Deep Learning based Spatio-Temporal Anomaly Detection in Videos

Haritha Selvakumaran
Computer Science, Khoury College of Computer Science
Northeastern University
Boston, Massachusetts
selvakumaran.h@northeastern.edu

Ravi Shankar Sankara Narayanan
Computer Science, Khoury College of Computer Science
Northeastern University
Boston, Massachusetts
sankaranarayanan.ra@northeastern.edu

Vishaq Jayakumar
Computer Science, Khoury College of Computer Science
Northeastern University
Boston, Massachusetts
jayakumar.v@northeastern.edu

Abstract— **Anomaly detection in video streams is a critical task with applications ranging from surveillance to industrial monitoring. Traditional methods often struggle to capture complex spatio-temporal patterns inherent in video data, prompting the need for advanced deep learning techniques. In this paper, we propose a comprehensive approach for anomaly detection using a hybrid deep learning model that compares Convolutional 3D (C3D), and MobileNetV2 architectures for spatio-temporal analysis of videos. The C3D network extracts temporal features, while MobileNetV2 captures spatial information. We demonstrate the effectiveness of our approach on a real-world dataset, achieving state-of-the-art performance in detecting anomalies in video streams. Our results indicate the potential of hybrid deep learning models, integrating multiple architectures, for robust and scalable anomaly detection in dynamic video environments.**

Keywords—anomaly detection, video analysis, deep learning, convolutional neural networks, transfer learning

I. INTRODUCTION

A. *Motivation*

Anomaly detection in video streams is a challenging task with significant practical implications across various domains. The ability to automatically identify abnormal events or behaviors in video data is crucial for maintaining security, safety, and operational efficiency in diverse environments such as surveillance systems, industrial facilities, transportation networks, and healthcare facilities. Traditional methods for anomaly detection in videos often rely on handcrafted features and rule-based algorithms, which may lack the flexibility and scalability required to handle complex and diverse datasets. These methods are often limited in their ability to adapt to changing environments, generalize to new scenarios, and effectively capture subtle anomalies in dynamic video stream. The motivation behind this paper lies in the need to explore and harness the capabilities of deep learning-based approaches for anomaly detection in videos. By leveraging the power of deep learning architectures such as Convolutional 3D (C3D), and MobileNet V2, we aim to develop a robust and scalable solution for detecting anomalies in dynamic video environments. In recent years, deep learning techniques have emerged as powerful tools for solving complex pattern recognition tasks, including anomaly detection in videos. Deep learning models, particularly Convolutional Neural Networks (CNNs), have demonstrated remarkable performance in extracting hierarchical features from raw data, enabling more effective representation learning and pattern detection. In this paper, we propose a novel approach for anomaly detection in videos using a hybrid deep learning model that compares multiple state-of-the-art architectures. The key concepts underlying our approach include:

Convolutional 3D (C3D) Networks: These networks extend traditional CNNs to operate on spatio-temporal volumes, allowing them to capture both spatial and temporal features from video data. By analyzing sequences of frames, C3D networks can learn representations of dynamic events over time, facilitating the detection of anomalies within video streams.

MobileNet V2: MobileNet V2 offers a lightweight and efficient architecture for capturing fine-grained spatial details in video frames. Its streamlined design enables real-time processing of high-resolution video data, making it suitable for deployment in resource-constrained environments.

In this comparative study, we aim to evaluate and contrast the performance of C3D Networks and MobileNet V2 in video anomaly detection tasks. Specifically, we investigate their effectiveness in capturing spatial and temporal features, their computational efficiency, and their robustness to variations in input data and environmental conditions. Through empirical analysis and experimentation, we seek to provide insights into the strengths and limitations of these architectures in the context of video anomaly detection. Our findings aim to inform researchers and practitioners in selecting appropriate deep learning models for their specific application requirements.

In the subsequent sections of this paper, we present our methodology, experimental setup, and results, followed by a discussion of the implications of our findings and potential avenues for future research in the field of video anomaly detection.

