FRANKFURT UNIVERSITY OF APPLIED SCIENCES

Rat Detection Project

Presented by
Group 5
Amandeep Singh Chhatwal (1386396)
Shounak Ozarkar (1386299)
Rajapreethi Rajendran (1396483)
Mansi Patil (1388967)
Sneha Srinivasa (1400476)
Mrinal Tyagi(1383988)

Under the Guidance
of
Prof. Dr. Christian Baun

Agenda

1.Introduction
2.Project Management
3.Architecture
4.Cluster Setup
5.Model Training
6.Edge Node Setup
7.UI development
8.Integration
9.Challenges
10.Summary

1
Introduction

Introduction

Cloud Computing offers wide range of services

Goal to develop an edge computing solution for the automatic detection of rats

Provides scalability, accessibility and cost-effectiveness

Eliminates the need for expensive hardware and storage solutions

2
Project Management

Project Management

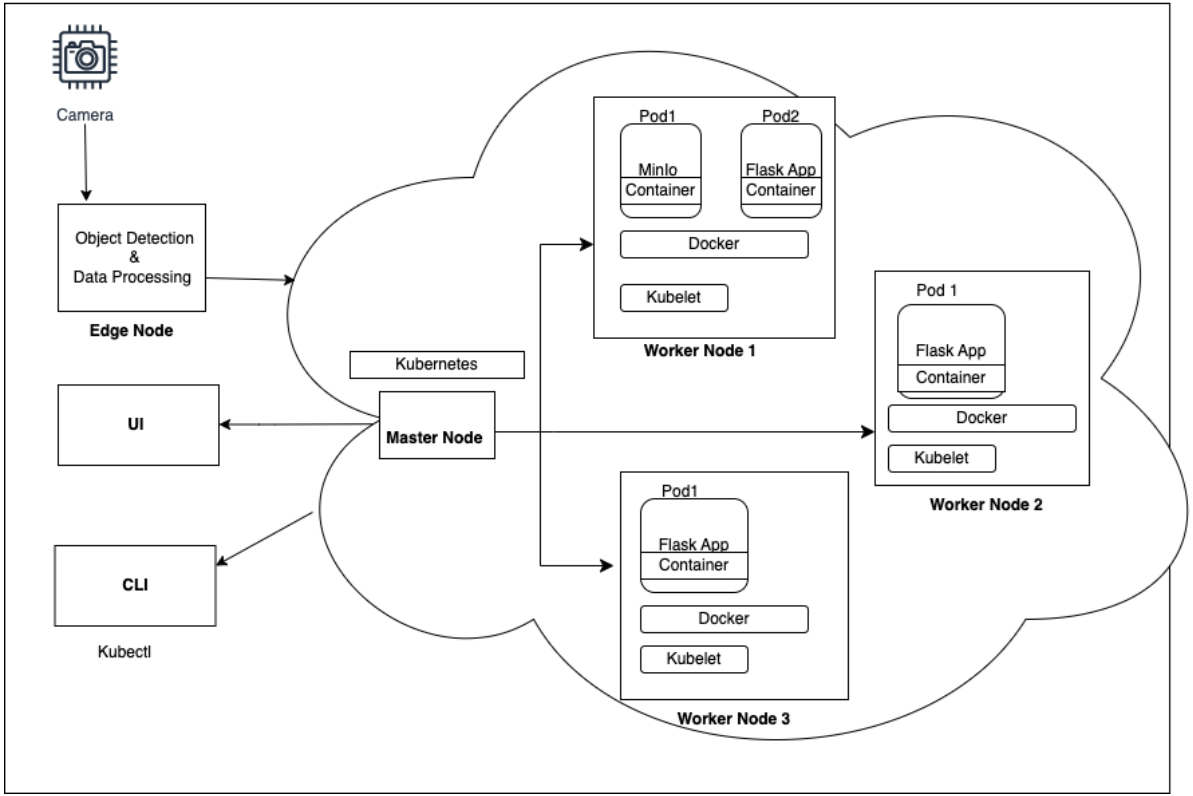
Task Description	Person Responsible
Cluster Setup/maintainability/performance	Amandeep
Minio setup and deployment	Amandeep
Model Training	Shounak/Mrinal
Edge Node & Camera module Setup	Rajapreethi
Rat detection in Edge Node	Rajapreethi
UI development	Mansi/Sneha/Shounak
Integration of individual parts	Rajapreethi/Shounak/Amandeep

