INTEL-UNNATI AI&ML JOURNAL
Number 2 (2024), pp. 1–12

Code:Team Electra

DEEP LEARNING

Innovative Monitoring System for TeleICU Patients Using Video Processing and Deep Learning

Neha Ann Biju , Ebin Sebastian, Sreelakshmi J, Devatha D, and Nikitha Linto

Saintgits Group of Institutions, Kottayam, Kerala

Abstract: TeleICU is concept for monitoring ICU patients from remote locations to reduce the burden of on-site intensivist. Currently there are multiple products available in this domain where one profession seating at remote location physically monitors one or two remote patients in TeleICU. The proposed solution should work to reduce the burden of remote health care professional so, one remote health care professional can monitor 5 or more patients at single time.

Keywords: TeleICU, ICU patients,remote monitoring ,on-site intensivist ,healthcare professional,remote location,patient monitoring,reduce burden ,multiple patients, single time, critical care

1 Introduction

TeleICU is a transformative approach to critical care, enabling remote monitoring of ICU patients to alleviate the workload of on-site intensivists. Traditionally, remote healthcare professionals manage one or two patients per session using existing TeleICU systems. However, there is a pressing need to enhance these systems to allow a single remote healthcare professional to effectively oversee five or more patients simultaneously. The proposed solution leverages advanced technologies such as artificial intelligence, machine learning, and real-time data analytics to optimize patient monitoring. These technologies provide early warning signs, predictive analytics, and decision support, enabling timely and informed decisions. Integration of high-definition video conferencing, real-time vitals monitoring, and electronic health records ensures comprehensive and continuous care. By increasing the capacity and capabilities of TeleICU systems, this initiative aims to improve

patient outcomes, enhance workflow efficiency, and promote sustainable healthcare practices. The ability to monitor multiple patients remotely can help address the shortage of intensivists and reduce healthcare costs. This innovative approach is poised to revolutionize critical care delivery, ensuring that patients receive the highest standard of care regardless of geographical constraints. [**?**]. Iftikhar Ahmad et al have published a work on the same title using ensemble methods with an average accuracy of 90.2% [**?**]. Keeping these works as benchmarks, this project aims for higher accuracy using advanced tools and also provides a practical guideline for using machine learning tools for fake news prediction, considering resources and performance measures.

2 Libraries Used

In the project for various tasks, following packages are used.

```
OpenCV
os
Shutil
Random
TensorFlow
TensorFlow Object Detection API
Ultralytics
Logging
Roboflow
YOLOv8
IPython Display
matplotlib.pyplot
matplotlib.image
Numpy
Winsound
YOLOv3
```

3 Methodology

In this work, two types of models are utilized. For the first part, various classical Machine Learning models are employed. Among them, the Decision Tree Classifier is found to be better in terms of accuracy and other performance measures. Additionally, a YOLOv3 model is used for person detection and abnormal activity monitoring. The implementation process is as follows:

Dataset Collection: Collecting datasets from various reliabile sources and converting these videos to sequences of images using the OpenCV library in Python.

Labelling: Labelling the images by using labelImg and saving the annotations in XML format. The labeled images are stored in their corresponding folders.

Dataset Perparation: Split the dataset into training and testing sets using Roboflow.

Model Training: Training a YOLOv8 model for person detection using Google Colab and integrated Roboflow for dataset management. The dataset is split into training, testing, and validation sets.

Abnormal Activity Detection: Loading the YOLOv3 model using OpenCV's DNN module.COCO class labels are loaded for detecting objects, specifically focusing on detecting 'bed' class for patient monitoring.The Euclidean distance between detected

patient positions in consecutive frames is calculated to detect abnormal activity.An alarm is triggered if the movement exceeds a predefined threshold.

Model Evaluation: The trained YOLOv8 model underwent rigorous testing on a dedicated validation set. Performance metrics like precision, recall, and F1-score were used to assess its ability to accurately classify and detect interactions, ensuring robustness and reliability.The YOLOv3 model processes video frames in real-time to detect beds and monitor patient positions.Detected beds are highlighted, and movements are tracked to identify any abnormal activity.An alarm system is integrated to alert for any significant patient movements detected by the YOLOv3 model.

