A PROJECT REPORT ON

"INNOVATIVE MONITORING SYSTEM FOR TELE ICU PATIENTS USING VIDEO PROCESSING AND DEEP LEARNING"

Submitted to

INTEL Unnati Industrial Training Program 2024.

Submitted By

Students of Department: ELECTRONICS AND COMMUNICATION EGINEERING

SHASHANK MARBE	**USN: SG21ECE082**
SHRINATH BIRADAR	**USN: SG21ECE088**
SIDDARAM	**USN:SG21ECE108**
VINAY KUMAR	**USN: SG21ECE093**

Guide name: Dr. JAGADEESH PATIL

INTRODUCTION

The incorporation of modern technology such as video processing and deep learning into telemedicine has changed the landscape of intensive care unit (ICU) patient monitoring. WHAT IS TELE-ICU Tele-Intensive Care, a rapidly growing field in medicine allowing health workers to virtually observe and treat the ills of ICU patients from anywhere in real-time over any geographic threshold. Through the capabilities of video processing and deep learning models this advances to a new frontier aimed at improving patient care within these environments.

Here, was explore the construction of a bespoke and unique monitoring system for Tele-ICU settings. This system, which combines video processing technology that extracts and analyses visual data from live patient feeds with deep neural networks capable of complex learning offers continuous and comprehensive analysis well beyond what traditional methods make possible. With the real-time analysis of signs, movements and other critical indices allows healthcare providers to identify anomalies fast, predict deterioration while there is still time for intervention resulting in better patient outcomes at reduced cost.

This introduction was the prelude to an extensive look at how tele-ICU patient care is being transformed by both video processing and deep learning techniques; reflecting their disruptive potential in terms of scaling up productivity, as well achieving a higher level of responsiveness through advanced remote monitoring helps streamlining overall Tele-Icu operations.

DATASET DESCRIPTION

OPENCV:

Image Data: Image format: OpenCV supports image formats like JPEG, PNG, BMP and so on. Colour Spaces - RGB, Grayscale, HSV (Hue, Saturation, Largunation), etc. Size refers to that images can have different size or dimensions (such as width and height) which is very important because the large-sized image also will be time-consuming based on memory usage.

Channels: The sum of the channels an image has (f.e 1 for grayscale, /3 RGB) to know how it is processed.

Video Data:

Format: OpenCV can use the video files (AVI, MP4 etc.) as a source to be processed.

Frames: Videos are made up of frames and OpenCV can load individual images from the video.

Training Data for ML Models: OpenCV is also conventionally combined with other libraries like TensorFlow or Porch to train deep learning models. The data on which the models are trained can be images or videos with annotations in form of labels for supervised learning tasks.

Data Preprocessing: OpenCV contains several functions for data-preprocessing (Resizing, Cropping, normalization...).

Preprocessing steps: These are basic, continuous or time consuming and can happen before the feeding of data into Machine learning models.

Object Detection and Object Recognition: Object Detection object datasets contain images (or video frames) with bounding boxes around the objects to be recognized. You can use OpenCV to read and pre-process these images from a dataset, e.g., Pascal VOC or Kaggle data set, so that you can train an object detection model by extracting features out of it.

Facial Recognition: Common datasets for facial recognition have images or videos with labelled face landmarks and identification labels. Face detection, landmark detection and facial recognition tools are available.

Camera Calibration: Camera Calibration Datasets, which are collections of images showing calibration patterns (for example chessboard pattern) taken from various angles and positions. These data sets are used by OpenCV to calibrate the cameras and correct deformations in images.

TENSORFLOW:

Image Data:

Formats: JPEG, PNG, BMP etc are supported - Regular image formats today. While the offered dataset is not big, working with it exposes challenge of preprocessing and model architecture choice due to variability in image sizes & aspect ratios.

Image Channels: Images will have multiple channels such as RGB or Grayscale, these can be easily handled in TensorFlow's image processing utilities.

Text Data:

Formats: Text data in plain text, CSV or other formatted files.