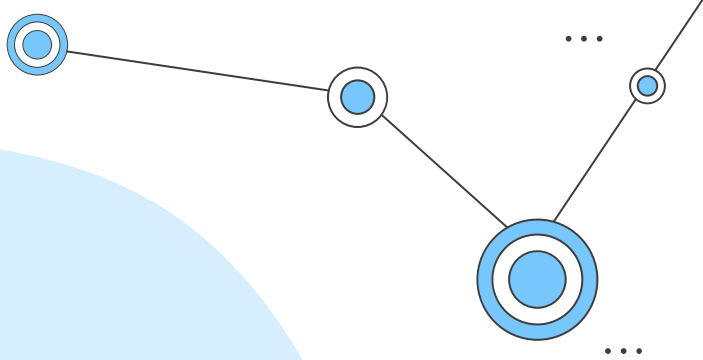
Trello https://trello.com/b/JskLvlXm/cloud-computing

Github https://github.com/AmandeepChhatwal/FindTheRats/tree/main/ObjectDetection

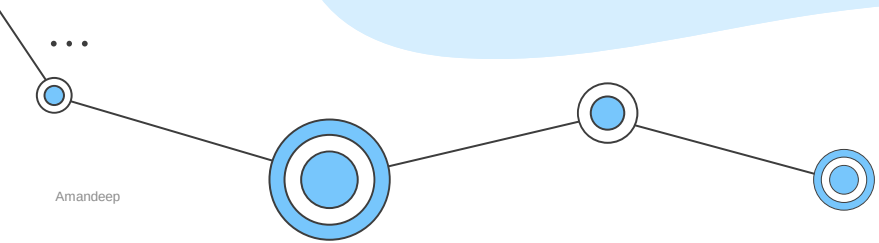
3
Architecture

The Diagram


Camera
Object Detection
&
Data Processing
Edge Node
UI
CLI
Kubectl
Kubernetes
Master Node
Pod1
Pod2
MinIo
Container
Flask App
Container
Docker
Kubelet
Worker Node 1
Pod 1
Flask App
Container
Docker
Kubelet
Worker Node 2
Pod1
Flask App
Container
Docker
Kubelet
Worker Node 3



4
Cluster Setup



Initial Configurations

- Headless version of Debian 64 bit OS (bullseye)
- Using Fritz box for DHCP services
- Setting Static IPs
- Enabling the cgroup flags for k3s deployment

Docker as Runtime

- Installing docker runtime on each node
- Setting the k3s on the master using docker as runtime instead of containerd
- Setting the worker nodes
- Enabling the kubectl for ease of use

```
PS C:\Users\chhat> kubectl get nodes --output=wide
NAME     STATUS   ROLES                  AGE   VERSION        INTERNAL-IP      EXTERNAL-IP   OS-IMAGE                         KERNEL-VERSION   CONTAINER-RUNTIME
node3    Ready    <none>                 16d   v1.25.5+k3s2   192.168.178.43   <none>        Debian GNU/Linux 11 (bullseye)   5.15.84-v8+      docker://20.10.21
node2    Ready    <none>                 16d   v1.25.5+k3s2   192.168.178.42   <none>        Debian GNU/Linux 11 (bullseye)   5.15.84-v8+      docker://20.10.21
node1    Ready    <none>                 16d   v1.25.5+k3s2   192.168.178.41   <none>        Debian GNU/Linux 11 (bullseye)   5.15.84-v8+      docker://20.10.21
master   Ready    control-plane,master   16d   v1.25.5+k3s2   192.168.178.40   <none>        Debian GNU/Linux 11 (bullseye)   5.15.84-v8+      docker://20.10.21
PS C:\Users\chhat> G
```