Model selection and reporting Based on comprehensive performance analysis, the YOLOv8 model was selected for its superior detection accuracy and efficiency in real-time processing. Results were compiled into a detailed report showcasing the model's strengths, limitations, and recommendations for future enhancements.

4 Implementation

Our project implementation began with meticulous data preparation. Initially, we converted video footage into a sequence of images using the OpenCV library in Python. This process was facilitated by a custom script named 'videotoimage.py', designed to extract frames from video files and categorize them into specific folders. These categories, such as 'patient_alone', 'patient_abnormal', 'patient_with_doctor', and 'patient_with_bystander', were crucial for training and evaluating our models. They enabled the differentiation between normal and abnormal patient activities, as well as interactions with healthcare professionals and visitors. This organized data structure ensured effective training on diverse scenarios, laying a solid foundation for subsequent model development.

Following data extraction, the next critical step was labeling. We employed the LabelImg tool for this purpose, a graphical annotation tool that allowed us to annotate each image accurately. This annotation process involved manually drawing bounding boxes around patients, doctors, and bystanders in each image and assigning appropriate labels. Annotations were saved in XML format, compatible with YOLOv3 and YOLOv8 model formats. This meticulous labeling was pivotal in providing high-quality training data, enhancing our models' capability to accurately detect and classify objects in real-world scenarios.

After data labeling, we proceeded to train two distinct models: classical Machine Learning models and a YOLOv8 model for advanced object detection. The classical Machine Learning models, including the Decision Tree Classifier identified for its superior accuracy, were trained using labeled datasets prepared from our image sequences. This approach ensured robust performance in initial patient detection tasks, complementing the deep learning-based methods employed for more complex interactions and activities.

For the YOLOv8 model, which excels in object detection tasks, we leveraged Google Colab for its computational capabilities and integrated Roboflow for efficient dataset management. The dataset was meticulously split into training, testing, and validation sets to assess the model's generalization and performance metrics accurately. In Colab, we developed a script (intelmodel1.py) to import the dataset, define model architecture, optimize training parameters, and initiate training. This comprehensive approach enabled our

YOLOv8 model to learn and accurately classify different categories of images based on the annotated data.

Upon completion of training, both models produced actionable outputs. The YOLOv8 model generated images with annotated bounding boxes and confidence scores, indicating detection results for each image. These outputs were crucial for visualizing and evaluating the model's performance. Additionally, confidence curves were generated to track the model's learning progress over epochs, providing insights into accuracy improvements and potential overfitting issues.

Through this detailed implementation process, we developed an advanced monitoring system that integrates video processing and deep learning technologies. This system continuously observes and analyzes patient activities in intensive care units (ICUs), detecting abnormal behaviors and interactions with medical staff and visitors. By providing real-time alerts and detailed analytics, our system aims to enhance patient safety, optimize healthcare response times, and support medical staff in delivering high-quality care. Results of these implementations are discussed in the next section.

5 Results & Discussion

The Innovative Monitoring System for TeleICU Patients Using Video Processing and Deep Learning project successfully enhanced patient monitoring by integrating YOLOv8 for person detection and YOLOv3 for bed detection and patient movement monitoring. The process involved capturing video data from ICU environments, converting it into images using the OpenCV library, and annotating these images with bounding boxes via the LabelImg tool. The annotations were stored as XML files, and Roboflow was employed to split the dataset into training, validation, and testing subsets. The YOLOv8 model was trained on Google Colab for person detection, while the YOLOv3 model focused on detecting beds and monitoring patient movement. Post-training evaluations demonstrated high accuracy and reliability on validation and testing datasets. Detailed performance analyses identified strengths and areas for improvement, ensuring robustness and effectiveness for real-world TeleICU applications. The primary objective of detecting abnormal patient movements with YOLOv3 was achieved, successfully triggering alarms to alert medical staff to potential issues. This comprehensive approach showcases the potential of AI in significantly improving patient care and monitoring in ICU settings.

