in parameters and see how this affects the model accuracy. Minor improvements were made during hyperparameter learning for the custom dataset owning to poor training data. Notable improvements were found in the plant dataset.

6.3.3 Comparison

In this section, we utilized the parameter epochs and accuracy to represent the variations and changes we came across our project. .The number of epochs is a hyperparameter that defines the number of times that the learning algorithm will work through the entire training dataset. One epoch means that each sample in the training dataset has had an opportunity to update the internal model parameters.

6.3.4 CUSTOM Model

Epoch	Accuracy
0	0
10	9.68
20	11.38
30	14.88
40	16.5
50	24.3
60	37.2
70	40.79

Table 6.1: Accuracy variation for custom model.

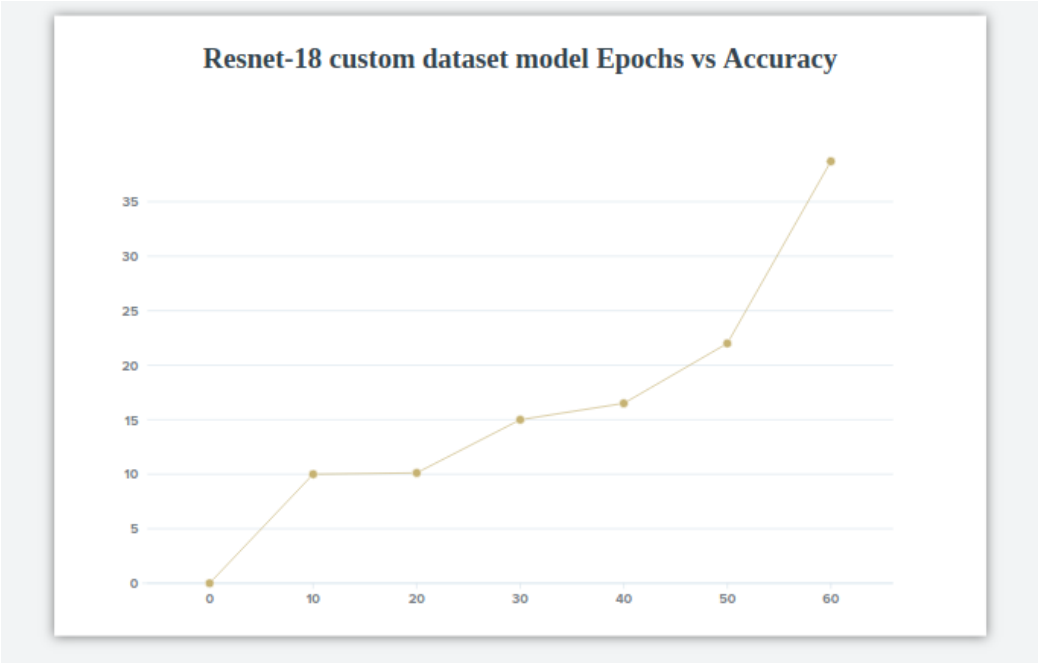


Figure 6.1: Comparison chart of custom model

In 6.1 comparison chart of the custom model shows the variation in accuracy with change in the epochs. The X-axis represents the number of epochs and Y-axis represents accuracy. Initially, accuracy was zero, and as the number of epochs increased it seemed that the accuracy too increased.

6.3.5 PLANTS Model

Epoch	Accuracy of Top-1	Accuracy of Top-5
0	10	42
10	19.68	53
20	24.5	60.38
30	30	66.28
40	40.03	75
50	42	77.5
60	50.2	80
70	55	85

Table 6.2: Accuracy variation for plant model.

