Cloud Computing Project Report

PET DETECTION

Alberto Mejia Diez - 1381799
Binit Bambhroliya - 1377211
Binyam Tekeste - 1051540
Klea Maloku - 1376461
Krishna Borisagar - 1384259
Monika - 1411779

Under the Guidance of Prof. Dr. Christian Baun
High Integrity systems
Frankfurt University of Applied Sciences

July 2023

Contents

List of Figures

1 Abstract

Cloud Computing is one of the most actual concepts now-days due to the flexibility it offers to users, that do not need to buy hardware or software devices in order to fulfill their demands. In this paper we will try to reflect our understanding of a fraction of cloud, by representing our project on "Pet Detection", developed using Raspberry Pi to set up our own cluster. The model developed will be able to detect in real time, if an object is a dog or a cat and to what percentage of accuracy.

2 Introduction

Cloud computing has already become an important topic in today's business industries and not only in the computer sciences or IT related ones. One of the main reasons for this would be that we can define cloud computing services with the "use what you need" and "pay only what you use" philosophy. Costumers in different needs, would demand and be provided with network access to a shared group of resources, that will execute their tasks as well as maybe other clients task. None of the costumers would not have to permanently buy the hardware or software components they would need for their tasks, just pay the cloud provider for the time they will use the services. Different services can be provided and accessed by cloud and some of them are: Infrastructure-as-a-Service (IaaS) (offers storage, virtual maschines, network,etc), Platform-as-a-Service (PaaS) (offers a complete developement environment), Software-as-a-Service (SaaS) (offers licenced application software). [1]

Narrowing it down from the clients and providers, to the process of learning and practicing, Cloud Computing is nowdays subject to many IT related fields students and not only. A better way of leaning would be by organising and conducting studies and projects offering a hands on experience, which in many cases includes Raspberry Pi as hardware. This is also our case and in the following sections we will give a representation of our project on the Cloud Computing SoSe23. In section 3 we will start with our model representation, continuing with Kubernetes Cluster on section 4, following with, API, Frontend, Conclusions and Challenges and Improvements.

3 Object detection model

The cat and dog detection algorithm was done using machine learning model YOLO (You Only Look Once). YOLOV1 paper was published in 2015 by Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadihas [10]. Since then, the model has been improved many times up until version 8, which is the version used in this project. It employs a single convolutional neural network and has gained great popularity because of its speed, detection accurracy, good generalization, and the fact that it is open-source. The YOLO prediction algorithm process can be summarized into 5 steps which are:

1. **Divide the Image:** YOLO starts by taking an image and dividing it into a grid. For example, it might divide the image into a 13x13 grid. Each cell in this grid is responsible for detecting objects that fall into it.

2. **Predict Bounding Boxes:** Each cell in the grid will then predict a certain number of bounding boxes. A bounding box is a rectangular box that can be drawn around the object that the model detects. Each bounding box prediction includes information about the box's center (x, y coordinates), its width and height, and a confidence score that indicates how certain the model is that the box contains an object.

3. **Class Prediction:** In addition to predicting bounding boxes, each cell also predicts a class probability. This is a probability distribution over all the possible classes (e.g., cat, dog, car, etc.) that tells us how likely it is that the detected object belongs to each class.

4. **Filtering:** After the bounding boxes and class probabilities are predicted, YOLO filters out the boxes that have a low confidence score (i.e., the model isn't very sure that these boxes contain an object). It also uses a technique called non-maximum suppression to remove redundant boxes that overlap too much with other boxes.

5. **Output:** The final output of YOLO is the remaining bounding boxes and their associated class labels. Each bounding box's coordinates are scaled up to match the original image size, and the class with the highest probability becomes the label for that box.

The Ultralytics pyton library offers a straightforward approach which allows a simple implementation of different YOLO model versions [9]. It also provides some sample datasets for the users to carry out basic implementations of the model with just a few lines of codes available in the documentation. It can be installed by just running the following line of code in the console.

Listing 1: Ultralytics install

```
pip install ultralytics
```

It must be mentioned that the Ultralytics library relies on PyTorch so it is recommended to install base on your machine specifications before installing Ultralytics [7]. After installing Ultralytics it was also necessary to search for an image repository which contained numerous examples of cats and dogs, along with the proper label for the images in order to train the model. Roboflow´s Oxford-Pets V2 library, maintained by Brad Dawyer, contains 3680 images of Cats and Dogs [8]. Once downloaded, the folder contains a data.yaml file with all the training data, which can then be directly used to train the model. The following lines of code show how the model can be trained:

Listing 2: Model Training

```
from ultralytics import YOLO

# Load a model
model = YOLO('yolov8n.yaml')  # build a new model from YAML
model = YOLO('yolov8n.pt')  # load a pretrained model (recommended for trai
model = YOLO('yolov8n.yaml').load('yolov8n.pt')  # build from YAML and tran

# Train the model
model.train(data='C:/Users/alber/Documents/Alberto/Trabajo/Maestria/FUAS/S5
```

As seen in image fig.1, the training data is only about 70% of the whole data (2.6k images) with 1/3 of the images being cats and the other 2/3 dogs.