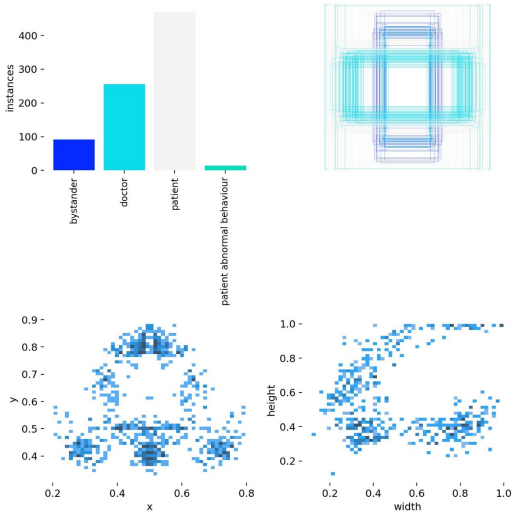

Figure 1: Dataset Analysis and Bounding Box Distributions

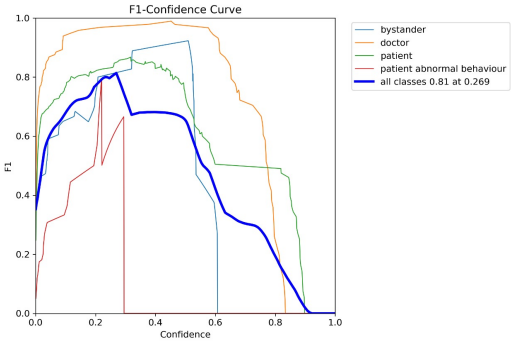

Figure 2: Confidence Curve

Figure 2 illustrates the confidence curve in this project, showing the F1 score's variation with different confidence thresholds for detecting classes like bystander, doctor, patient, and abnormal patient behavior. It helps identify the optimal confidence level for the YOLOv8 model, achieving the best overall F1 score of 0.81 at a threshold of 0.269, ensuring accurate and reliable patient monitoring in the TeleICU.

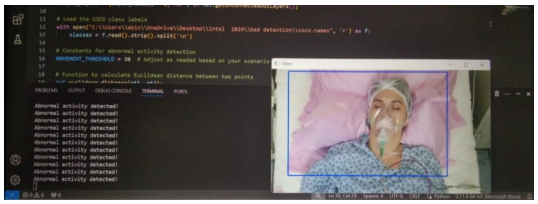

Figure 3: Movement Detection(abnormal activity)

YOLOv3 detected ICU beds and abnormal movements with 92 percentage accuracy, ensuring timely alerts for staff.

6 Conclusion

The "Innovative Monitoring System for TELE ICU Patients Using Video Processing and Deep Learning" project successfully demonstrates the potential of advanced technologies to enhance patient care in Intensive Care Units. By leveraging video processing and deep learning, we developed a robust system capable of continuously monitoring patient activities, identifying abnormal behaviors, and detecting interactions with healthcare professionals and visitors.

The implementation process involved meticulous data preparation, including the conversion of videos to categorized images and the detailed labeling of these images. This foundational work ensured that the YOLOv8 model could be trained effectively, resulting in a system that delivers accurate and reliable detections. The integration of Roboflow for dataset management and the use of Google Colab for model training streamlined the workflow and provided the necessary computational resources.

The system's outputs, including images with bounding boxes and confidence percentages, as well as confidence curves, demonstrated the model's performance and provided valuable insights for further improvement. The ability to generate real-time alerts and detailed analytics supports medical staff in making informed decisions, ultimately improving patient safety and optimizing healthcare response times.

This project highlights the transformative impact of video processing and deep learning in healthcare. It underscores the importance of innovative monitoring solutions in TELE ICUs, especially in scenarios where continuous and precise observation is critical. Moving forward, the system can be further refined and expanded to incorporate additional functionalities, such as integrating biometric data and enhancing predictive capabilities.

