ISSN No:-2456-2165

Detecting the Unseen: A Study of CNNs for Real Time Abnormal Event Detection

Zainab Khan[1]
AMC Engineering College Bengaluru, India

Varsha K[2]
AMC Engineering College Bengaluru, India

Uzaira Sahar[3]
AMC Engineering College Bengaluru, India

Vaibhavi M.G[4]
AMC Engineering College Bengaluru, India

Velvizhi Ramya R[5] Guide (Professor)
Department of CSE
AMC Engineering College Bengaluru, India

Abstract:- Abnormal event detection is a critical task in various domains, including surveillance, healthcare, and industrial monitoring. This paper explores the application of Convolutional Neural Network (CNNs) for detecting abnormal events in dynamic environments. By leveraging CNNs' ability to extract spatial and temporal features, we primarily aim to enhance the accuracy and efficiency of anomaly detection. To validate our approach, we employed a comparative analysis of supervised and unsupervised learning techniques. Extensive experimentation on our datasets demonstrated that CNNs consistently outperformed traditional methods in identifying anomalies with higher precision and recall rates. The results highlight the potential of CNNs as a robust solution for abnormal event detection, effectively balancing computational efficiency and detection accuracy. Our findings highlight the importance of integrating domain-specific insights and using advanced architectures to address all the challenges in anomaly detection.

Keywords:- *Event Detection, CNN, K-means, Logistic Regression, Supervised, Unsupervised, Model, NLP, Image Recognition.*

I. INTRODUCTION

➢ *AI Integration for Enhanced Detection*

AI-driven event detection mainly relies on data collection and preprocessing. [1] Data is generally acquired from varied assets such as surveillance systems, IoT devices, social media feeds, sensors and satellite imagery. This untreated data is cleaned and standardized to ensure it is fit for analysis. High-Quality and authenticated data is essential for accurate detection. Once prepared, this data is input to machine learning (ML) algorithms that are built to identify patterns indicative of important events. Real-time analytics is a hallmark of AI-powered event detection. In contrast to conventional systems that examine data after an event has occurred, AI allows for immediate detection of events as they unfold. This feature is crucial in time-critical situations, such as emergency responses, where delays can lead to severe outcomes. Technologies like edge computing improve real-time processing by carrying out computations near the data origin, minimizing latency and enabling rapid responses. The rapid advancements in Artificial Intelligence (AI) have revolutionized anomaly detection, a critical task in applications such as surveillance, healthcare, industrial monitoring, and autonomous systems. Among AI techniques, Convolutional Neural Networks (CNNs) have emerged as a powerful tool due to their ability to automatically learn hierarchical features from complex datasets. Abnormal event detection aims to identify rare, unexpected, or suspicious patterns in data, which often lack explicit labels and are difficult to capture using Traditional methods.

II. TRANSFORMATION OF REAL TIME INSIGHTS IN THE REAL WORLD

AI-driven event detection is transforming industries by addressing critical challenges and unlocking new possibilities. In public safety, AI- powered surveillance systems analyze video feeds to detect suspicious activities or potential threats. By identifying unusual behavior in real-time, these systems enhance security and prevent incidents before they escalate.

In healthcare, wearable devices and medical sensors continuously monitor patient data. AI algorithms process this data to detect critical health events, such as irregular heart rhythms or seizures, enabling timely interventions. This not only saves lives but also improves the quality of care and patient outcomes.

Environmental monitoring is another area where AI has made a profound impact. By analyzing data from satellites and sensors, AI systems detect natural events like earthquakes, floods, and wildfires. Early detection facilitates swift responses, minimizing damage and aiding in disaster management. Similarly, in finance, AI models monitor transactions and market trends to detect fraud or anomalies, ensuring the security and stability of financial systems.

ISSN No:-2456-2165

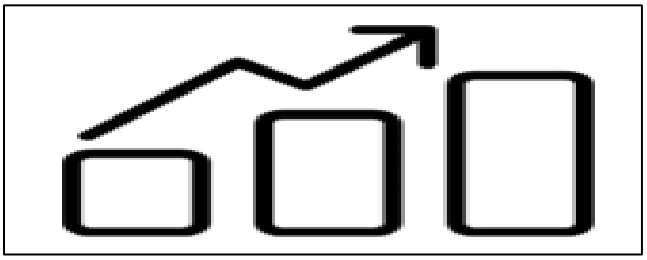

Fig 1 Insights Representation

III. DRIVING AI INTEGRATION – KEY TECHNOLOGIES USED

Event detection relies on a combination of advanced technologies to achieve its capabilities. Machine learning is central to recognizing patterns and anomalies within data. Supervised learning models excel in detecting predefined events, such as mechanical failures in industrial equipment, by learning from labelled datasets. On the other hand, unsupervised learning models identify outliers or unexpected occurrences, making them invaluable for uncovering rare or new events. In applications involving textual data, natural language processing (NLP) plays a crucial role.

NLP algorithms analyze language patterns, sentiment, and context to extract meaningful insights. For instance, during natural disasters, NLP can process social media posts to identify early warnings, assess damage, and coordinate response efforts. This ability to analyze unstructured text data significantly expands the scope of event detection. Predictive analytics is another significant advancement enabled by AI. By analyzing historical data, AI models can forecast future events, empowering industries to take proactive measures. Predicting disease outbreaks, financial market fluctuations, or equipment malfunctions are just a few examples of how predictive capabilities enhance decision- making and preparedness.