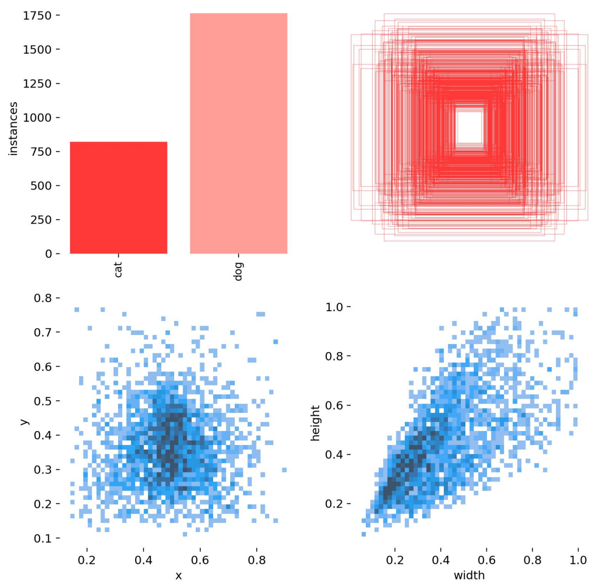


Figure 1: Image labels

The statistical results of the model training are illustrated in Fig.2. The graphs shows the distinction in the model performance between the training and validation processes.

Training is conducted in stages known as epochs, which in the case of this model, was set to 100. In each epoch, the training subset is utilized to enhance the model's performance. Following each epoch, the model undergoes validation using a separate validation set. This procedure ensures that the training halts when the error of the

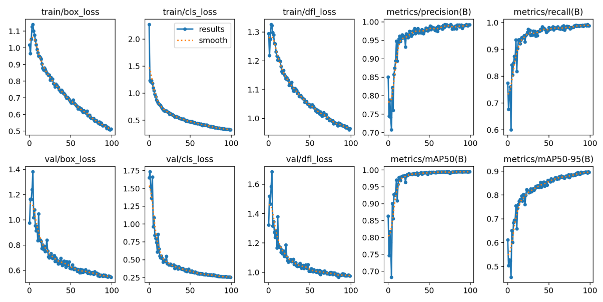


Figure 2: Model Performance Results

validation set starts to increase while the error of the training subset decreases, thereby preventing overfitting.

The term 'Box loss' refers to the model's ability to accurately locate the center of an object and the precision of the predicted bounding box in covering the object. For both the training and validation processes, the box loss decreases.

'Object loss' measures the model's proficiency in predicting a region that contains an object. Similarly as before, the object loss for the training decreases, as well as for the validation.

The 'Class loss' signifies the model's capability to predict the correct class. In this instance, the class loss starts low, followed by a sudden increase, which could be due to the fact that there is a bigger amount of dogs compared to cats, resulting in short over-fitting in the first epochs. After further runs the model prediction performance improves again.

Precision and recall are calculated using the formulas:

$$Precision = \frac{TP}{TP + FP} \tag{1}$$

$$Recall = \frac{TP}{TP + FN} \tag{2}$$

The Mean Average Precision (mAP) is computed for a confidence threshold of < 0.5 and > 0.5 and < 0.95. Both precision and recall, as well as mAP0.5 and mAP0.5-0.95, show an increasing trend during the training process. This indicates a low amount of false negatives and false positives in the model which is good.

Fig.3 shows the classification output obtained by the model using the validation data which were in total 736 images. In total there are only 21 miss-classifications out of the 3 possible classes which are: Cats, Dogs, and Background.

Once completed this whole process, the new model weights are stored and can afterwards be used to classify new images.

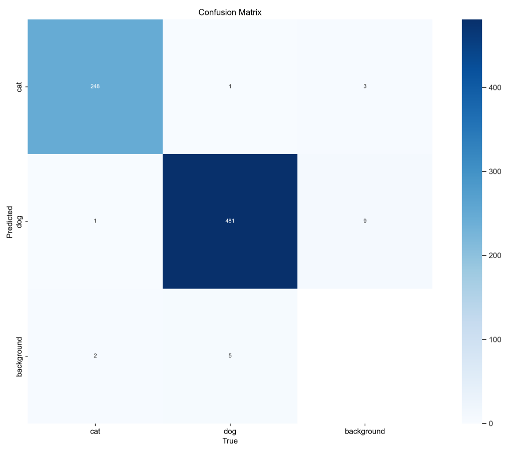


Figure 3: Confusion Matrix

4 Kubernetes Cluster

4.1 Installation procedure of Raspberry Pi OS

First needs to download Raspberry Pi Imager and install on a local machine [5]. After installing, choose Raspberry Pi OS (32-bit) for the operating system then select mircroSD card as the following Figure 1 & Figure 2.

After successful installation, assign a unique hostname for each Raspberry Pi, for example, ”kmaster” for the master node and ”knode1” for the first worker node. Save and Write in the SD Card.

After successful installation, assign a unique hostname for each Raspberry Pi, for example, ”master” for the master node and ”worker1” for the first worker node.Similarly, install OS on two more raspberry pi worker nodes ”worker2” and ”worker3”.Save and Write in the SD Card shown in Figure 3.