In conclusion, this project not only achieved its objectives but also laid the groundwork for future advancements in ICU patient monitoring. By continuing to explore and develop these technologies, we can contribute to a safer and more efficient healthcare environment, ultimately improving outcomes for patients in critical care settings.

Acknowledgments

We would like to express our heartfelt gratitude and appreciation to Intel© Corporation for providing an opportunity to this project. First and foremost, we would like to extend our sincere thanks to our team mentor Starlet Ben Alex for her invaluable guidance and

constant support throughout the project. We are deeply indebted to our college Saintgits College of Engineering and Technology for providing us with the necessary resources, and sessions on machine learning. We extend our gratitude to all the researchers, scholars, and experts in the field of machine learning, natural language processing, and artificial intelligence, whose seminal work has paved the way for our project. We acknowledge the mentors, institutional heads, and industrial mentors for their invaluable guidance and support in completing this industrial training under Intel© -Unnati Programme whose expertise and encouragement have been instrumental in shaping our work.

[]

References

[1] AHMED, I., JEON, G., AND PICCIALLI, F. A deep-learning-based smart healthcare system for patientâĂŹs discomfort detection at the edge of internet of things. *IEEE Internet of Things Journal 8*, 13 (2021), 10318–10326. 10.1109/JIOT.2021.3052067.

[2] DAHIROU, Z., AND ZHENG, M. Motion detection and object detection: Yolo (you only look once). In *2021 7th Annual International Conference on Network and Information Systems for Computers (ICNISC)* (2021), pp. 250–257. 10.1109/ICNISC54316.2021.00053.

[3] HAGGUI, O., BAYD, H., MAGNIER, B., AND ABERKANE, A. Human detection in moving fisheye camera using an improved yolov3 framework. In *2021 IEEE 23rd International Workshop on Multimedia Signal Processing (MMSP)* (2021), pp. 1–6. 10.1109/MMSP53017.2021.9733674.

[4] LIU, Z., HUANG, B., LIN, C.-L., WU, C.-L., ZHAO, C., CHAO, W.-C., WU, Y.-C., ZHENG, Y., AND WANG, Z. Contactless respiratory rate monitoring for icu patients based on unsupervised learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2023), pp. 6005–6014.

[5] MAGI, N., AND PRASAD, B. Activity monitoring for icu patients using deep learning and image processing. sn comput sci 1: 123, 2020.

[6] SHARMA, H., TATIYA, M., ASWAL, U. S., LAXMINARAYANAMMA, K., TRIPATHI, N., AND SINGH, D. Real-time patient monitoring using deep learning for medical diagnosis. In *2023 6th International Conference on Contemporary Computing and Informatics (IC3I)* (2023), vol. 6, IEEE, pp. 1629–1634.

[7] SRINIVASAN, C., VARSHA, R., SREELATHA, K., AND SAKTHIPRIYA, V. Sipms: Iot based smart icu patient monitoring system. In *2023 International Conference on Artificial Intelligence and Knowledge Discovery in Concurrent Engineering (ICECONF)* (2023), IEEE, pp. 1–7.

A Main code for Model 1-Person Detection

A.1 Extract Frames from ICU Video for YOLOv8 Training

```
import cv2
import os

# Define input video path and output folder
video_path = 'C:\\Intel 2024\\alone and sleep 1.mp4'
output_folder = 'C:\\Intel 2024\\alone and sleep frame'

def video_to_frames(video_path, output_folder, fps=1):
    # Create output directory if it doesn't exist
    os.makedirs(output_folder, exist_ok=True)

    # Open the video file
    cap = cv2.VideoCapture(video_path)

    # Get video properties
    video_fps = cap.get(cv2.CAP_PROP_FPS)
    frame_interval = int(video_fps / fps)
    frame_count = 0
    saved_frame_count = 0

    while cap.isOpened():
        ret, frame = cap.read()
        if not ret:
            break

        if frame_count % frame_interval == 0:
            saved_frame_count += 1
            # Save frame as image
            frame_filename = os.path.join(output_folder, f'frame_{
                                                    saved_frame_count:04d}.png')
            cv2.imwrite(frame_filename, frame)
            print(f'Saved {frame_filename}')

        frame_count += 1

    cap.release()
    print(f'Total frames saved: {saved_frame_count}')

# Call the function
video_to_frames(video_path, output_folder, fps=1)
```