IV. HARNESSING SUPERVISED LEARNING MODELS FOR ABNORMAL EVENTS

A. KNN (K Nearest Neighbours)

KNN is a simple, non-parametric, supervised learning algorithm. It classifies data based on the majority class of the 'K' nearest points to a new data point in the feature space. The distance between data points is typically measured using Euclidean distance or other distance metrics like Manhattan distance.

➢ *Application in Abnormal Event Detection:*

[3] Detecting Abnormal Events: In anomaly detection, KNN works by comparing the distance between a new data point and its nearest neighbours. If the distance is significantly larger than that of most other data points, the new point can be classified as an anomaly. In other words, an anomaly is detected if it doesn't resemble the majority of data in the feature space.

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(algorithm = 'brute', n_jobs=-1)
knn.fit(x_train, y_train)
print("train shape: " + str(x_train.shape))
print("score on test: " + str(knn.score(x_test, y_test)))
print("score on train: "+ str(knn.score(x_train, y_train)))
```

```
train shape: (9137, 24)
score on test: 0.7422849638870 65
score on train: 0.8098938382401226
```

Fig 2- KNN train-test code

➢ *Advantages:*

- *Simple and Intuitive:* KNN is easy to understand and implement, and it doesn't require a complex training phase.

➢ *Challenges:*

- Computational Complexity: KNN requires storing the entire dataset and computing distances for every prediction, which makes it inefficient for large datasets.

B. Support Vector Machines (SVMs)

[4] SVM is a supervised learning algorithm used for classification. It finds a hyper plane that best separates different classes in the feature space, maximizing the margin between the classes. In the context of anomaly detection, the SVM algorithm is used to separate normal and abnormal data by constructing a hyper plane that best distinguishes normal data points from abnormal ones.

➢ *Application in Abnormal Event Detection:*

- One- Class SVM: A variation of SVM, known as one-class SVM, is specifically used for anomaly detection. It learns a boundary around the normal data points and considers any data point that lies outside this boundary as abnormal.

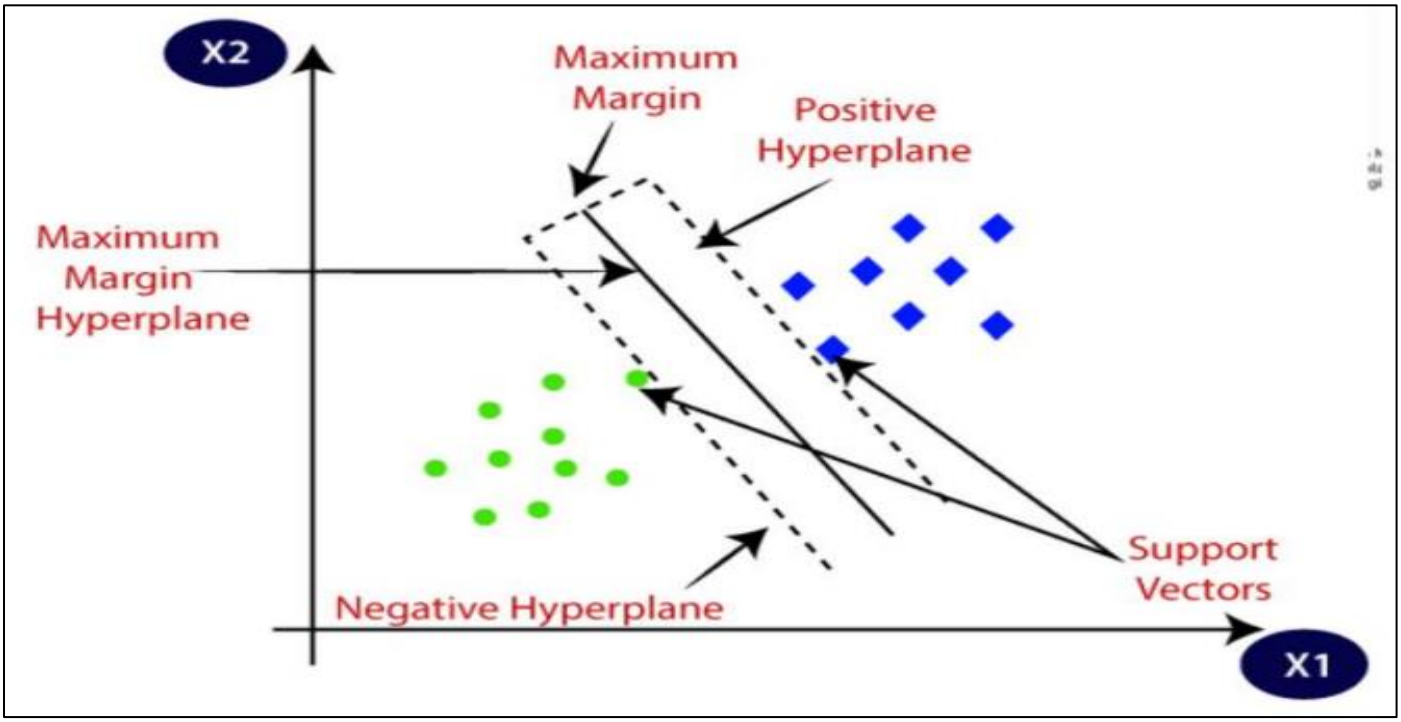

Fig 3 SVM Graph Representation

➢ *Advantages:*

- Effective in High-Dimensional Data: SVM works well in cases where the data is high- dimensional and non-linear. It uses kernel functions to transform data into higher dimensions for better separation.

➢ *Challenges:*

- Parameter Tuning: The performance of SVM is highly sensitive to the choice of kernel, regularization parameter, and other hyper parameters, requiring careful tuning.