Insert all the SD cards back to the Raspberry Pi and power up and connect them to your network via LAN switch

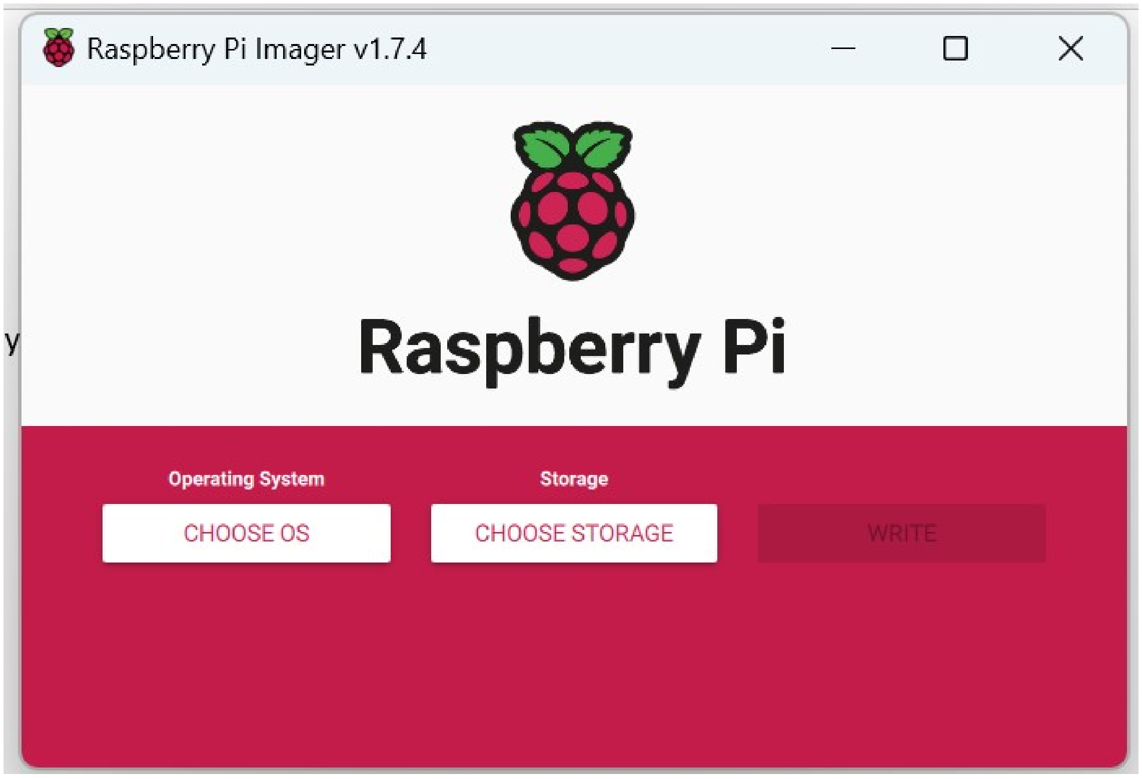
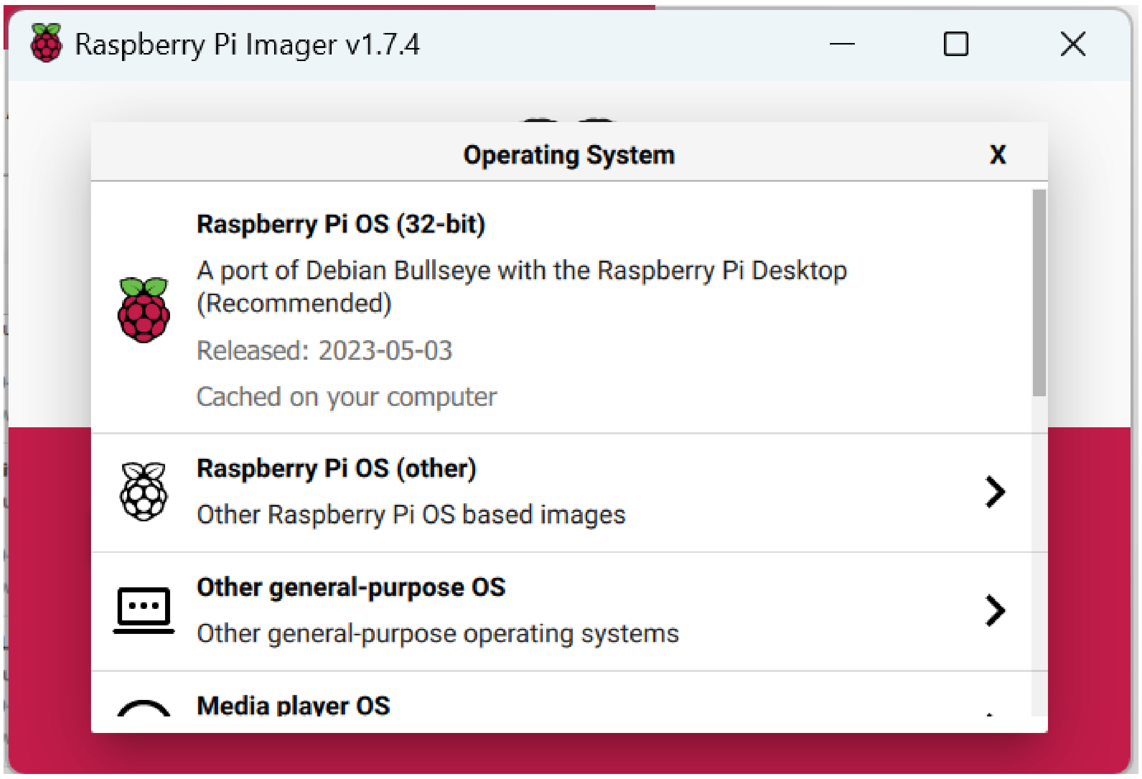