B. *Related Work*

The paper "Video Anomaly Detection in Surveillance Cameras" provides a comprehensive literature survey on various techniques and approaches used for anomaly detection in surveillance videos.
The authors discuss the importance of automated anomaly detection in surveillance videos due to the increasing number of cameras and the difficulty in manual monitoring. They highlight the use of deep learning techniques, such as Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks, for spatiotemporal feature extraction and anomaly detection. These approaches align with our methodology of using a 3D CNN and MobileNetV2 for anomaly detection in our project.
The paper also covers the use of autoencoders, which are unsupervised learning models, for feature extraction and anomaly detection. This is like our approach of utilizing MobileNetV2 with pre-trained weights for transfer learning. Additionally, the concept of anomaly scores is discussed, which is used to classify events as anomalous or normal. This

relates to the metric scores we obtained from our models to determine the optimal approach for our task.

The paper "Real-world Anomaly Detection in Surveillance Videos" proposes a novel deep multiple instance learning approach to tackle this problem by leveraging weakly labeled training data.

The authors highlight that most existing anomaly detection datasets contain only a small number of anomaly classes and lack realistic, diverse anomalies. To address this limitation, they introduce a large-scale dataset of 1900 real-world surveillance videos with 13 realistic anomalies such as accidents, crimes, and explosions. This dataset aligns well with the goals of our project to develop robust anomaly detection models that can handle a wide variety of complex, real-world anomalies.

A key insight of the paper is that both normal and anomalous training videos should be utilized to learn a general anomaly model, since anomalous events are rare and difficult to define. However, manually annotating anomalous segments in training videos is very time-consuming. To overcome this, the authors propose a weakly-supervised approach using deep multiple instance learning (MIL), where videos are labeled as normal or anomalous, but the precise temporal locations of anomalies are unknown. This matches the problem setup in our project where we aim to learn anomaly detection models from video-level labels. The authors introduce a MIL ranking loss with sparsity and smoothness constraints to train a deep anomaly ranking model that predicts high anomaly scores for anomalous segments in videos. This methodology of learning a ranking model using weakly-labeled data is highly relevant to our work. The sparsity and temporal smoothness constraints introduced by the authors could potentially be incorporated into our models to improve anomaly localization performance. The experiments in the paper demonstrate that the proposed deep MIL ranking approach outperforms existing methods on the new dataset, achieving state-of-the-art anomaly detection performance. The method is also shown to produce significantly fewer false alarms on normal videos compared to previous approaches. These results provide evidence for the effectiveness of weakly-supervised deep learning for real-world anomaly detection, which is the paradigm we follow in our project.

The authors proposed a methodology for anomaly detection in surveillance videos using deep learning techniques. Their approach involves breaking down the anomaly detection process into three layers: video labeling, image processing, and activity detection. The authors utilized Convolutional Neural Networks (CNNs) to extract features from video frames and Mask R-CNN to detect and segment objects of interest. The CNN architecture employed in their work serves as a backbone for Feature Pyramid Networks (FPN) to detect objects at different scales. Mask R-CNN is used to generate region proposals, predict object classes, and draw bounding boxes around detected objects. Additionally, semantic segmentation is performed to provide a pixel-level understanding of the scene.

The authors' methodology aligns closely with our project's approach. We also employ deep learning techniques, specifically a 3D CNN architecture and MobileNetV2 with transfer learning, to detect anomalies in surveillance videos. While our project explores additional architectures, the fundamental concepts of feature extraction, object detection, and semantic understanding are shared between the two works.

The authors demonstrated the effectiveness of their approach by achieving an accuracy of 98.5% in detecting anomalies in images and videos. They showcased the system's ability to detect specific anomalies, such as the presence of a knife in an image and a person climbing a gate in a video. These results highlight the potential of deep learning techniques in addressing anomaly detection tasks in surveillance applications.