```
from sklearn.svm import LinearSVC
svm=LinearSVC(C=0.0001)
svm.fit(x_train, y_train)
print("score on test: " + str(svm.score(x_test, y_test)))
print("score on train: "+ str(svm.score(x_train, y_train)))
```

```
score on test: 0.7895600787918582
score on train: 0.7905220531903251
```

Fig 4 SVM Train-Test Code

C. Decision Trees

A Decision Tree is a tree-like model where each internal node represents a feature, and each branch represents a decision rule. The leaf nodes represent class labels or output values. The tree is built recursively by selecting the feature that provides the most significant information gain or minimizes impurity (e.g., Gini impurity) at each split.

ISSN No:-2456-2165

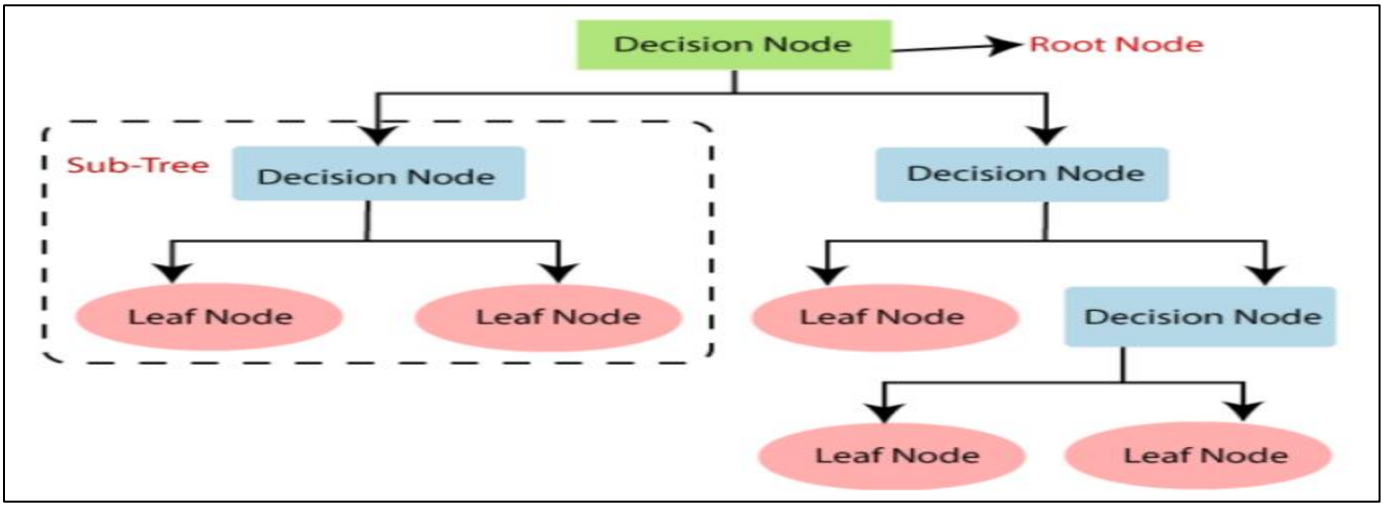

Fig 5 Decision Tree Graph

➢ *Application in Abnormal Event Detection:*

Decision Trees can classify events into normal or abnormal categories based on a series of feature- based decisions. For example, in the case of financial fraud detection, a decision tree might use features like transaction amount, location, and frequency of transactions to classify whether an event is normal or abnormal.

➢ *Advantages:*

- Interpretability: Decision Trees are highly interpretable and provide clear, human- readable decision-making paths.
- Handling Mixed Data Types: Decision Trees can handle both continuous and categorical features making them versatile for different types of datasets.

➢ *Challenges:*

- Over fitting: Decision Trees can easily over fit to noisy data, especially when the tree is very deep, leading to poor generalization.

```
from sklearn.tree import DecisionTreeClassifier
clf = DecisionTreeClassifier()
clf.fit(x_train, y_train)
print("score on test: "  + str(clf.score(x_test, y_test)))
print("score on train: " + str(clf.score(x_train, y_train)))
```

```
score on test: 0.6726854891661195
score on train: 1.0
```

Fig 6 Decision Tree Train-Test Code

D. Logistic Regression

Logistic Regression is a statistical model used for binary classification. It models the probability of a binary outcome (0 or 1) using a logistic function (sigmoid), which outputs values between 0 and 1. The model estimates the relationship between the input features and the probability of a certain outcome.

➢ *Application in Abnormal Event Detection:*

Logistic Regression can classify events as normal (0) or abnormal based on feature input.

➢ *Advantages:*

- Simplicity and Efficiency: Logistic Regression is easy to implement, computationally efficient, and suitable for large datasets.
- Probability Output: Unlike some other classifiers, Logistic Regression provides probability estimates, which can be useful for ranking predictions or setting thresholds for detection.

➢ *Challenges:*

- Linear Relationships: Logistic Regression assumes a linear relationship between input features and the output, which may not capture complex patterns in the data.