Tokenization: Tools to convert words or characters into numerical representations so we can feed them directly to our machine learning models, ready for model training. In case of text, the length of sequences also might vary and need different processing such as padding or truncating.

Structured Data:

Formats: structured data may be in forms like CSV (Comma-Separated Values), TSV (Tab-separated values) or a database. Data columns e.g. numerical, categorical, text-based features (requiring preprocessing steps that you might take).

Audio Data:

Formats: Audio data could be stored in different formats like.wav, mp3 etc.

Here: who cannot process TensorFlow for example to work with sound class, spoken recognition because the AI works requested are audio data of varying duration and a different sampling rate (voice/song/so so)

Video Data:

Formats: Video data formats are MP4, AVI etc and TensorFlow can work with individual frames such as for those tasks where frame level predictions need to be made like action recognition or video classification Video classification involves sequences of frames, which is Temporal Data and required special type models to handle temporal information.

Transfer learning Datasets TensorFlow provides pretrained models on ImageNet, which are commonly used for transfer learning applications. These are large-scale image datasets annotated with object and scene categories.

Custom Datasets: Users can use them to organize their own datasets with either labelled or feature related data for machine learning tasks. TensorFlow APIs and tools for preprocessing as well performing custom pre-processing, efficient data loading, and create batches.

Reinforcement Learning Datasets Datasets are equivalent to reinforcement learning tasks, where an environment is composed of some states, action rewards and possible environments (simulated or in the real-world)

OS:

System Configuration:

Hardware: Datasets which provide the specs of hardware components, including CPU type and count, RAM size, disk size...Operating System Information is generic information about the version of the OS, as well optional field that specifies a kernel and distribution (together with packages).

System Logs:

Event Logs: Information about system events such as startups, shutdowns and critical errors or warnings.

Performance Logs: System performance related metrics such as CPU usage, memory utilization disk input/output operations and network traffic... etc.

User Activity & Interaction: Logs / Cerebral plexus**;** recording user login/logout time, the length of sessions and activities performed while in session Application Usage - Information on apps opened, how they are used and the demand for resources.

Security and Audit Data: Access Logs - Records of accesses from users to files, directories or system resources.

Security Events: Logs linked to security breaches, login tries failed and succeeded, firewall operations etc.

Network Data:

Network Traffic: data sets showing the inbound and outbound traffic, protocols used, Amount of Data etc.

Network Information: Which network interfaces exist on the system, what IP addresses are available for each interface, and so forth.

Filesystem Information:

File Metadata: file and directory attributes like permission, owner, time stamps etc. You can generate an audit of logs indicating files being accessed, edited or deleted which assists in auditing and performance analysis.

Application Specific Data: Applications that auto-create data about their usage: or performance. For example, databases could log queries, transactions and times.

Time-Series Data: Most of the data in OS are time series, as it captures events and changes over a period. These can then be examined for trends, anomalies or used to forecast potentiality improvements.

Benchmarking and performance test:

A dataset for the purpose of benchmarking a certain area, e.g. in terms of running different aspects related to an operating system or any executable code thereon These may be such as a SPEC CPU benchmark, disk I/O benchmarks and the like. Data for the VMM or Container For virtualized environments or containerized applications, a dataset may be something like info on your VMs, containers and the resources that you've allocated etc.