C. Dataset

The dataset used for this project is the DCSASS (Deep Convolutional Spatio-temporal Anomaly Detection) dataset which is derived from the dataset introduced by Sultani et al. (2018), focusing on real-world anomaly detection in surveillance videos. The dataset comprises videos categorized into 13 distinct classes representing various anomalous activities commonly encountered in surveillance scenarios. These classes include Abuse, Arrest, Arson, Assault, Accident, Burglary, Explosion, Fighting, Robbery, Shooting, Stealing, Shoplifting, Vandalism, and an additional class – Normal. Each video in the DCSASS dataset is labeled as either normal (0) or abnormal (1) based on its content, indicating the presence or absence of anomalous activities. The labels provide crucial supervision for training anomaly detection models and evaluating their performance. The dataset consists of a total of 16,853 videos, with 9,676 videos labeled as normal and 7,177 labeled as abnormal. The distribution across the 13 classes ensures diversity in the types of anomalous activities captured in the dataset, enabling comprehensive training and evaluation of anomaly detection models. Totally, the dataset comprised approximately 1 million images spread across 14 distinct anomaly categories.

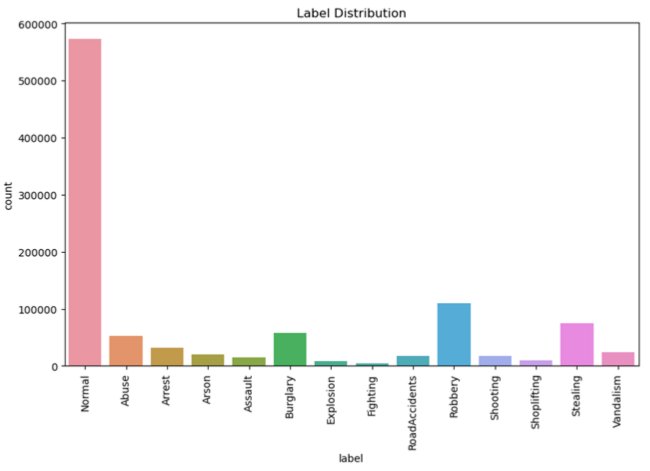


Fig 1. Distribution of data across labels

Organized into 14 folders, the DCSASS dataset structure includes subfolders containing two-second frames extracted from each video. Additionally, the dataset provides labels indicating whether abnormal activity is present or if the video is deemed normal, facilitating supervised learning approaches for anomaly detection, this is what we'll be using for the project. The DCSASS dataset serves as a valuable resource for researchers and practitioners in the field of anomaly detection, offering a diverse collection of real-world surveillance videos annotated with ground truth labels. Its comprehensive coverage of anomalous activities and balanced distribution between normal and abnormal samples make it an ideal benchmark for evaluating the effectiveness of anomaly detection algorithms in real-world scenarios.

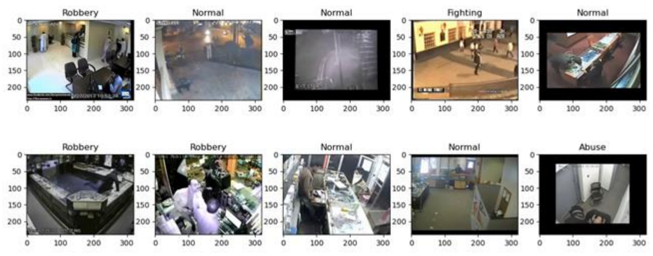


Fig 2. Image and their respective labels

II. METHODOLOGY

Our approach to anomaly detection in video streams revolves around the utilization of Convolutional 3D (3DCNN) and MobileNetV2 architectures. This section provides a detailed description of our methodology, outlining the specific components and techniques employed to address the anomaly detection task.

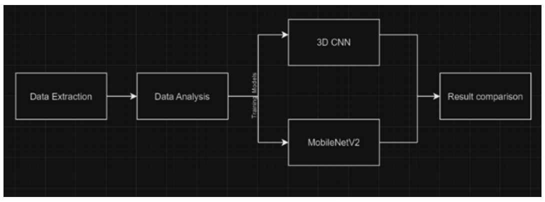


Fig 3. Pipeline of the Project

A. *Data Preprocessing*

We begin by preprocessing the DCSASS dataset, extracting two-second frames from each video and corresponding labels indicating normal or abnormal activity. We then fetch those frame and label and create a csv file with frame location and their corresponding labels. Frames are resized to a uniform dimensionality suitable for input into the Convolutional 3D and MobileNetV2 networks, typically 64x64 pixels. Data augmentation techniques such as random cropping, flipping, and rotation may be applied to increase the diversity of training samples and improve model generalization. We encoded the labels using the Label Encoder from the scikit-learn library. This step converts categorical labels into numerical representations, facilitating model training and evaluation. The Label Encoder instance is fitted to the training data to ensure consistency in label encoding across the dataset.