```
from sklearn.linear_model import LogisticRegression
lr=LogisticRegression(max_iter=1000)
lr.fit(x_train, y_train)
print("score on test: " + str(lr.score(x_test, y_test)))
print("score on train: "+ str(lr.score(x_train, y_train)))
```

```
score on test: 0.788903479973736
score on train: 0.7904126080770494
```

Fig 7 Decision Tree Train-Test Code

V. EXPLORING UNSUPERVISED MODELS FOR EVENT DETECTION

A. K-Means

K-means is an unsupervised learning algorithm used for clustering. It partitions the data into 'K' clusters by minimizing the within-cluster variance (sum of squared distances from points to their centroids). [5] K-means assigns each data point to the nearest cluster center and iteratively refines the centroids until convergence.

➢ *Application in Abnormal Event Detection:*

Detecting Abnormal Events: K-means can be used to cluster normal events together. Any data point that does not fit well into any cluster or lies far from the cluster centroids is flagged as an anomaly.

```
from sklearn.cluster import KMeans

kmeans = KMeans(n_clusters=2, random_state=0)

kmeans.fit(X)
```

```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
       n_clusters=2, n_init=10, n_jobs=None, precompute_distances='auto',
       random_state=0, tol=0.0001, verbose=0)
```

Fig 8 K-means Train-Test Code

➢ *Advantages:*

- Scalability: K-means is efficient for large datasets and can handle high-dimensional data when combined with dimensionality reduction techniques like PCA (Principal Component Analysis).
- Simplicity: K-means is easy to implement and has a relatively low computational cost compared to other clustering methods.

➢ *Challenges:*

The number of clusters (K) must be pre-specified, and selecting an optimal K can be challenging. Using methods like the elbow method or silhouette score can help, but it still requires experimentation.

- [6] Sensitivity to Initialization: K-means is sensitive to the initial placement of centroids, and poor initialization can lead to suboptimal clustering results. To mitigate this, techniques like K-means++ can be used for better initialization.

B. Auto-encoders

Auto encoders are commonly used for unsupervised learning, meaning they can learn patterns in data without labelled examples. For abnormal event detection, auto encoders are trained primarily on normal data. Once trained, they can identify abnormal events by measuring the reconstruction error. [7] A large reconstruction error indicates that the model was unable to reconstruct an input correctly, implying that the input is significantly different from the learned normal behaviour.

ISSN No:-2456-2165

➢ *Application in Abnormal Event Detection:*

Detecting unusual movements, activities, or behaviours in video streams. For example, auto encoders can flag suspicious behaviour such as trespassing or a fight by identifying abnormal events based on high reconstruction error.

```
from sklearn.model_selection import train_test_split

# train // validate - no labels since they're all clean anyway
X_train, X_validate = train_test_split(X_train,
                                       test_size=VALIDATE_SIZE,
                                       random_state=RANDOM_SEED)