Figure 4: Raspberry Pi Imager



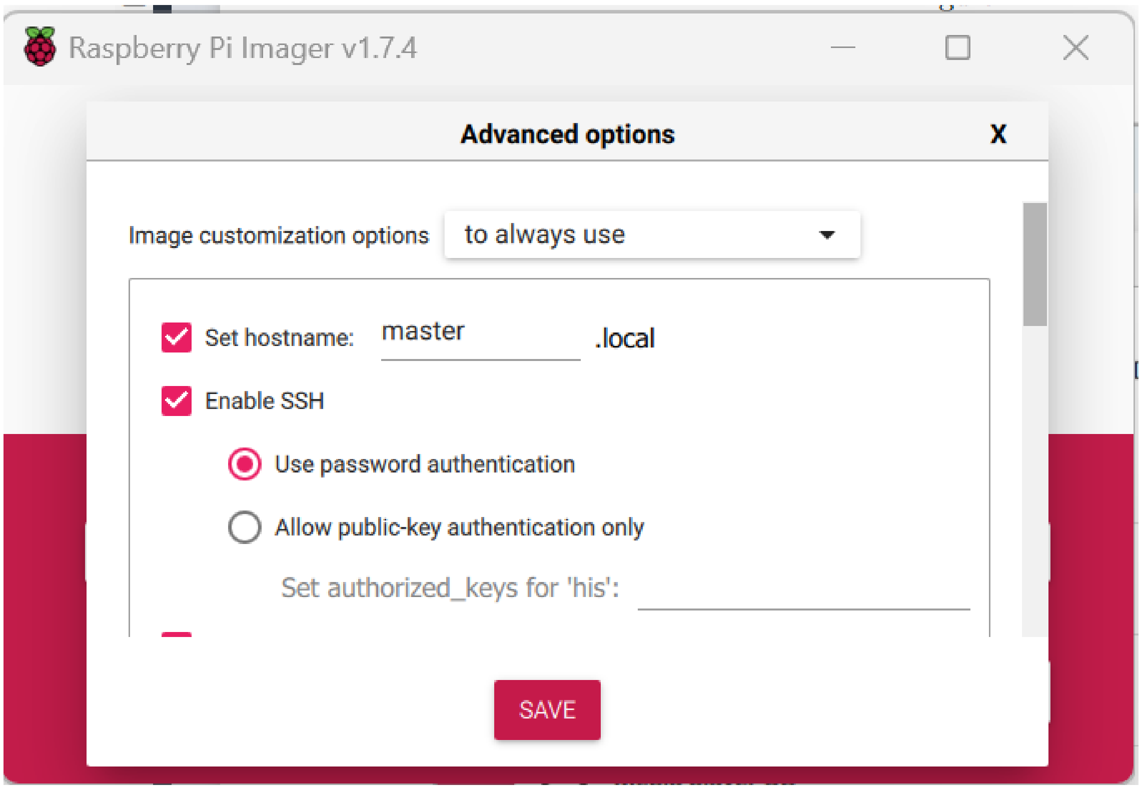
Raspberry Pi OS (32-bit)
A port of Debian Bullseye with the Raspberry Pi Desktop (Recommended)
Released: 2023-05-03
Cached on your computer

Raspberry Pi OS (other)
Other Raspberry Pi OS based images

Other general-purpose OS
Other general-purpose operating systems

Media player OS

Figure 5: Raspberry Pi OS(32-bit)


Raspberry Pi Imager v1.7.4

Advanced options

Image customization options: to always use

[✓] Set hostname: master .local

[✓] Enable SSH

(●) Use password authentication

( ) Allow public-key authentication only

Set authorized_keys for 'his':

Figure 6: Image Customization Options for all nodes

4.2 Setting up k3s cluster

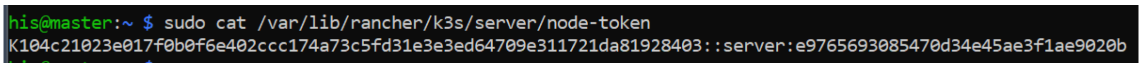
important step we followed is to enable the cgroup memory. SSH into all the Raspberry Pi 3 and update the cmdline.txt file. We used “sudo nano /boot/cmdline.txt” command in all the hosts and added “cgroup_memory=1 cgroup_enable=memory” on the end of the first line and save the file. then reboot using “sudo reboot” and start again.

1.To install K3s [6] the following command must be executed on the master node:

```
curl -sfL https://get.k3s.io | K3S_KUBECONFIG_MODE="644" sh -s -
```

2. Run : **sudo systemctl status k3s** to check status of the master nodes

3. After this step, the token from the master node can be read from the file with the code **sudo cat /var/lib/rancher/k3s/server/node-token** shown on figure 4.



```
his@master:~ $ sudo cat /var/lib/rancher/k3s/server/node-token
K104c21023e017f0b0f6e402ccc174a73c5fd31e3e3ed64709e311721da81928403::server:e9765693085470d34e45ae3f1ae9020b
```

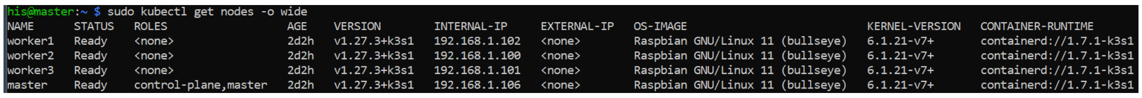
Figure 7: Node-tocken from k3s server

4. With this token, k3s can be installed[3] on worker nodes by the following command:

5. Run : **sudo systemctl status k3s-agent** to check status of the worker nodes and repeat these steps in all worker nodes

```
curl -sfL https://get.k3s.io|K3S_TOKEN="<TOKEN>" K3S_URL="https://<master_node_ip>:6443" sh -
```