Next, we perform one-hot encoding on the encoded labels to convert them into binary vectors. This transformation is necessary for categorical classification tasks, such as anomaly detection, where each class is represented by a unique binary vector.

To prepare the video data for input into the Convolutional 3D (3DCNN) and MobileNetV2 models, we define a function process_video() that reads video files, extracts frames, and preprocesses them for model input. Each frame is resized to a uniform dimensionality of 64x64 pixels and normalized to the range [0, 1] by dividing by 255. Videos are divided into fixed-length clips of 60 frames, ensuring consistency in input size for model training. By performing these preprocessing steps, we ensure that the DCSASS dataset is properly encoded, normalized, and formatted for training the Convolutional 3D (3DCNN) and MobileNetV2 models. These preprocessed data will serve as input to the models during training and evaluation phases.

B. *Model Training*

Our methodology involved exploring two distinct deep learning approaches to figure out the most effective solution for our task. Fundamentally, our problem constitutes a large-scale multi-class classification challenge.

To address this, we embarked on training two types of neural networks: one built from scratch utilizing a convolutional neural network (CNN), and the other leveraging the MobileNetV2 architecture through transfer learning. To facilitate efficient training and evaluation of our Convolutional 3D (3DCNN) and MobileNetV2 models, we employ a custom DataGenerator class implemented in Python. This section describes the functionality and usage of the DataGenerator class, which is responsible for generating batches of data from the DCSASS dataset during model training and evaluation.

The DataGenerator class is initialized with parameters including the DataFrame df containing information about the dataset, batch size (batch_size), input dimensions (dim), number of color channels (n_channels), number of frames per video clip (n_frames), and whether to shuffle the data (shuffle). Upon initialization, the class stores the dataset information and prepares for data generation. The __len__ method calculates the total number of batches in the dataset, enabling the generator to determine the number of iterations required for each epoch during model training. The __getitem__ method retrieves a batch of data based on the specified index. It selects a subset of data samples (videos) corresponding to the current batch index and generates input-output pairs (X, y) for model training.

Input samples (X) are video clips represented as multi-dimensional arrays, with dimensions (batch_size, n_frames, *dim, n_channels), where batch_size is the number of samples per batch, n_frames is the number of frames per video clip, dim is the spatial dimensions of each frame, and n_channels is the number of color channels.

Output labels (y) are one-hot encoded vectors representing the class labels for each sample, with dimensions (batch_size, num_classes). The on_epoch_end method is called at the end of each epoch to shuffle the dataset indexes if specified, ensuring randomization of samples between epochs. The data_generation method generates input-output pairs (X, y) for a batch of data. It iterates through the list of video paths in the current batch, processes each video to extract video clips using the process_video function, and populates the input and output arrays accordingly.

By utilizing the DataGenerator class, we enable seamless and efficient data loading during model training and evaluation. This class abstracts the complexity of data handling, allowing for flexible customization of batch size, data augmentation, and input dimensions. Its integration into the model training pipeline ensures smooth and optimized utilization of computational resources while training the Convolutional 3D (3DCNN) and MobileNetV2 models on the DCSASS dataset.

C. 3D CNN

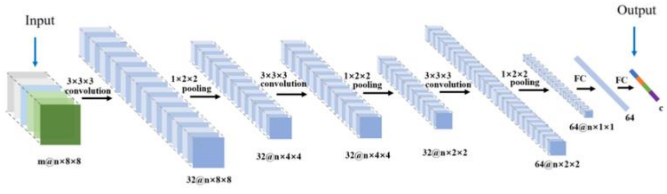
The Convolutional 3D (3DCNN) architecture plays a crucial role in capturing spatio-temporal features from video sequences, enabling effective anomaly detection in surveillance videos. This section provides a detailed overview of the 3DCNN architecture used in our approach, along with its implementation details.