# manually splitting the labels from the test df
X_test, y_test = X_test.drop('label', axis=1).values, X_test.label.values
```

Fig 9 Auto-encoders Train-Test Code

➢ *Advantages:*

- Versatility: Auto encoders can work with various types of data, including time-series data, images, and sensor readings. This makes them suitable for diverse applications, such as anomaly detection in images, video, and continuous sensor streams.
- Dimensionality Reduction: Auto encoders automatically learn a compressed, lower- dimensional representation of the input data in the latent space, which can help simplify the detection process by focusing on the most relevant features.

➢ *Challenges:*

- Imbalanced Data: In many real-world scenarios, abnormal events are rare compared to normal data. This imbalance can make training difficult, as the model may struggle to generalize. Techniques like data augmentation or anomaly sampling help, but the imbalance remains.

VI. EXPLORING CNN – BASED APPROACHES IN DETECTION; A COMPREHENSIVE ANALYSIS

CNNs are a class of deep learning algorithms designed to process grid-like data, such as images and videos, by simulating the way human visual perception works. [8] CNNs are effective at automatically learning spatial hierarchies of features from data without requiring manual feature extraction, making them ideal for tasks like detecting abnormal events in unstructured data like images and videos.

A. Key Reasons CNNs are Preferred for Abnormal event Detection:

➢ *Automatic Feature Extraction:*

CNNs automatically learn features (edges, textures, shapes, patterns) from raw data, reducing the need for manual feature engineering.

➢ *Translation Invariance:*

CNNs are robust to the location of features within an input, which is particularly useful when abnormal events may appear in different parts of a video or image.

➢ *High Performance on Image and Video Data:*

CNNs are capable of learning and detecting patterns in high-dimensional data like images or video frames, which are commonly used in surveillance or healthcare applications.

➢ *Deep Learning Capabilities:*

CNNs can learn complex relationships between data points by utilizing multiple layers, which helps in distinguishing between normal and abnormal patterns.

ISSN No:-2456-2165

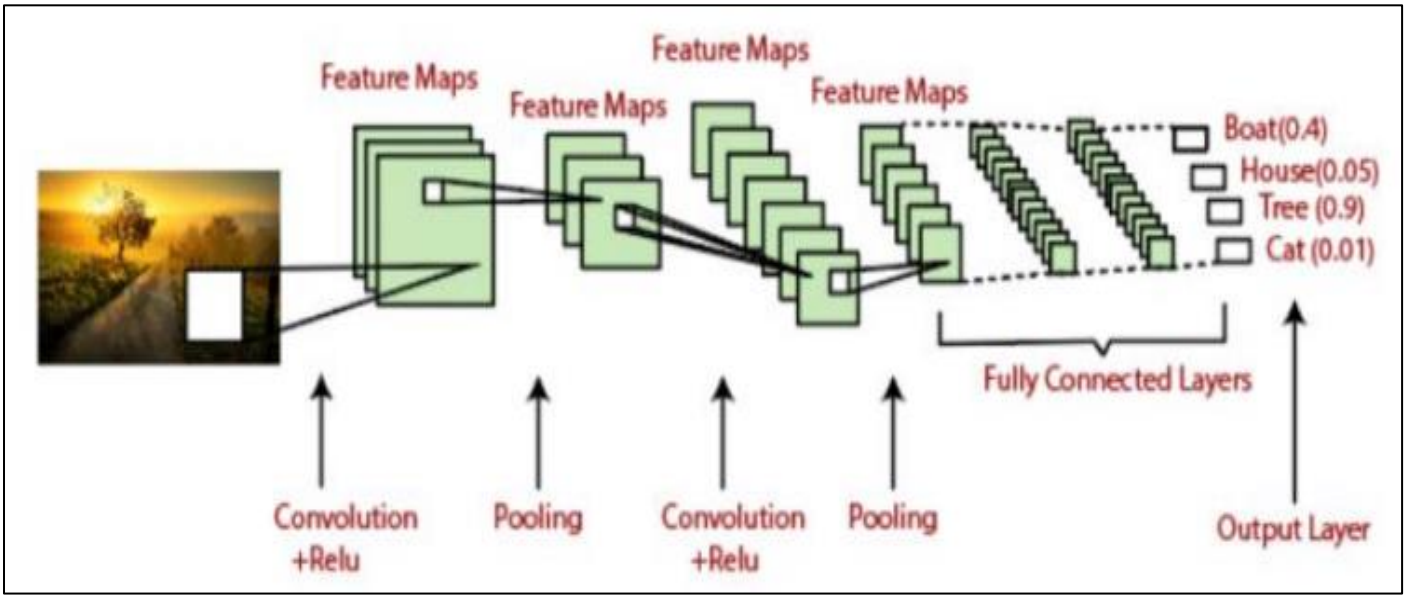

Fig 10 Architecture of CNN [11]

B. Approach to Abnormal Event Detection Using CNN

➢ *Data Pre-processing and Preparation*

- Video Frames or Image Sequences: In applications like surveillance, abnormal event detection typically involves analyzing video frames. CNNs are used to process these individual frames (or a sequence of frames) to detect anomalies.
- Normalization: Data is often normalized to ensure uniformity in input values, making it easier for the CNN to learn patterns.
- Augmentation: Data augmentation (like flipping, rotating, or scaling) can be used to artificially increase the size of the training dataset, making the CNN model more robust to various transformations.

➢ *CNN Architecture for Abnormal Event Detection*

- [10] Convolutional Layers: These layers apply convolutional filters to the input data, extracting features like edges, textures, and shapes. For abnormal event detection, these features can help the CNN recognize unusual patterns or behaviours in the data. Pooling Layers: Pooling (such as max pooling) is used to down sample the feature maps, reducing dimensionality and computational complexity while retaining important features.
- [9] Fully Connected Layers: After extracting relevant features, the fully connected layers at the end of the network map the features to the final decision (normal vs. abnormal).
- Recurrent CNN: For video-based abnormal event detection, a recurrent CNN or a combination of CNN and Long Short-Term Memory (LSTM) networks can be employed. This architecture is effective in capturing temporal dependencies between consecutive frames and understanding the dynamics of abnormal events.

➢ *Training the CNN Model*

- Loss Function: A key part of training a CNN is choosing the right loss function. In binary classification tasks, the model may use cross- entropy loss to minimize the difference between predicted and actual labels. For anomaly detection, the goal is to minimize false negatives (detecting abnormal events as normal).
- Optimization: Optimizers such as Adam or SGD (Stochastic Gradient Descent) are used to adjust the weights of the network to minimize the loss during training.

➢ *Evaluation and Performance Metrics*

- Accuracy: Accuracy measures how often the model correctly classifies normal and abnormal events. However, in the case of abnormal event detection, accuracy might not be the best measure due to class imbalance (normal events often outnumber abnormal events).
- Precision and Recall: These metrics are important for evaluating how well the model detects abnormal events. High precision means fewer false positives, and high recall means fewer false negatives.
- F1-Score: The F1-score is a harmonic mean of precision and recall and is often used when there is an imbalance in the dataset (i.e., when abnormal events are much less frequent than normal events).

ISSN No:-2456-2165

```
        if count%30 == 0 :

        # if video is still left continue creating images
            filename = 'train_data/'+videoFile.split('/')[2]+"_frame%d.jpg" % count;