6. Now all worker nodes are connected to master node with k3s cluster and to validate k3s on master node run: **sudo kubectl get node -o wide** and result is shown in figure 5.



```
his@master:~ $ sudo kubectl get nodes -o wide
NAME      STATUS   ROLES                  AGE    VERSION        INTERNAL-IP     EXTERNAL-IP   OS-IMAGE                         KERNEL-VERSION   CONTAINER-RUNTIME
worker1   Ready    <none>                 2d2h   v1.27.3+k3s1   192.168.1.102   <none>        Raspbian GNU/Linux 11 (bullseye)   6.1.21-v7+       containerd://1.7.1-k3s1
worker2   Ready    <none>                 2d2h   v1.27.3+k3s1   192.168.1.100   <none>        Raspbian GNU/Linux 11 (bullseye)   6.1.21-v7+       containerd://1.7.1-k3s1
worker3   Ready    <none>                 2d2h   v1.27.3+k3s1   192.168.1.101   <none>        Raspbian GNU/Linux 11 (bullseye)   6.1.21-v7+       containerd://1.7.1-k3s1
master    Ready    control-plane,master   2d2h   v1.27.3+k3s1   192.168.1.106   <none>        Raspbian GNU/Linux 11 (bullseye)   6.1.21-v7+       containerd://1.7.1-k3s1
```

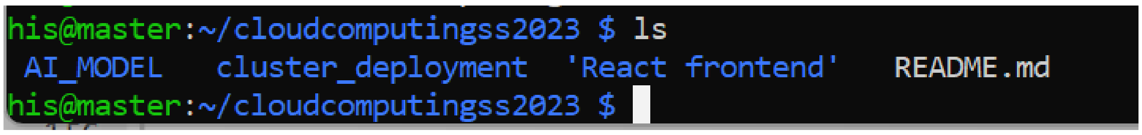
Figure 8: Nodes connected to k3s cluster

4.3 Downloading / Copying deployment files

Copy or clone project on the master node from github repository

git clone https://github.com/Binit888/cloudcomputingss2023.git

Now we have an access the data in Our Masternode is look like,



```
his@master:~/cloudcomputingss2023 $ ls
 AI_MODEL   cluster_deployment  'React frontend'   README.md
his@master:~/cloudcomputingss2023 $
```

Figure 9: List Of Directory In Github

4.4 Creating Docker Image of the Kubernetes Cluster Applications

4.4.1 webapp

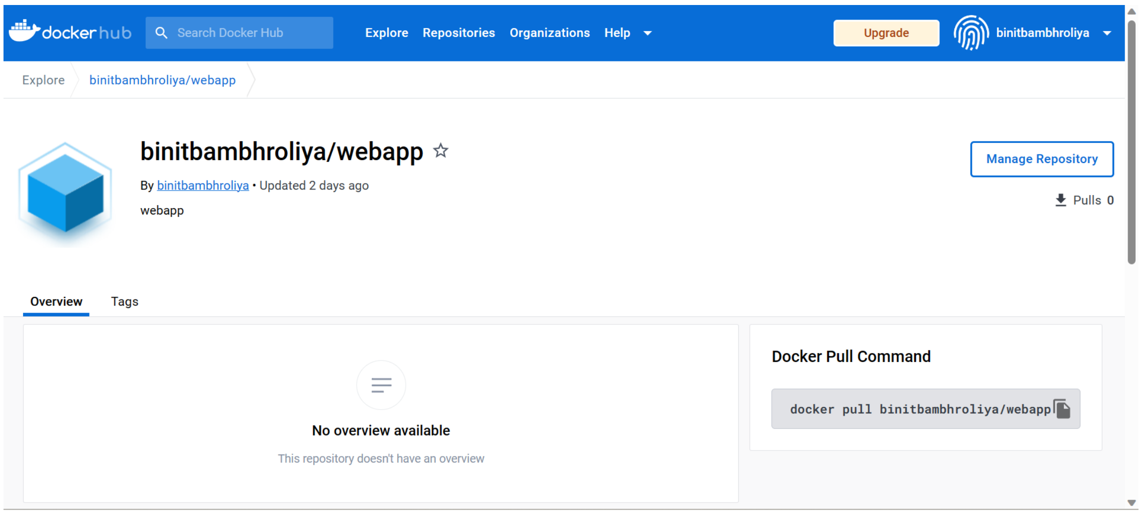
WebApp is the main custom user interface which shows all the cats and dogs detected by pat detection model on rasperypi 4 . It reads all the objects stored in the database and shows them in the Web browser.

Docker image of application can be pulled from the https://hub.docker.com/r/binitbambhroliya/webapp and can also be built locally by executing script buildImage.sh in the respective application directory

4.5 Deploying Pods and Services on Worker Nodes

4.5.1 WebApp Deployment

Go to directory : cd binitbambhroliya/cluster_deploylemt/React frontend
Execute: sh deploy.sh
After successful deployment, running 'sudo kubectl get pods -o wide' will give the output of running pods.



binitbambhroliya/webapp

By binitbambhroliya • Updated 2 days ago

webapp

No overview available

This repository doesn't have an overview

Docker Pull Command

```
docker pull binitbambhroliya/webapp
```

Figure 10: webapp docker image

5 API

5.1 NPM Deployment

Npm is a package manager and runtime environment that can be used to run javascript applications. In order to install npm, follow this tutorial: https://docs.npmjs.com/downloading-and-installing-node-jsand-npm . In order to deploy the Web-App locally on an amd Windows/Linux machine, using the npm package manager.