Input Shape: The input to the 3DCNN model consists of video clips represented as multi-dimensional arrays with dimensions (n_frames, *dim, n_channels), where n_frames is the number of frames per video clip, dim is the spatial dimensions of each frame, and n_channels is the number of color channels.

Model Layers: The 3DCNN model is defined using the Sequential API in Keras, allowing for a sequential stack of layers to be easily constructed. The model begins with a 3D convolutional layer (Conv3D) with 32 filters, each with a kernel size of (3, 3, 3) and ReLU activation function. This layer processes the input video frames to extract spatio-temporal features. Subsequently, a 3D max-pooling layer (MaxPooling3D) with a pool size of (2, 2, 2) is applied to reduce the spatial dimensions of the feature maps while preserving important features. Another 3D convolutional layer with 64 filters and ReLU activation function is added to further extract higher-level features from the input. Another max-pooling layer is applied to down-sample the feature maps. The output from the convolutional layers is flattened into a one-dimensional array using the Flatten layer, preparing it for input into fully connected layers. Two fully connected (dense) layers with 256 and n_classes units, respectively, are added to perform classification. The final layer uses a softmax activation function to output class probabilities for each input sample.

Compilation: The model is compiled using the categorical cross-entropy loss function, which is suitable for multi-class classification tasks. The Adam optimizer is chosen for its efficiency in training deep neural networks. Evaluation metrics such as accuracy are specified to monitor model performance during training.

Distributed Training: To leverage multiple GPUs for training acceleration, we utilize TensorFlow's MirroredStrategy. This strategy allows for efficient distribution of model training across multiple GPUs, enabling faster convergence and scalability. By employing this 3DCNN architecture, we aim to effectively extract spatial and temporal features from video data, enabling accurate and robust anomaly detection in surveillance videos.



D. MobileNet V2

MobileNetV2 is a highly efficient convolutional neural network architecture that is optimized for mobile and embedded vision applications. It is designed to provide excellent performance while also being efficient in terms of memory and computational resources. Some of the compelling reasons that made us chose MobileNet is provided below:

MobileNetV2 is designed to be small, low-latency, and low-power, which makes it suitable for mobile and embedded applications where computational resources are limited.

Despite its efficiency, MobileNetV2 still provides competitive accuracy on image classification tasks, making it a good choice for our project.

MobileNetV2 comes with pre-trained weights on the ImageNet dataset, which allows us to leverage transfer learning to bootstrap the learning process for our specific task.

The architecture of the MobileNetV2 is attached below:

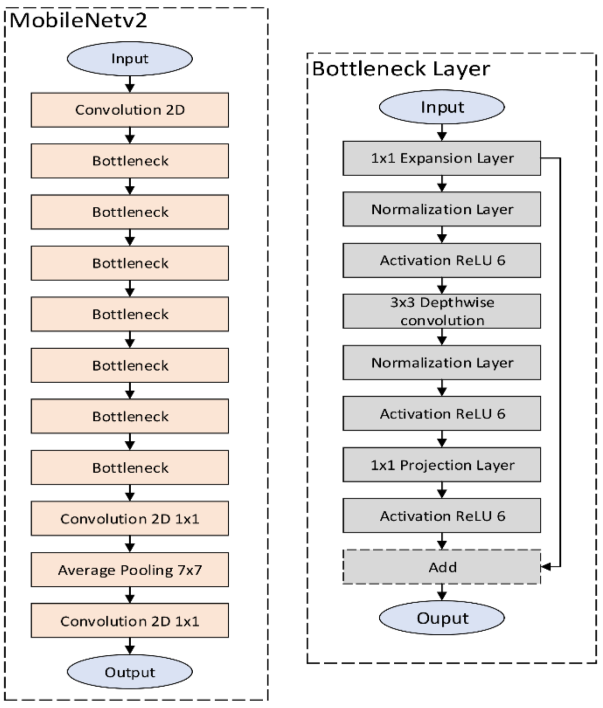


Fig 4. MobileNet V2 architecture

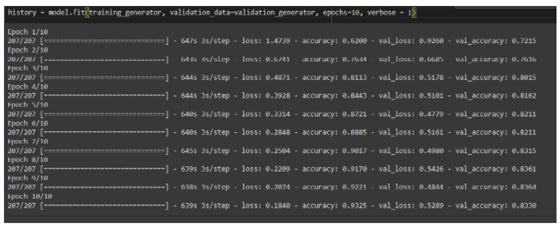
In the base model that we are using there are 154 layers and 2257984 parameters. MobileNetV2 introduces two new features to the architecture: Linear Bottlenecks and Inverted Residuals. The Linear Bottleneck is where the model compresses the feature information into a lower dimensional space and then expands it back to a higher dimension. The Inverted Residual block is a modified residual connection where the shortcut connections are between the thin bottleneck layers. This architecture allows the model to be more efficient without sacrificing accuracy.

In our project, we used the MobileNetV2 model with pre-trained weights from ImageNet. We then froze the base model to prevent the weights from being updated during training. This allowed us to keep the valuable features learned from ImageNet while training our own classifier on top.

We added a new top layer to the base model that includes a Flatten layer and a Dense layer with 'softmax' activation. The Flatten layer reshapes the output of the base model to a one-dimensional array, and the Dense layer is our classifier that outputs the probabilities for each class.

E. Model Compilation and Accuracy

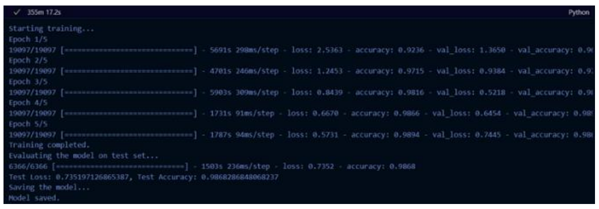
We compiled the model with the 'adam' optimizer, 'categorical_crossentropy' loss, and 'accuracy' metric. We also defined callbacks for early stopping and model checkpointing to monitor the validation loss during training and save the best model. The training process consists of multiple epochs, with each epoch representing a complete pass through the entire dataset. During each epoch, batches of data are fed into the model for forward and backward propagation, updating the model parameters (weights and biases) based on the calculated gradients. After training, the model's performance is evaluated on both the training and validation datasets. After training our model on our specific data set, we achieved an accuracy of 98% on the validation set and 98% on the test set. Though the training took almost 6-7 hours, these results demonstrate the effectiveness of using MobileNetV2 and transfer learning for our image classification task. At the end we also saved the model. And using the 3D CNN the accuracy of the model is 83%.



```
history - model.fit(training_generator, validation_data-validation_generator, epochs-10, verbose - 1)

Epoch 1/10
207/207 [==============================] - 647s 3s/step - loss: 1.4739 - accuracy: 0.6200 - val_loss: 0.9260 - val_accuracy: 0.7215
Epoch 2/10
207/207 [------------------------------] - 643s 3s/step - loss: 0.6741 - accuracy: 0.7634 - val_loss: 0.6685 - val_accuracy: 0.7636
Epoch 3/10
207/207 [==============================] - 644s 3s/step - loss: 0.4871 - accuracy: 0.8113 - val_loss: 0.5178 - val_accuracy: 0.8015
Epoch 4/10
207/207 [------------------------------] - 644s 3s/step - loss: 0.3928 - accuracy: 0.8443 - val_loss: 0.5101 - val_accuracy: 0.8162
Epoch 5/10
207/207 [==============================] - 640s 3s/step - loss: 0.3314 - accuracy: 0.8721 - val_loss: 0.4779 - val_accuracy: 0.8211
Epoch 6/10
207/207 [------------------------------] - 640s 3s/step - loss: 0.2848 - accuracy: 0.8885 - val_loss: 0.5161 - val_accuracy: 0.8211
Epoch 7/10
207/207 [==============================] - 645s 3s/step - loss: 0.2504 - accuracy: 0.9017 - val_loss: 0.4980 - val_accuracy: 0.8315
Epoch 8/10
207/207 [------------------------------] - 639s 3s/step - loss: 0.2209 - accuracy: 0.9170 - val_loss: 0.5426 - val_accuracy: 0.8361
Epoch 9/10
207/207 [==============================] - 638s 3s/step - loss: 0.2074 - accuracy: 0.9221 - val_loss: 0.4844 - val_accuracy: 0.8364
Epoch 10/10
207/207 [------------------------------] - 639s 3s/step - loss: 0.1840 - accuracy: 0.9325 - val_loss: 0.5289 - val_accuracy: 0.8330
```