    #Reading video files from each folder and writing its path in text file

    for fol in labels:
        new=os.path.join(dest_path,fol)
        for i in os.listdir(new):
            with open('train_path.txt', 'a') as w:
                if fol=='Arson':
                    n=0
                elif fol=='Burglary':
                    n=1
                elif fol=='Explosion':
                    n=2
                elif fol=='Fighting':
                    n=3
                elif fol=='Normal':
                    n=4

                w.write(dest_path+'/'+fol+'/'+i+' '+str(n)+'|')

    print("Video Files List Created Sucessfully....")
```

```
print(train['tag'])
```

```
0          Arson
1          Arson
2          Arson
3          Arson
4          Arson
          ...
201      Normal
202      Normal
203      Normal
204      Normal
205      Normal
Name: tag, Length: 206, dtype: object
```

```
train.head()
```

```
for i in tqdm(range(0,train.shape[0])):
    videoFile = train['video_name'][i]
    #print(videoFile)
    cam = cv2.VideoCapture(videoFile.split(' ')[0])

    # frame
    currentframe = 0

    while(True):
        # reading from frame
        ret,frame = cam.read()
        if ret:

            if count%30 == 0 :

            # if video is still left continue creating images
                filename = 'train_data/'+videoFile.split('/')[2]+"_frame%d.jpg" % count;