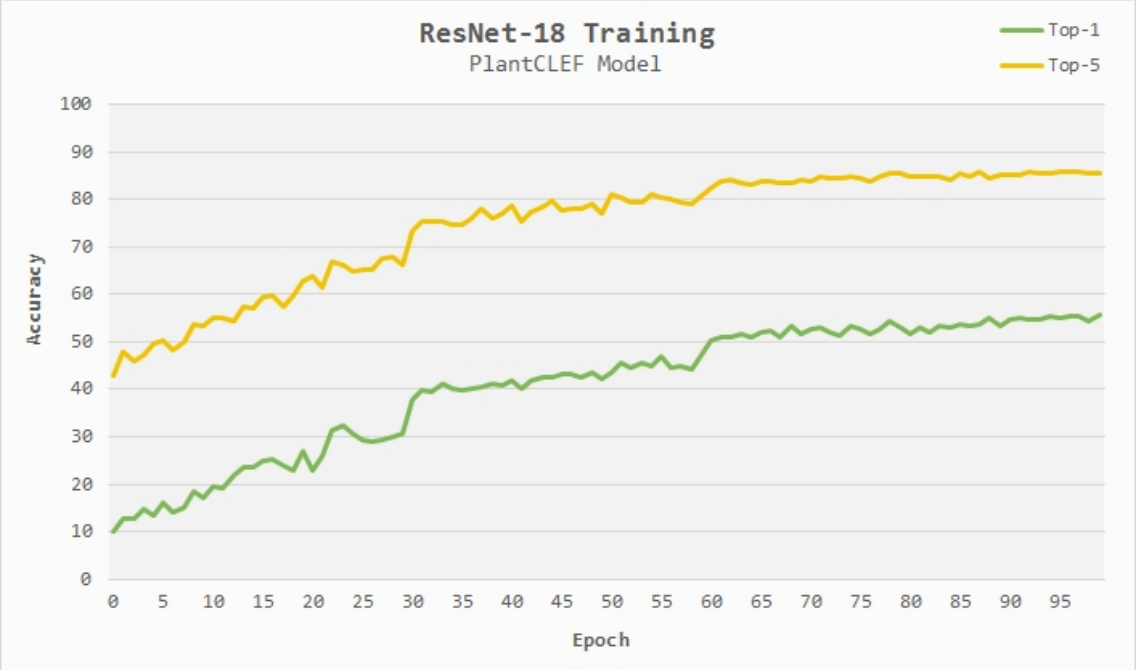


Figure 6.2: Comparison chart of plants model

In 6.2 comparison chart of the plant model shows the variation in accuracy with change in epochs. The X-axis represents number of epochs and the Y-axis represents accuracy. There are two lines shown, one for top 1 accuracy while other for top 5 accuracies. The top-1 accuracy rate is the ratio of images whose ground truth category is exactly the prediction category with maximum probability, while the top-5 accuracy rate is the ratio of images whose ground-truth category is within the top-5 prediction categories.

6.3.6 CAT-DOG Model

Epoch	Accuracy
0	50
10	59.68
20	69.38
30	72.08
40	79.5
50	78.5
60	80.2
70	82.5

Table 6.3: Accuracy variation for cat-dog model.