MinIO deployment

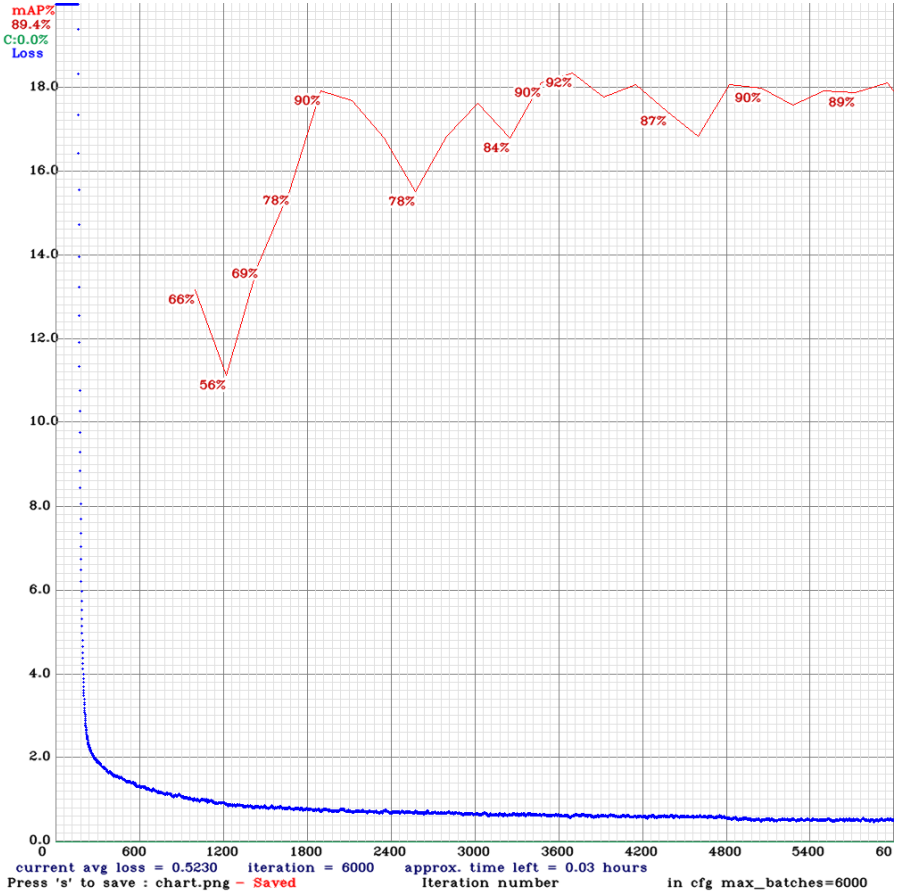
- Using MinIO for resilience, scalability features
- Efficient for Object detection software
- Availability of RESTful API suited for Kubernetes clusters
- Single Node Single Drive v/s Multi Node Multi Drive
- Investigated use of Hazelcast for synchronization
- Enabled as a service for UI Edge Node usage
- Use of Persistent Volume claims

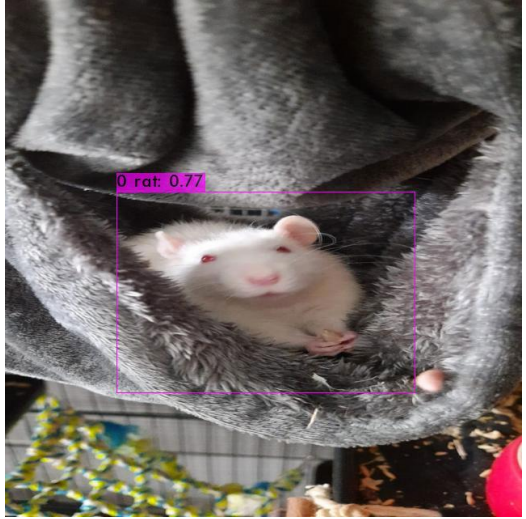
```
PS C:\Users\chhat> kubectl get services
NAME                TYPE           CLUSTER-IP     EXTERNAL-IP                                  PORT(S)                       AGE
kubernetes          ClusterIP      10.43.0.1      <none>                                       443/TCP                       6d4h
flask-image-gallery LoadBalancer   10.43.51.91    192.168.178.41,192.168.178.42,192.168.178.43 5000:30890/TCP                14m
minio               LoadBalancer   10.43.34.64    192.168.178.41,192.168.178.42                9090:31239/TCP,35585:32729/TCP 4m42s
```