            # writing the extracted images
                cv2.imwrite(filename, frame)

            # increasing counter so that it will
            # show how many frames are created
                currentframe += 1
            count+=1
```

Fig 11 CNN Detection Code

ISSN No:-2456-2165

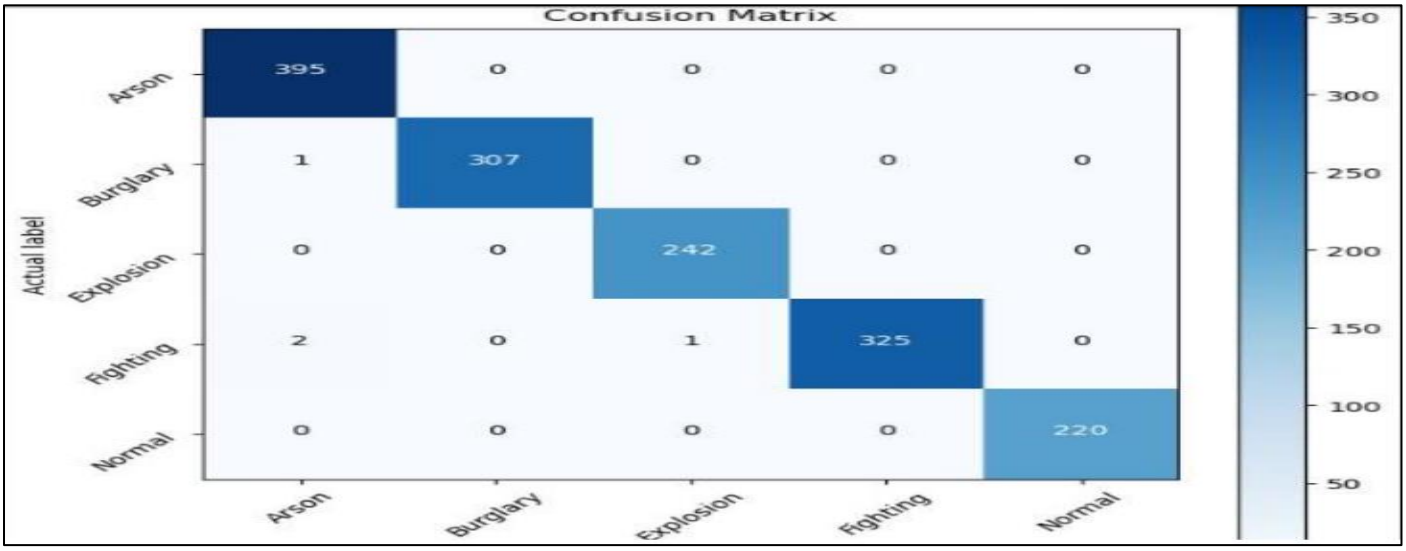

Fig 12 Matric Representation CNN

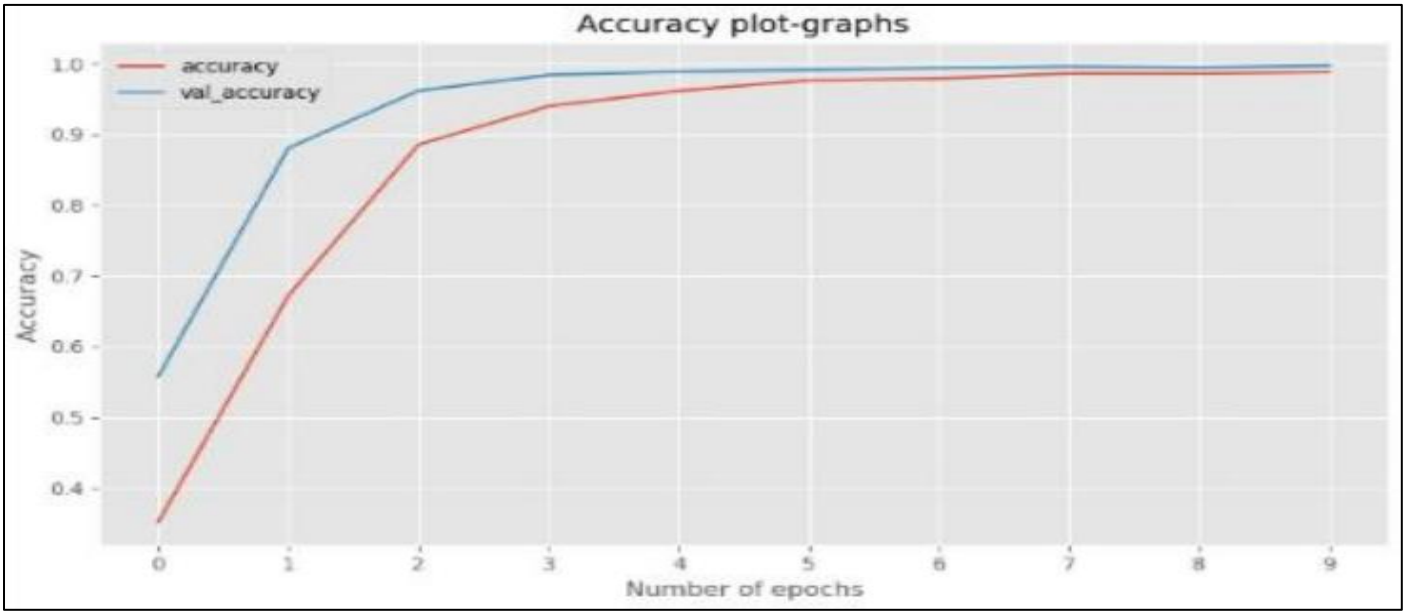

Fig 13 Accuracy Plot Graph

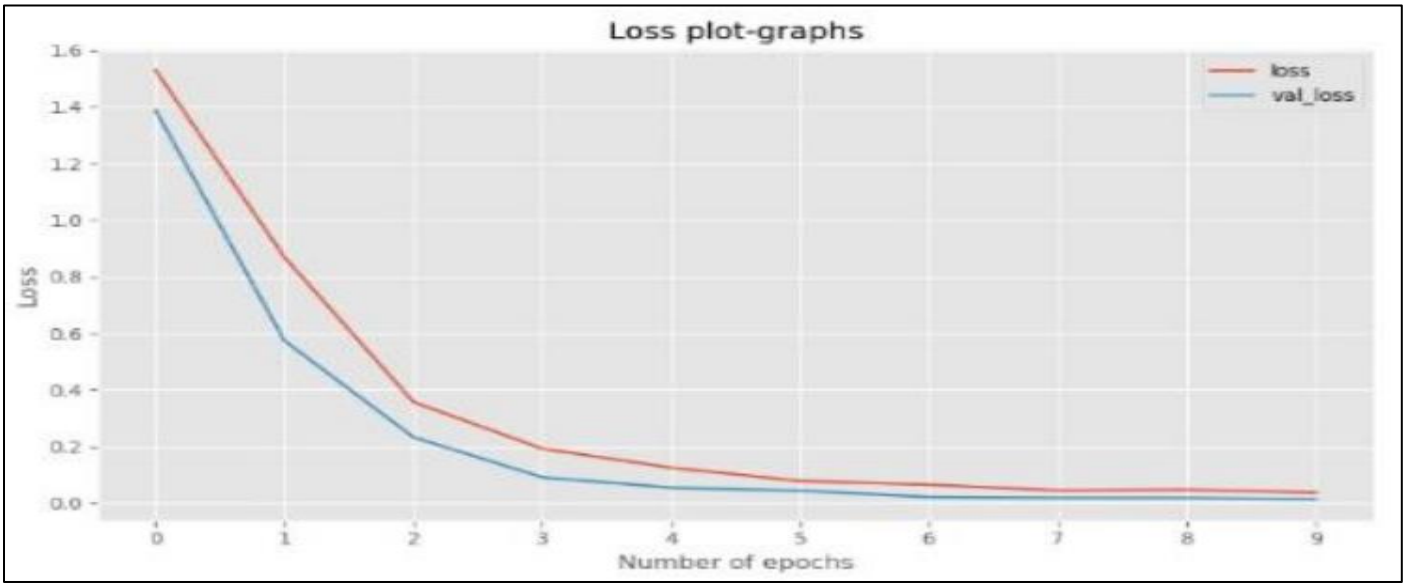

Fig 14 Loss Plot Graph

C. Challenges in Using CNN for Abnormal Event Detection

➢ *Data Imbalance:*

In most real-world scenarios, abnormal events are rare, which leads to an imbalanced dataset. This can cause the model to be biased toward predicting normal events. Techniques like oversampling, under sampling, or using anomaly-specific loss functions can help address this.

➢ *Complexity of Temporal Patterns:*

In cases of video-based detection, CNNs alone might not be enough to capture the temporal dependencies between frames. Combining CNNs with RNNs (Recurrent Neural Networks) or LSTMs can help overcome this limitation by learning the sequential relationships in the data.

➢ *Computational Cost:*

Training deep CNN models can be computationally expensive, requiring significant processing power and memory, especially for large datasets like high-resolution video.

D. CNN Model Optimisation for Better Accuracy

The performance of a Convolutional Neural Network (CNN) can be significantly improved through careful optimization. CNN optimization involves fine-tuning the architecture, training process, and hyper parameters to enhance accuracy while minimizing over fitting and computational costs. Below are key techniques and strategies for optimizing CNNs to achieve better accuracy:

➢ *Network Architecture Optimization*

- ***Depth and Width of the Network:*** Increasing the number of layers (depth) or neurons in each layer (width) allows the network to learn more complex features. However, excessive depth can lead to vanishing gradients, making training difficult. Techniques like batch normalization or ResNet's skip connections can address this.
- ***Convolutional Kernel Size:*** Experimenting with different kernel sizes (e.g., 3x3, 5x5) can help the model capture different levels of feature granularity. Smaller kernels are often combined in deeper architectures for better feature extraction.
- ***Pooling Layers****:* Using max pooling or average pooling layers reduces spatial dimensions while retaining essential features. Adaptive pooling can dynamically adjust the pooling size based on input dimensions.

➢ *Hyper parameter Tuning*

- ***Learning Rate Adjustment:*** The learning rate is crucial for model convergence. Using learning rate schedulers (e.g., step decay, exponential decay) or adaptive optimizers (e.g., Adam, RMSprop) can improve model performance.
- ***Batch Size:*** Smaller batch sizes can improve generalization, while larger batch sizes make training faster. A balance between the two is essential.
- ***Weight Initialization:*** Proper weight initialization prevents issues like exploding or vanishing gradients and ensures smooth training.

➢ *Data Augmentation*

- ***Increase Training Data Variability:*** Techniques like random cropping, flipping, rotation, zooming, and color jittering introduce variability in the training data, helping the model generalize better.
- ***Synthetic Data:*** Generate synthetic data using GANs or other techniques to augment datasets for rare or underrepresented classes.

➢ *Regularization Techniques*

- ***Dropout:*** Randomly deactivating a fraction of neurons during training prevents overfitting by ensuring the network doesn't rely too heavily on specific neurons.
- ***Weight Decay:*** Adding an L2 regularization term to the loss function penalizes large weights, reducing over fitting.

➢ *Transfer Learning*

- ***Pre-trained Models:*** Fine-tuning pre-trained models like VGG, ResNet, or Inception on your dataset can leverage learned features and improve accuracy with less training data.