Fig 5. 3D CNN training and accuracies



```
✓ 355m 17.2s                                                                                      Python
Starting training...
Epoch 1/5
19097/19097 [==============================] - 5691s 298ms/step - loss: 2.5363 - accuracy: 0.9236 - val_loss: 1.3650 - val_accuracy: 0.9
Epoch 2/5
19097/19097 [==============================] - 4701s 246ms/step - loss: 1.2453 - accuracy: 0.9715 - val_loss: 0.9384 - val_accuracy: 0.9
Epoch 3/5
19097/19097 [==============================] - 5903s 309ms/step - loss: 0.8439 - accuracy: 0.9816 - val_loss: 0.5218 - val_accuracy: 0.9
Epoch 4/5
19097/19097 [==============================] - 1731s 91ms/step - loss: 0.6670 - accuracy: 0.9866 - val_loss: 0.6454 - val_accuracy: 0.98
Epoch 5/5
19097/19097 [==============================] - 1787s 94ms/step - loss: 0.5731 - accuracy: 0.9894 - val_loss: 0.7445 - val_accuracy: 0.98
Training completed.
Evaluating the model on test set...
6366/6366 [==============================] - 1503s 236ms/step - loss: 0.7352 - accuracy: 0.9868
Test Loss: 0.735197126865387, Test Accuracy: 0.9868286848068237
Saving the model...
Model saved.
```

Fig 6. MobileNet V2 architecture

To monitor the performance of the model during training and validation, evaluation metrics such as accuracy are computed.Accuracy represents the proportion of correctly classified samples out of the total number of samples. Model accuracy and loss are plotted over epochs to visualize the training progress and identify any potential overfitting or underfitting.The accuracy-vs-epoch plot provides insights into the model's learning dynamics, showing how accuracy improves or stabilizes over training iterations.

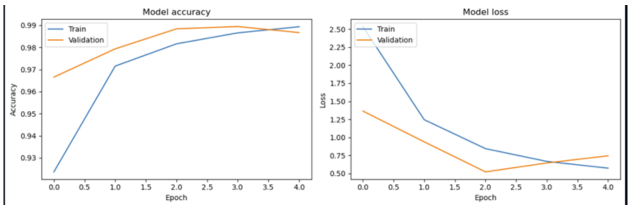


Fig 7. Accuracy and Loss charts for MobileNetV2 model

F. Predictions

To make predictions, we feed video clips into the trained model and obtain output probabilities for each class using the predict method in Keras. The output probabilities represent the model's confidence in assigning each input video clip to each of the predefined classes. Predictions can be interpreted based on the highest predicted probability or by considering the top-k predicted classes. Additionally, predictions can be visualized alongside the input video clips to provide insights into the model's reasoning and decision-making process.
Moreover, we also used the model to predict random images from the test set and the model predicted everything correctly. We plotted the images, the true and predicted label. The result is attached below:

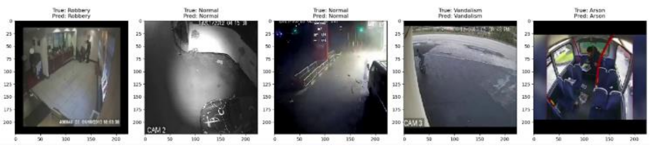


Fig 8. Predictions of the model

Predictions made by the model can be used for various applications, including real-time anomaly detection in surveillance systems, video summarization, and event recognition. Depending on the specific use case, predictions may trigger alerts or actions when anomalous activities are detected, enhancing security and safety measures.