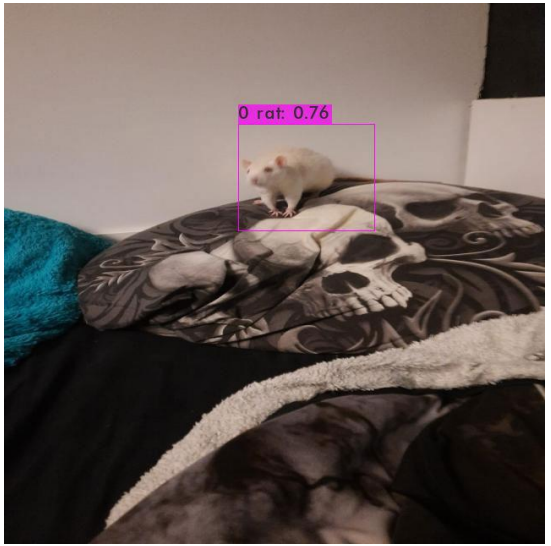
5
Object Detection Model

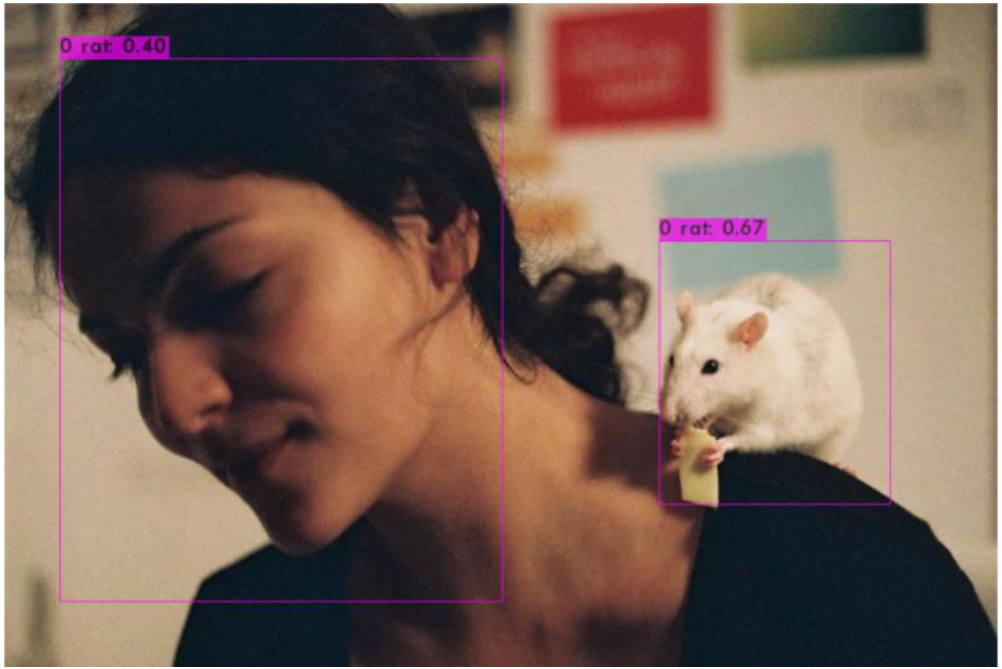
Object Detection using YOLO

Trained model using YOLO v4 tiny on 3000 images


mAP%
89.4%
C:0.0%
Loss
18.0
16.0
14.0
12.0
10.0
8.0
6.0
4.0
2.0
0.0
0
600
1200
1800
2400
3000
3600
4200
4800
5400
60
66%
56%
69%
78%
90%
78%
84%
90%
92%
87%
90%
89%
current avg loss = 0.5230 iteration = 6000 approx. time left = 0.03 hours
Press 's' to save : chart.png - Saved
Iteration number
in cfg max_batches=6000