METHODOLOGY

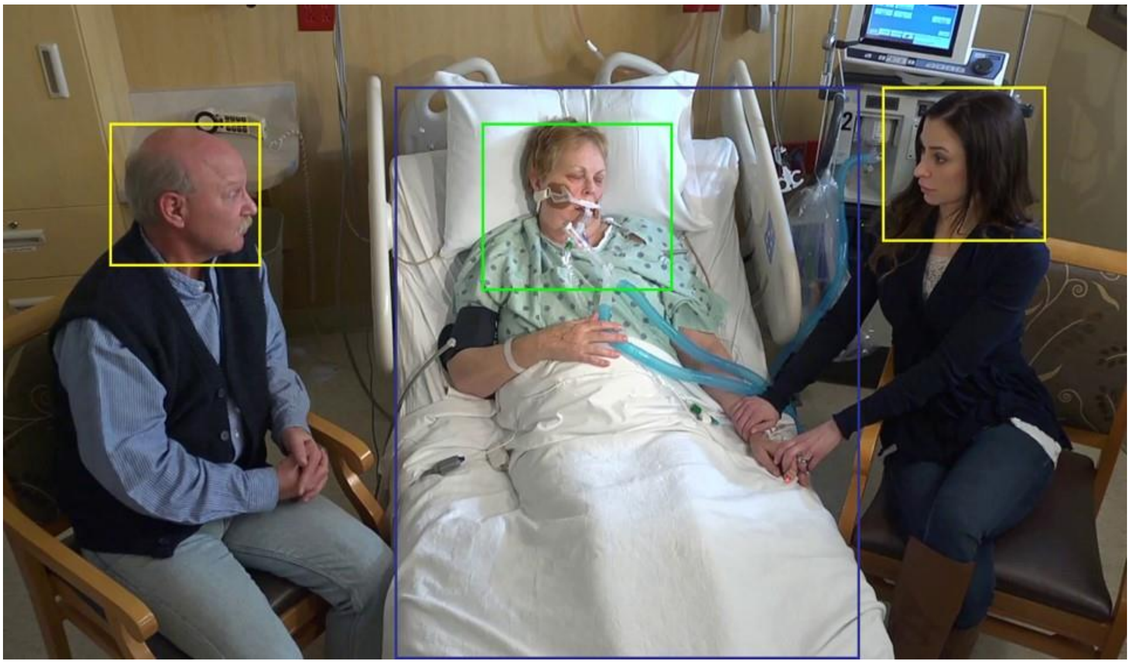
Key Steps of Designing a Novel Monitoring System for Tele-ICU Patients Based on Video Processing and Deep Learning It starts by setting out clear aims and requirements from the healthcare providers to understand their wants and limitations. The feed from Tele-ICU video is collected to work data which will then be subjected to intense pre-processing and cleaning for making the information standardised further. Feature extraction algorithms are then used to extract pertinent visual and physiological cues from the video streams, employing computer vision techniques like motion recognition and heart rate estimation.

Finally, using the extracted features different deep learning models like convolutional neural networks (CNN) or recurrent neural networks (RNN) are selected to develop. Trained on annotated data sets these models are trained to identify patterns in patient activity and any deviation. The system relies on a configuration of real-time processing pipelines to enable continuous analysis of incoming video data, integrated within existing Tele-ICU systems and electronic health records (EHR).

Its basic feature is to provide alerting capability (alerts and symptoms) to healthcare providers on noticing critical events or abnormal patient states. Measurement of performance with controlled environment and pilot studies in clinical settings; assessment metrics like accuracy, sensitivity. Ethical issues like protecting patient privacy, data security and respecting the right of patients to control their own information are closely attended from beginning all through its development till deployment.

A process-oriented deployment approach, meanwhile, guarantees roll-out in steps and scale that is followed-up by sustainable maintenance and system updates to upkeep the performance of your application. Their end-to-end methodology applies recent advances in video processing and deep learning technology to boost Tele-ICU patient-watching, register an early intervention that is central; the success of healthcare care on time which calls for timely clinical judgement.

RESULTS AND DISCUSSION



CONCLUSION

Overall, the creation of an advanced monitoring system for Tele-ICU Patients was a giant step in transforming critical care medicine via video processing and deep learning algorithms. The powerful combination of advanced technologies applied to health has given healthcare providers the ability to remotely monitor patients, in real-time and without geographic or resource limitations.

The system leverages video processing to extract and analyse subtle visual signals from live patient feeds, such as motions of the body, facial expressions of emotion or pain and cardio-respiratory vital signs. These deep learning models are then trained to detect otherwise subtle patterns and anomalies that can signify changes in health state or risk. By integrating these technologies, we can monitor the patient continuously and intervene if necessary, so that complications are caught quickly, and medical interventions made promptly to dramatically improve outcomes for our patients.