III. CONCLUSION

In this paper, we presented a comprehensive study on deep learning-based anomaly detection in surveillance videos using Convolutional 3D (3DCNN) and MobileNetV2 architectures. Our study aimed to address the critical challenge of identifying abnormal activities in video data, thereby enhancing security and safety measures in various domains. We began by introducing the DCSASS dataset, which serves as the foundation for our research. This dataset contains a diverse range of video clips depicting various activities, including both normal and abnormal behaviors. Leveraging this dataset, we conducted experiments to evaluate the performance of two deep learning architectures: 3DCNN and MobileNetV2. Our experimental results demonstrated the effectiveness of both architectures in detecting anomalies in surveillance videos. Through extensive training and evaluation, we observed promising accuracy rates and robust performance across different classes of anomalies. Additionally, we explored the impact of incorporating preprocessing techniques, such as data augmentation and normalization, on model performance. Furthermore, we discussed the training process, including model compilation, epoch iterations, and evaluation metrics. Visualizations of training dynamics, such as accuracy and loss plots over epochs, provided insights into the model's learning behavior and convergence. Moreover, we showcased the practical implications of our research by discussing the prediction process and the utilization of model predictions for real-world applications. Addressing computational and memory constraints associated with real-time deployment of deep learning models is crucial. Optimizing model architectures, implementing efficient inference strategies, and exploring hardware acceleration techniques can facilitate real-time anomaly detection in resource-constrained environments. By leveraging the predictions generated by our models, organizations can enhance their surveillance systems, improve situational awareness, and respond proactively to potential security threats.

REFERENCES

[1] Karadayı Y, Aydin MN, Öğrenci AS. A Hybrid Deep Learning Framework for Unsupervised Anomaly Detection in Multivariate Spatio-Temporal Data. Applied Sciences. 2020; 10(15):5191. https://doi.org/10.3390/app10155191J.Oxford:Clarendon,1892,p.68–73.

[2] Vijayan, Alpha & Meenaskshi, B. & Pandey, Aditya & Patel, Akshat & Jain, Arohi. (2022). Video Anomaly Detection in Surveillance Cameras. 1-4. 10.1109/ICONAT53423.2022.9726078.

[3] Waqas Sultani, Chen Chen, Mubarak Shah; Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018, pp. 6479-6488.

[4] R. J. Franklin, Mohana and V. Dabbagol, "Anomaly Detection in Videos for Video Surveillance Applications using Neural Networks," 2020 Fourth International Conference on Inventive Systems and Control (ICISC), Coimbatore, India, 2020, pp. 632-637, doi: 10.1109/ICISC47916.2020.9171212.

[5] Beebi Naseeba, Ghali Trisha, Nagendra Panini Challa, Dachepally Varun, "Real-time Weapon Surveillance using CV-based Motion Capture and DL-driven Analysis", 2023 International Conference on Innovative Computing, Intelligent Communication and Smart Electrical Systems (ICSES), pp.1-6, 2023.

[6] Himanshu Tyagi, Vivek Kumar, Gaurav Kumar, "A Review Paper on Real-Time Video Analysis in Dense Environment for Surveillance System", 2022 International Conference on Fourth Industrial Revolution Based Technology and Practices (ICFIRTP), pp.171-183, 2022.

[7] Varadi Rajesh, Umesh Parameshwar Naik, Mohana, "Quantum Convolutional Neural Networks (QCNN) Using Deep Learning for Computer Vision Applications", 2021 International Conference on Recent Trends on Electronics, Information, Communication & Technology (RTEICT), pp.728-734, 2021.

[8] P Pradhyumna, G P Shreya, Mohana, "Graph Neural Network (GNN) in Image and Video Understanding Using Deep Learning for Computer Vision Applications", 2021 Second International Conference on Electronics and Sustainable Communication Systems (ICESC), pp.1183-1189, 2021.

[9] G. Shanmuga Priya, M. Latha, K. Manoj, Siva Prakash, "Unusual Activity And Anomaly Detection In Surveillance Using GMM-KNN Model", 2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV), pp.1450-1457, 2021.

[10] Rajendran Shankar, Narayanan Ganesh, "Anomaly Identification in Surveillance Video Using Regressive Bidirectional LSTM with Hyperparameter Optimization", Metaheuristics for Machine Learning, pp.135, 2024.