0 rat: 0.77


0 rat: 0.76


0 rat: 0.40
0 rat: 0.67

6
Edge Node Setup

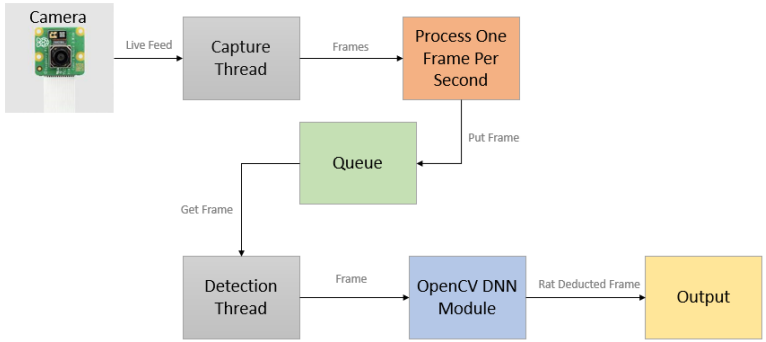
Edge Node In Object Detection

- Main idea – Edge node acts as gatekeeper
- Reduce Latency
- Reduce load on cloud
- Increase scalability
- Efficient deployment of Computer vision algorithms

Initial Configurations

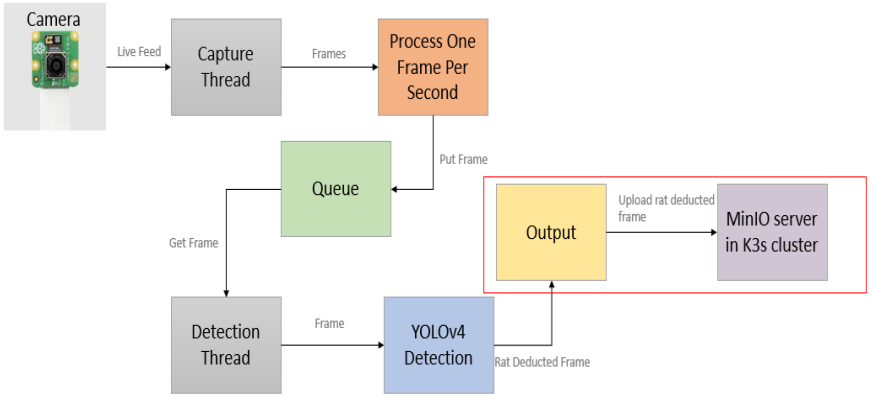
- Desktop version of Debian 64 bit OS (bullseye)
- Camera integrated and configuration change to capture live feed
- Installed Object detection dependencies
  - Python libraries
  - OpenCV 4.7.0
  - MinIO client library
- SSH configuration enabled

Rat Detection In Edge Node



- Multithreading
- OpenCV - Captures live feed
- OpenCV DNN module
- YOLO v4 trained weights, classes, configuration
- Forward propagation

Image Upload to MinIO



- MinIO client object
- Set - MinIO server endpoint, access key, secret key, security
- To store images - MinIO bucket Created
- put_object method