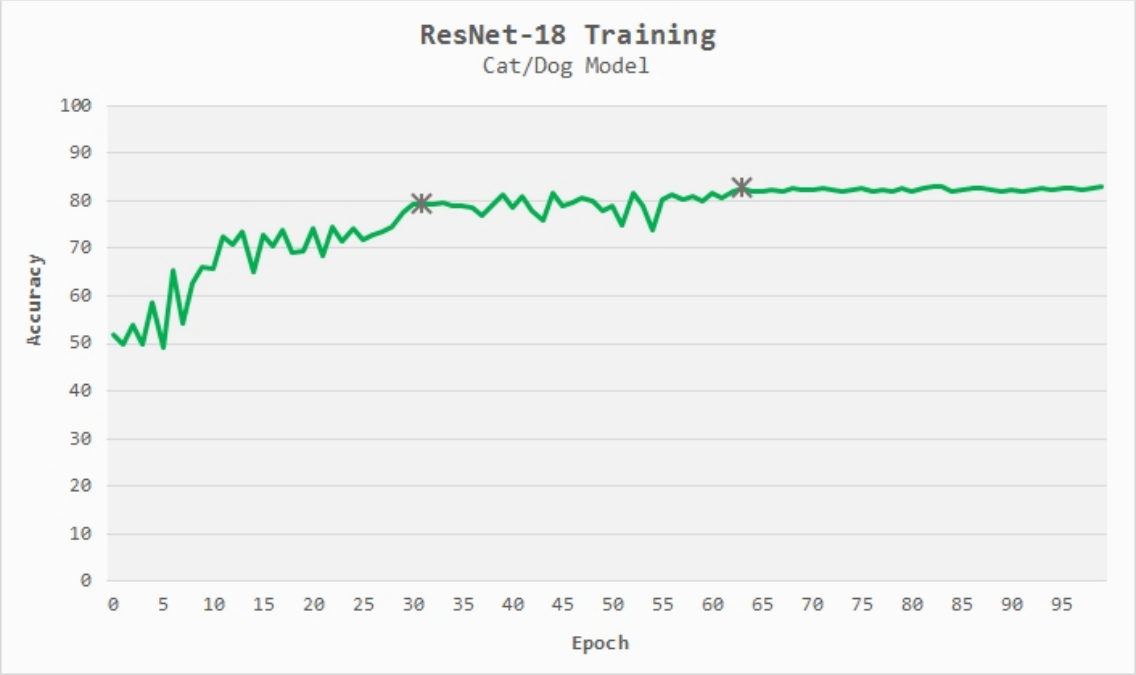


Figure 6.3: Comparison chart of cat-dog model

In 6.3 comparison chart of the cat-dog model shows the variation in accuracy with change in the epochs. The X-axis represents the number of epochs and the Y-axis represents the accuracy. There are no two lines shown here because it has only two classes. Since this Cat/Dog example only has 2 classes (Cat and Dog), Top-5 is always 100.

6.3.7 Analysis

Our proposed system retrains the last few layers of the traditional CNN to classify its inputs more deeply. The system is easier to implement with the use of transfer learning where a pre-trained model is used. The proposed system will provide real-time classification along with its confidence score. The model was able to classify the plants with good confidence score but the score for the custom dataset was relatively low, this was because of the lack of training data and the data that could be collected was of low quality. During the research phase of the project, it was found out that the accuracy of the model depends on:

- The size of training data up to a point.More the data, the better the accuracy.
- The number of training rounds or Epochs done. On average 35 epochs produce good results and accuracy starts to converge at around 70 epochs.
- The number of input layers used. More layers provide better top class predictions.
- Hyperparameter learning where we manually reshape the neural network was able to provide an accuracy boost of 2-10 percent provided the training data was sufficiently large.

6.4 SUMMARY

Through performance evaluation, we could find out the limitations regarding our proposed method.

CHAPTER 7
DOCUMENTATION

7.1 INTRODUCTION

This section contains the documentation for the product "Custom image classifier". Classification is the process of predicting the class of given data points. Classes are sometimes called as targets/ labels or categories. Classification predictive modeling is the task of approximating a mapping function (f) from input variables (X) to discrete output variables (y). We use ResNet-18, a CNN which is 18 layers deep as our pre-trained model. Simply, a pre-trained model is a model created by someone else to solve a similar problem. Instead of building a model from scratch to solve a similar problem, you use the model trained on other problems as a starting point. Hence, we reshape and re-train the imported ResNet network to form a model that can classify according to our end user's demands. The user's demand can be a deep classification of any classes of images provided there are sufficient data set available.

If you are given a few plants, the current pre-trained model will classify them as plants and nothing more. However, our model which has undergone significant training in a required class will identify the plants more specifically.

7.2 INSTALLATION

If the user wishes to do more training on our trained model, they may install miniconda3 (https://docs.conda.io/en/latest/miniconda.html), an environment manager on python that helps to separate the system environment and the training environment so that changes in one may not affect the other. And also need to familiarize themselves with PyTorch

(https://pytorch.org/) which is an ML library used to download and train models.

If the user only wishes to run our model and identify the classes he wants to, there is no need for further setup as everything will be pre-installed in our jetson system. The user only needs to connect the jetson board to a monitor, keyboard, mouse or any other i/o devices needed as the board can also be used as a normal desktop.

7.3 WORKING WITH THE PRODUCT

Once the product has been set up in the user's desired place, they only have to enter the run command for the trained model and bring any object or image they want to be identified. The user can provide input to the model in two different ways, either through the camera or terminal. Commands for both these ways will be provided in the system. The user only need to run it and then have their objects or images classified.
Sample command (cat-dog model) for camera input classification which will be provided in the system, the user only needs to copy-paste it in the terminal.

```
imagenet.py --model=cat_dog/resnet18.onnx
--input_blob=input_0 --output_blob=output_0
--labels=$DATASET/labels.txt csi://0
```

The UI will pop up with a real time stream from the camera being shown.