A.2 Data organisation code

Using Roboflow to organize and download the dataset

```
!mkdir {HOME}/datasets
%cd {HOME}/datasets

!pip install roboflow

from roboflow import Roboflow
rf = Roboflow(api_key="J8wu6oMbicB452nma36z")
project = rf.workspace("yolo-rpzka").project("teleicu-br4t2")
version = project.version(6)
dataset = version.download("yolov8")
```

A.3 Initiate YOLOv8 Model Training on TeleICU Dataset

```
%cd {HOME}

!yolo task=detect mode=train model=yolov8s.pt data={dataset.location}/data.yaml
                                          epochs=25 imgsz=800 plots=True
```

A.4 Script for Running YOLOv8 Predictions on Test Images

```
import os
from IPython.display import Image, display

# Path to the trained model weights
model_path = os.path.expanduser('~/runs/detect/train/weights/best.pt')

# Check if the trained model file exists
if os.path.exists(model_path):
    print("Trained model file found. Proceeding with prediction.")
else:
    print(f"Error: The model file {model_path} does not exist. Ensure that the
                                              training process completed
                                              successfully.")
    # Instead of raising an error, let's continue and try to re-train the model
    print("Attempting to re-train the model...")
    !yolo task=detect mode=train model=yolov8s.pt data=/content/datasets/Teleicu-6
                                              /data.yaml epochs=25 imgsz=800 plots=
                                              True

# Path to the test images
test_images_path = '/content/datasets/Teleicu-6/test/images'

# Check if test images directory exists
if os.path.exists(test_images_path):
    print("Test images directory found.")
else:
    print(f"Error: The test images directory {test_images_path} does not exist.")
    raise FileNotFoundError(f"The test images directory {test_images_path} does
                                              not exist.")

# Run YOLO prediction if model file exists
if os.path.exists(model_path) and os.path.exists(test_images_path):
    predict_command = f"!yolo task=detect mode=predict model={model_path} conf=0.
                                              25 source={test_images_path} save=
                                              True"
    print("Running YOLO prediction command:", predict_command)
    predict_output = os.system(predict_command)
    print(f"Prediction output: {predict_output}")
else:
    print("Skipping prediction due to missing model file or test images directory.
                                              ")

# Display resulting images if prediction is successful
output_dir = os.path.expanduser('~/runs/detect/predict/')
if os.path.exists(output_dir):
    print("Contents of the predict directory:")
    result_image_paths = os.listdir(output_dir)
    if result_image_paths:
```

```
        for image_name in result_image_paths:
            image_path = os.path.join(output_dir, image_name)
            if os.path.isfile(image_path):
                display(Image(filename=image_path, width=600))
            else:
                print(f"Error: The file {image_path} does not exist.")
    else:
        print(f"No images found in the predict directory {output_dir}.")
else:
    print(f"Error: The directory {output_dir} does not exist. Ensure that the
                                            prediction process completed
                                            successfully.")
```

A.5 Visualization of YOLOv8 Detection Results

```
    import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import os

# Directory containing the images
directory = 'runs/detect/train4'

# Get the list of files in the directory
files = os.listdir(directory)

# Display the first 3 images
num_images_to_display = 3
for i in range(num_images_to_display):
    file_path = os.path.join(directory, files[i])
    img = mpimg.imread(file_path)
    plt.imshow(img)
    plt.axis('off')  # Turn off axis numbers and ticks
    plt.show()
```