7

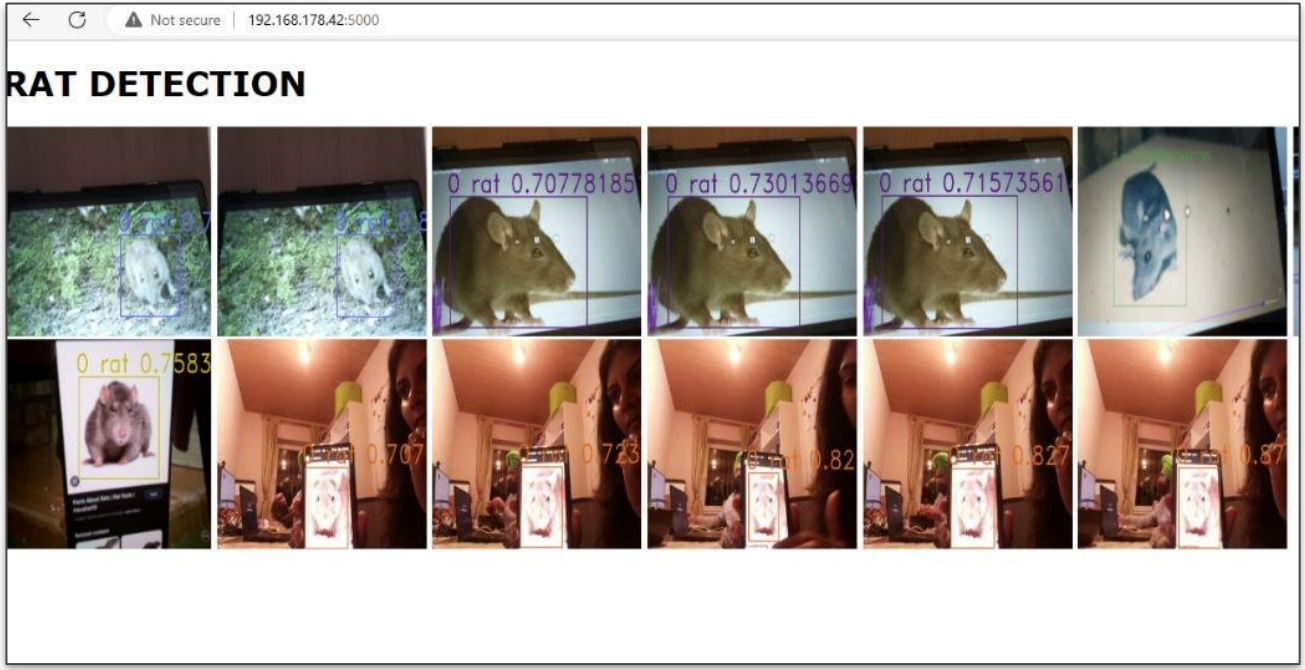
UI Development

Flask Web App

- Lightweight web framework for Python.
- The basic base API is well-designed and well-coordinated.
- Suitable for Small and medium sized projects.

Flask Web App Development

- The UI displays the images the Rats from the MinIO database.
- The images from Database is copied into a folder.
- The latest images should be displayed.
- Deployment on Cluster.




```
PS C:\Users\chhat> kubectl get pods --output=wide
NAME                                READY   STATUS    RESTARTS   AGE     IP           NODE    NOMINATED NODE   READINESS GATES
flask-image-gallery-78c7f6f457-bnvpb   1/1     Running   0          5m36s   10.42.1.61   node1   <none>           <none>
flask-image-gallery-78c7f6f457-ff449   1/1     Running   0          5m36s   10.42.2.79   node2   <none>           <none>
flask-image-gallery-78c7f6f457-rffv7   1/1     Running   0          5m36s   10.42.3.73   node3   <none>           <none>
```

8
Integration

Integration

- UI deployment in K3s Cluster
- Detected image upload from Edge node to MinIO Server hosted in K3s cluster

9
Challenges

Challenges

- Challenges
  - Hardware sharing - parallel working on it
  - Reliability of the hardware in general
  - MinIO Multi Node installations
  - Configuration issues with OpenCV live feed
  - Object detection - performance issue
  - Started front end with prometheus
  - Moving UI from React to Python

...

10
Summary

Summary

- Scalable Object Detection with cloud infrastructure
- Challenges - data privacy & security issue, regulatory Compliance
- Combination of technologies - flexible and powerful solution for object detection.

...