5.2 API

The back-end was responsible for routing the data from the sensor node to the database and from the database to the front-end for it to be displayed there. In order to be able to handle those tasks, the backend offers an API. There are essentially two endpoints that can be used and that are reachable under the configured back-end address, plus "/get", plus "/post". So if the Web-App is running on a local machine, ...

Follow the Steps to open "BackendAPI" folder and to run a project on local maschine:
2. Open the "workspace.code-workspace" file.
3. Open new Terminal.
4. Write the code as shown below in steps:
i) npm install ii) npm run dev
5. Paste the following link in browser : http://localhost:8080/post

5.2.1 GET

For the GET API the address could look like this: "http://localhost:8080/get/[image]".

Providing image-name with its extension will download an image in your local device. This is how GET API works.

5.2.2 POST

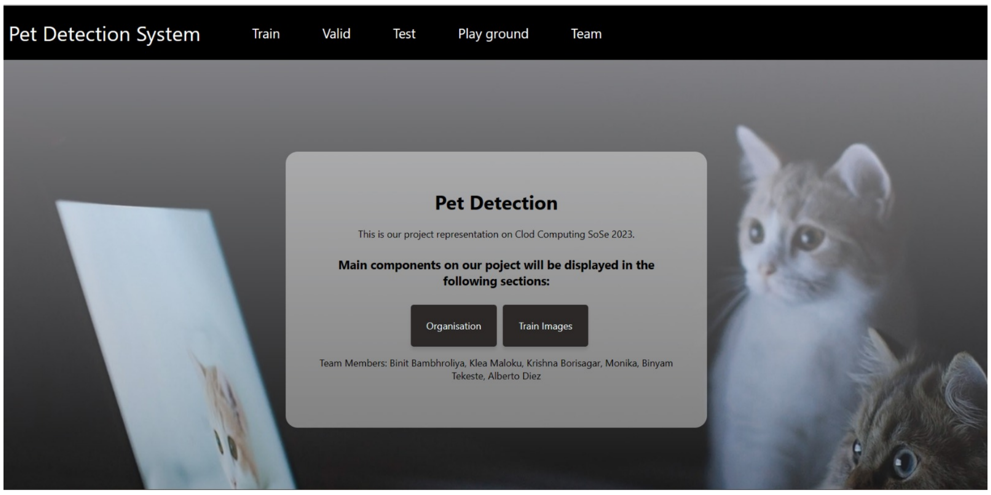
For the POST API the address could look like this: "http://localhost:8080/post".

The first functionality realized in the API is the data flow from the sensor node over the back-end to the database. If the sensor node detects a Dogs or Cats, it can send a REST post-request to the API address, where the body contains data needed to save a Dogs or Cats detection correctly. To be precise, the back-end expects to receive JSON data that sends the base64 encoded JPG or PNG of the detection. Additionally, the number of detected Dogs or Cats and their confidence levels must be sent.

6 Frontend

The frontend task in our project included creating a webpage displaying a representation of our work. For this task Visual Studio combined with React have been used. On this page you will find the pictures that we used to train, validate and test our model, also some general information about the team.

The following pictures will show how the frontend currently looks like, seeing the name of the project "Pet Detection System", and 5 more fields on Train, Valid, Test, Playground, Team.



Pet Detection

This is our project representation on Clod Computing SoSe 2023.

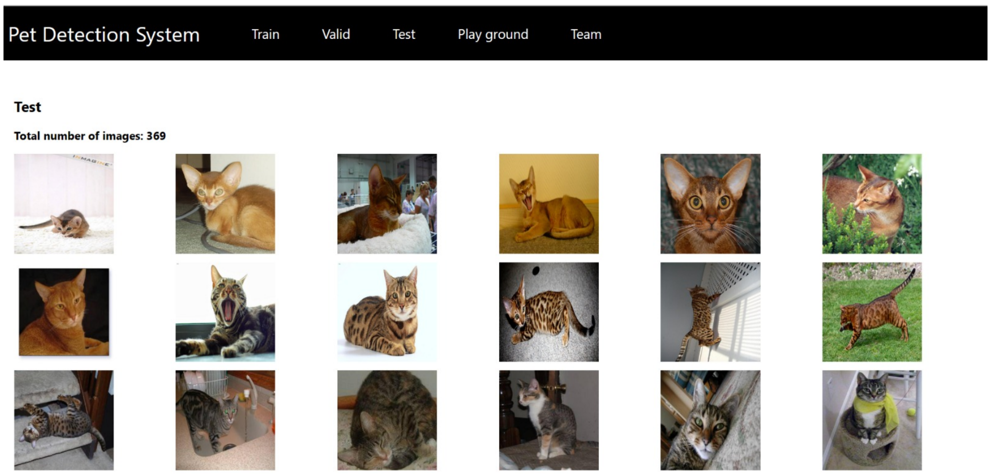
Main components on our poject will be displayed in the following sections:

Team Members: Binit Bambhroliya, Klea Maloku, Krishna Borisagar, Monika, Binyam Tekeste, Alberto Diez

Figure 11: Frontend

On of the features on this webpage is that allows you too choose from the pictures in the Test Dataset and see how the model classifies that specific picture.