7.4 CONTACT

For more info contact the following

Jithin T Mathews- jithintmathews@gmail.com

Amrutha K V - amruthakvammu@gmail.com

CHAPTER 8
CONCLUSION AND FUTURE WORK

8.1 CONCLUSION

We were success-full in classifying images specifically with upmost accuracy of 60% - 90%. The confidence score provided along with the classification helped the user in knowing how accurate the identification was. Through our project Deep learning architecture known as the convolutional neural networks(CNN) has proved to be a powerful tool for classification with high accuracy which can train large data sets with millions of parameters. With the availability of data sets, enhanced computing power, and new algorithms to train CNN, we were successful in applying CNN to image processing with impressive performances, such as extraction of data required, etc. In our project, we were able to create a real-time multiple object detection and recognition application in Python with the help of Nvidia Jetson nano. We hope that our system was efficient enough to support the people to identify images/objects automatically. This can be accessed to all users due to its low cost and high efficiency.

8.2 ADVANTAGES

With today's technology, image recognition is becoming more and more reliable. The success rate is high as it can deal with 2D and well as 3D images. The image recognition using a convolutional neural network can fully automate the process and ensure its accuracy at a very high rate and can even train large data sets with millions of parameters. This says about higher convenience and lowers costs. By retraining the last few layers of the convolutional neural network precise classification with better accuracy is

possible. As a result, can identify more classes than the original model. Their ability to learn by example makes them very flexible and powerful. They are also very well suited for real-time systems because of their fast response and computational times which are due to their parallel architecture. Another advantage is that system can be integrated into machines so that it can do a specific action once a specific class is recognized, for example, opening the dog house if it's your pet dog and not for any stray animal. The GPU-powered platform in Nvidia Jetson Nano had the capability of training models and deploying online learning models and is most suited for deploying pre-trained AI models for real-time high-performance inference. Image recognition is a powerful technology that brings immense advantage to the companies and end-users, helps them enhance their security, and track down the trespassers.

8.3 LIMITATIONS

Since there are a lot of image classes that can be more deeply and specifically classified, it would be impossible to create a perfect one for all models that can classify everything it is given as input. Therefore we have to choose things the model will classify and train the model on those sample images. A new model will have to be constructed for each instance of new user demand and that will be time consuming and training will have to be done from the beginning. Sometimes the accuracy of the model is deeply affected by the quality of the camera attached to the development board. The camera attached to the development board had poor autofocus performance and its quality was not much satisfying. Another limitation was that the model may be quick to detect the noise in the test data image's environment and classify it accordingly, due it this reason

the model may sometimes classify based on the background data. A large amount of data was required to obtain enough accuracy and that was the major limitation we faced in our project due to the lack of availability of data set for pen, marker, etc. This made the project hard because data size is important in machine learning. Another aspect that was not considered in the project was that the output obtained after classification was not utilized or was not converted to any mechanical action for example raising an alarm if it identified a snake.

8.4 FUTURE EXPANSIONS

Image recognition has become a reality and a popular topic of research. It has become one of the most worked technologies of this era. Image recognition has completely changed the way we perceive, classify, and search for visual information. Many companies and industrial sectors like e-commerce, marketing, gaming, social media, sports, as well as healthcare, and automotive safety are working to provide better solutions. Given the rise of image recognition usage in the medical industry, there will be an ever-growing need for smarter technology in the future. This advancement will benefit people by using technology to help with tasks that are too complex or time-consuming for humans to complete alone.

CNN has a wide scope in the future. Researchers are constantly working on new technologies based on neural networks. Everything gets converted into automation hence there is much efficiency in dealing with changes and can adapt accordingly. Its applications also help to minimize the number of errors by increasing accuracy by the use of epoch. Due to the growth in new technologies, there are many opportunities for engineers

and neural network experts. Hence in the future neural networks will prove to be a major job provider.

REFERENCES

1. Paul Viola Michael Jones. Rapid object detection using a boosted cascade of simple features. *Mitsubishi Electric Research Labs Compaq CRL 201 Broadway, 8th FL One Cambridge Center*, 14(2):131–142, 2001.

2. Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. *INRIA Rhone-Alps*, 2005.

3. Andrew G. Howard Menglong Zhu Bo Chen Dmitry Kalenichenko Weijun Wang Tobias Weyand Marco Andreetto Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *Google Inc.*, 9(3), 2015.

4. Ferhat Culfaz. transfer learning using mobilnet and keras. *towardsdatascience.*, 9(3), 2018.

5. Zepan. Train, convert, run mobilenet on sipeed maixpy and maixduino. *bbs.sipeed*, 9(3), 2019.