Lastly, the timeliness with which alerts and notifications can be generated by the system ensures that healthcare teams are informed quickly so they can act in an efficient manner to utilize resourcesoptimaly for delivering care. The system drives efficiencies in the monitoring process and brings surrounding intelligent insights, contributing to both patient safety as well as helping to make health care more efficient and sustainable on a comprehensive level.

The moral implications of patient privacy, consent and data security were considered throughout the development process (and beyond deployment). Strong laws help establish adherence to best practices and building trust in how personal patient data is used responsibly within healthcare.

Future developments in artificial intelligence and medical technology will pave the way for more sophisticated Tele-ICU monitoring systems going forward. To continue to build on these methods, with a goal of improving implementation and applicability in other clinical contexts so that ultimately the advancement spread standard for critical care worldwide.

Simply put, the new monitoring system using video processing and deep learning is one of the key technologies that convey tele-icu care into 21st century healthcare allowing unprecedented abilities by clinicians to offer rapid personalized critical time sensitive individualized treatment for Tele ICU patient setting fully realizing a future direction towards modern era digital or eICU-intensive medicine.

CODE

```
import cv2
import os
import numpy as np
from tensorflow. keras.models import load model

# Function to extract frames from a video
def extract_frames(video_path, output_folder):
    if not os.path.exists(output_folder):
        os.makedirs(output_folder)

    cap = cv2.VideoCapture(video_path)
    count = 0
    success = True

    while success:
        success, frame = cap.read()
        if success:
            frame_filename = os.path.join(output_folder, f"frame_{count}.jpg")
            cv2.imwrite(frame_filename, frame)
            count += 1

    cap.release()
    cv2.destroyAllWindows()

# Function to load and preprocess the extracted frames
def load_data(image_folder):
    images = []
    frame_paths = []

    for img_name in os.listdir(image_folder):
        img_path = os.path.join(image_folder, img_name)
        img = cv2.imread(img_path)
        img = cv2.resize(img, (128, 128))  # Resize to the input shape of the model
        images.append(img)
        frame_paths.append(img_path)
```

```
    images = np.array(images) / 255.0  # Normalize the images
    return images, frame_paths

# Function to classify frames and give response
def classify_frames(model, images, frame_paths):
    predictions = model.predict(images)
    for i, prediction in enumerate(predictions):
        condition = "Alert" if prediction > 0.5 else "Normal"
        print(f"Frame: {frame_paths[i]} - Condition: {condition}")

# Example usage
video_path = r'C:\Users\Pavan P Kulkarni\Desktop\New folder\video.mp4'
output_folder = r'C:\Users\Pavan P Kulkarni\Desktop\New folder\output'
model_path = r'C:\Users\Pavan P Kulkarni\Desktop\New folder\output.h5'

# Step 1: Extract frames from video
extract_frames(video_path, output_folder)

# Step 2: Load and preprocess the extracted frames
images, frame_paths = load_data(output_folder)

# Step 3: Load a pre-trained model
model = load_model(model_path)

# Step 4: Classify frames and give response
classify_frames(model, images, frame_paths)
```

SOLUTION FEATURES

1. Real-Time Patient Monitoring
2. Health Monitoring and Vital Signs Analysis
3. Patient Safety and Risk Assessment
4. Behavioral Analysis and Patient Care Optimization

TEAM CONTRIBUTION

The contribution of INNOVATIVE MONITORING SYSTEM FOR TELE ICU PATIENTS USING VIDEO PROCESSING AND DEEP LEARNING. This project is obediently completed with the contribution of team members, but also there are few cases where individuals worked hard for a particular work such as

1. Shashank Marbe and Shrinath Baradar worked on the resource related to coding and algorithms and to presentthe working of project.

2. Siddaram and Vinay Kumar worked on the report and studied the whole project given a bestpresentation in report.