Just by clicking in any of them, allows you to automatically copy it. As a next step, would be to continue on the "Play Ground" field. There you would have the opportunity to paste the picture details that was copied. By clicking "submit", the picture will be loaded and some extra information will be shown. This information includes: "time" (representing on how much time the model makes the prediction), image dimensions on "width" and "height", "confidence" representing the percentage of certainty the model

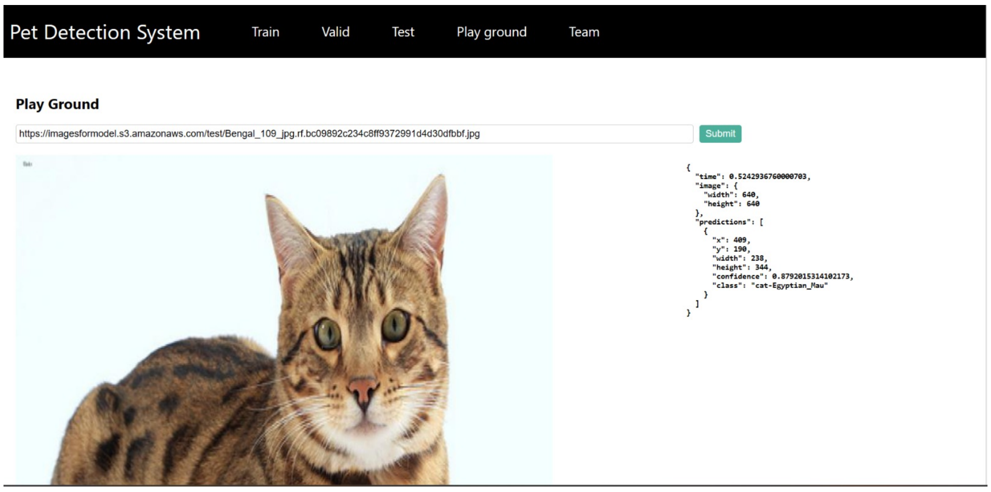


Test

Total number of images: 369

Figure 12: Test Dataset

is giving for the prediction made, ”class” giving the classification result as a cat or dog. An example of this is illustrated in the following figure, where we have the probability of approximately 0.87, that a cat is being detected in the picture.



Play Ground

https://imagesformodel.s3.amazonaws.com/test/Bengal_109_jpg.rf.bc09892c234c8ff9372991d4d30dfbbf.jpg

```
{
  "time": 0.5242936760000703,
  "image": {
    "width": 640,
    "height": 640
  },
  "predictions": [
    {
      "x": 409,
      "y": 190,
      "width": 238,
      "height": 344,
      "confidence": 0.8792015314102173,
      "class": "cat-Egyptian_Mau"
    }
  ]
}
```

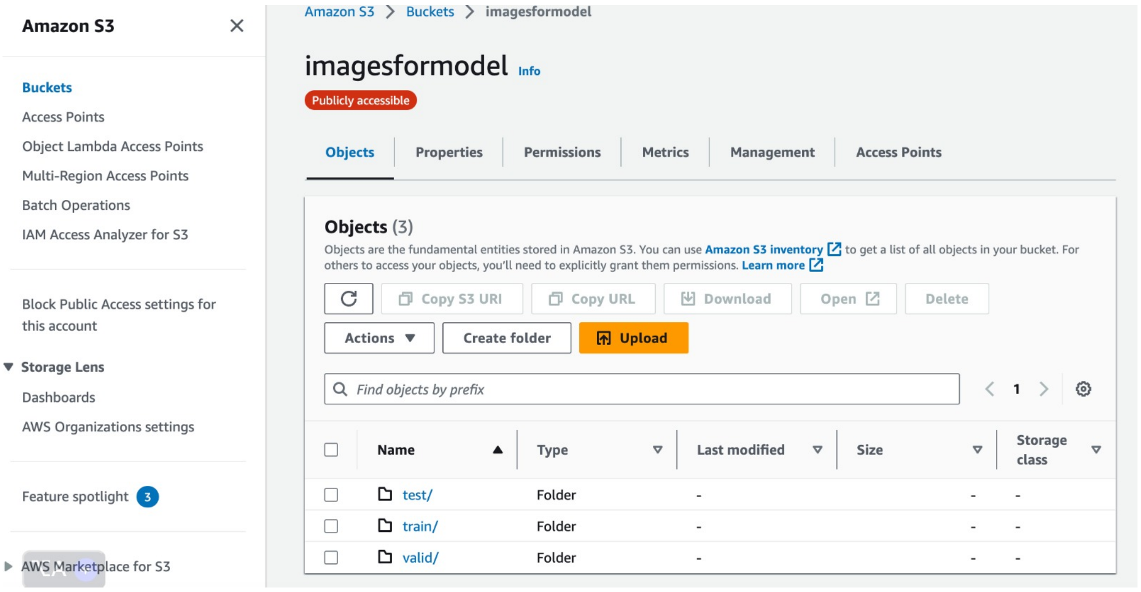
Figure 13: Play Ground

6.1 How are we getting the pictures?

From Roboflow where the dataset of images used to train and test the prediction model was stored, we needed a way to fetch them and display them in our webpage. The option that we choosed to achieve this was to create a AWS S3 instance, store the images there and then display them as we already showed in previous pictures.