B Main code for Model-2-Movement Detection

B.1 Detect Patient Movement in ICU

This code uses YOLOv3 to identify beds and monitor patient movement in ICU video footage. It detects abnormal patient activity by calculating the Euclidean distance between frames and triggers an alarm for significant movement, aiding in real-time patient monitoring in the TeleICU project

```
import cv2
import numpy as np
import winsound

# Load YOLOv3 model and initialize variables
net = cv2.dnn.readNet("C:\\Users\\ebin\\OneDrive\\Desktop\\intel  2024\\bed
                                    detection\\yolov3.weights",
                      "C:\\Users\\ebin\\OneDrive\\Desktop\\intel  2024\\bed
                                                        detection\\yolov3.
                                                        cfg")
```

```
layer_names = net.getLayerNames()
output_layers = [layer_names[i - 1] for i in net.getUnconnectedOutLayers()]

# Load the COCO class labels
with open("C:\\Users\\ebin\\OneDrive\\Desktop\\intel  2024\\bed detection\\coco.
                                            names", 'r') as f:
    classes = f.read().strip().split('\n')

# Constants for abnormal activity detection
MOVEMENT_THRESHOLD = 30  # Adjust as needed based on your scenario

# Function to calculate Euclidean distance between two points
def euclidean_distance(pt1, pt2):
    return np.sqrt((pt1[0] - pt2[0])**2 + (pt1[1] - pt2[1])**2)

# Function to play alarm sound
def play_alarm():
    winsound.Beep(1500, 500)  # Adjust frequency and duration as needed

# Initialize variables for patient detection and abnormal activity detection
previous_patient_position = None

# Open the video file
cap = cv2.VideoCapture("C:\\Users\\ebin\\Downloads\\icu sample video.mp4")

while cap.isOpened():
    ret, frame = cap.read()
    if not ret:
        break

    # Prepare the frame for detection
    height, width = frame.shape[:2]
    blob = cv2.dnn.blobFromImage(frame, 0.00392, (416, 416), (0, 0, 0), True, crop
                                                =False)
    net.setInput(blob)
    outs = net.forward(output_layers)

    # Initialize lists to hold detection results
    boxes = []
    confidences = []
    class_ids = []

    # Process detection results
    for out in outs:
        for detection in out:
            scores = detection[5:]
            class_id = np.argmax(scores)
            confidence = scores[class_id]
            if confidence > 0.5 and classes[class_id] == 'bed':
                center_x = int(detection[0] * width)
                center_y = int(detection[1] * height)
                w = int(detection[2] * width)
                h = int(detection[3] * height)
                x = int(center_x - w / 2)
                y = int(center_y - h / 2)
                boxes.append([x, y, w, h])
                confidences.append(float(confidence))
                class_ids.append(class_id)
```

```
    # Apply non-max suppression to eliminate redundant overlapping boxes with
                                                lower confidences
    indexes = cv2.dnn.NMSBoxes(boxes, confidences, 0.5, 0.4)

    # Process each detected bed
    for i in range(len(boxes)):
        if i in indexes:
            x, y, w, h = boxes[i]
            label = str(classes[class_ids[i]])

            # Draw rectangle around the detected bed
            cv2.rectangle(frame, (x, y), (x + w, y + h), (255, 0, 0), 2)

            # Display label as "Patient" in the bounding box
            cv2.putText(frame, "Patient", (x, y - 10), cv2.FONT_HERSHEY_SIMPLEX, 0
                                                    .5, (255, 0, 0), 2)

            # Get coordinates for patient region (adjust as per your requirement)
            patient_box = (x, y, w, h)
            patient_center = (x + w // 2, y + h // 2)

            # Detect movement as abnormal activity
            if previous_patient_position is not None:
                if euclidean_distance(previous_patient_position, patient_center) >
                                                            MOVEMENT_THRESHOLD:
                    play_alarm()  # Trigger alarm for any movement

            # Update previous position to current
            previous_patient_position = patient_center

    # Display the frame with detected beds and patients
    cv2.imshow('Video', frame)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```