6.2 AWS S3 instance

AWS offers many different services and opportunities of storage or computation. In our case the Simple Storage Service - S3 functionality was a better fit, since any kind of object

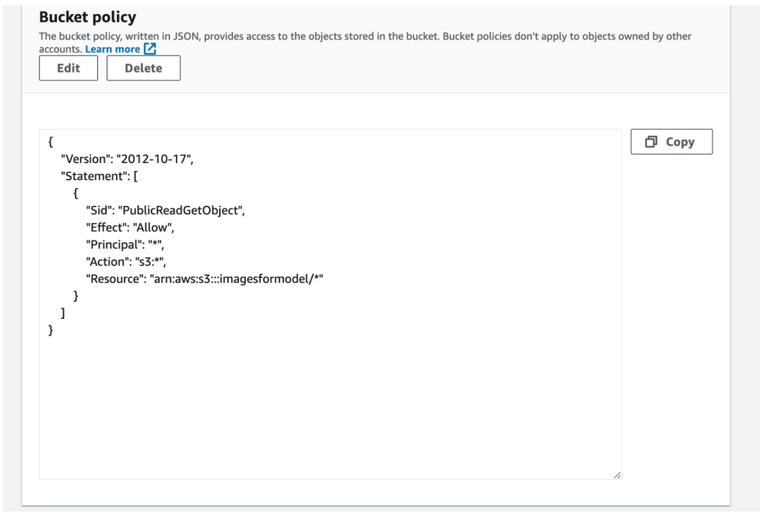


Objects (3)

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Name	Type	Last modified	Size	Storage class
test/	Folder	-	-	-
train/	Folder	-	-	-
valid/	Folder	-	-	-

Figure 14: S3 Bucket



Bucket policy

The bucket policy, written in JSON, provides access to the objects stored in the bucket. Bucket policies don't apply to objects owned by other accounts. Learn more

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "PublicReadGetObject",
         "Effect": "Allow",
         "Principal": "*",
         "Action": "s3:*",
         "Resource": "arn:aws:s3:::imagesformodel/*"
      }
   ]
}
```

Figure 15: S3 Policies

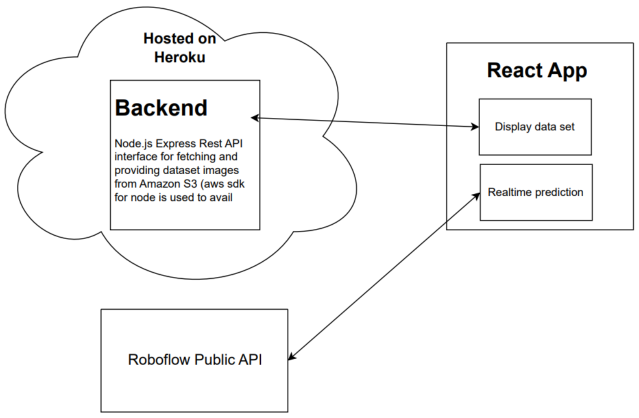

Hosted on Heroku
Backend
Node.js Express Rest API interface for fetching and providing dataset images from Amazon S3 (aws sdk for node is used to avail
React App
Display data set
Realtime prediction
Roboflow Public API

Figure 16: Diagram

can be stored and organised there, including images which was our need. Scalability, good performance, accessibility and security is offered by this service. [3]

To upload our images in the S3 instance we firstly needed to create a Bucket. Bucket is like a container for storing different objects, and an S3 can support up to 100 Buckets. Having your objects organized in smaller quantities makes management easier, and also the access of the data or the deletion of those after finishing your tasks.

Figure 14 will give a sight of our Bucket, called "imagesformodel", where folders of "train", "valid", "test" are stored as objects. This bucket is publicly accessible. [4]

Also a representation of the policies for this bucket can be found as displayed below, on Figure 15, while figure 16 represents a general understanding of how we got the images.

7 Conclusion

To conclude this work representation we can say that it was a very challenging process, where we learned a lot, tried new things and got a valuable hands on experience.

Our model is working predicting whether different objects are cats or dogs, the Kubernetes Cluster is working and we already have a webpage showing the dataset of images that we used to train validate and test our model.

Our next challenges and improvements would be to connect our Raspberry Pi to the AWS S3 istance, in order to show the live images being captured also in our webpage; train our model to detect new pets, organize for future development.

Github link: Our GitHub Link

References

[1] Madhan Kumar Srinivasan, K. Sarukesi, Paul Rodrigues, M. Sai Manoj, and P. Revathy. 2012. State-of-the-art cloud computing security taxonomies: a classification of security challenges in the present cloud computing environment. In Proceedings of the International Conference on Advances in Computing, Communications and Informatics (ICACCI '12). Association for Computing Machinery, New York, NY, USA, 470–476. https://doi.org/10.1145/2345396.2345474

[2] Redmon, J., Divvala, S.K., Girshick, R.B., & Farhadi, A. (2015). You Only Look Once: Unified, Real-Time Object Detection. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 779-788.

[3] Amazon Simple Storage Service

[4] Amazon Simple Storage Service: Bucket Overview

[5] Install Raspberry Pi OS using Raspberry Pi Imager

[6] Install K3S on master node and worker nodes

[7] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E. Z., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., & Chintala, S. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. CoRR, abs/1912.01703. Retrieved from http://arxiv.org/abs/1912.01703

[8] Dwyer, B. (2021). Oxford-IIIT Pet Dataset. Roboflow. Retrieved from https://universe.roboflow.com/brad-dwyer/oxford-pets/dataset/2

[9] Jocher, G., Chaurasia, A., & Qiu, J. (2023). Ultralytics YOLOv8 (Version 8.0.0) [Computer software]. https://github.com/ultralytics/ultralytics

[10] Redmon, J., Divvala, S. K., Girshick, R. B., & Farhadi, A. (2015). You Only Look Once: Unified, Real-Time Object Detection. CoRR, abs/1506.02640. Retrieved from http://arxiv.org/abs/1506.02640