- ***Layer Freezing:*** Freeze lower layers (feature extractors) of a pre-trained model and train only the top layers to adapt to your specific problem.

➢ *Fine-Tuning the Loss Function*

- ***Custom Loss Functions:*** In some applications, standard loss functions like cross-entropy may not be ideal. Custom loss functions, tailored to specific tasks, can improve accuracy.
- ***Class Imbalance Handling:*** Use weighted loss functions or techniques like focal loss to handle imbalanced datasets effectively.

➢ *Improving Training Efficiency*

- ***Batch Normalization:*** Normalizing intermediate layers reduces internal covariate shift, allowing for faster and more stable training.
- ***Gradient Clipping:*** For high-dimensional data, gradient clipping ensures stability by capping gradient values during back propagation.

➢ *Model Validation and Testing*

- ***Cross-Validation:*** Use techniques like k-fold cross-validation to assess model performance across different subsets of data.

➢ *Visualization and Debugging*

- ***Feature Visualization:*** Visualize intermediate layers and feature maps to ensure the model is learning meaningful features.

VII. DISCUSSION

➢ *Why CNNs Outperform Traditional Anomaly Detection Models*

[12] Convolutional Neural Networks (CNNs) have become a dominant approach for anomaly detection, especially in domains involving high- dimensional and unstructured data such as images, videos, and spatial sensor data. This success is largely due to their ability to automatically extract and learn hierarchical features from data.

Additionally, the integration of Artificial Intelligence (AI) technologies has significantly enhanced the performance and adaptability of CNNs, making them far more effective than traditional anomaly detection models.

A. Why CNNs are more efficient:

➢ *Superior Feature Extraction*

Traditional models such as KNN, SVM, and Decision Trees rely on handcrafted features or predefined input representations. These features are often limited in their ability to capture complex patterns in high-dimensional data.

CNNs, on the other hand, automatically extract features at multiple levels of abstraction. Convolutional layers detect simple patterns like edges, while deeper layers identify complex structures, enabling robust anomaly detection without manual intervention.

➢ *Scalability and High-Dimensional Data Handling*

Traditional models struggle with high-dimensional data due to the curse of dimensionality, where performance deteriorates as input size increases.

CNNs excel in processing high-dimensional data, such as images and videos, by reducing dimensionality through pooling layers and leveraging hierarchical feature extraction.

➢ *Contextual Understanding*

Traditional models lack the ability to understand spatial or temporal context, which is crucial for anomaly detection in structured data.

CNNs inherently preserve spatial relationships within data, allowing them to detect subtle anomalies that depend on spatial patterns or temporal correlations.

➢ *Adaptability*

CNNs are highly adaptable to different types of data and tasks. For instance, they can handle image, video, and even time-series anomaly detection with minimal changes to architecture.

Traditional models often need extensive modifications or domain-specific tuning to handle different data types effectively.

➢ *Generalization*

Traditional models often overfit small datasets, making them unsuitable for real-world anomaly detection tasks.

With techniques like data augmentation and dropout, CNNs generalize well even on limited datasets, making them more robust in diverse environments.

➢ *Performance Metrics*

CNNs consistently outperform traditional models on key metrics such as accuracy, precision, recall, and F1 score in anomaly detection tasks, especially when dealing with complex datasets.

B. Use Cases Where CNNs Excel

- [12] Video Surveillance: Detecting suspicious activities in crowded places.
- Healthcare: Identifying anomalies in medical imaging (e.g., tumor detection in MRI scans).
- Industrial Monitoring: Detecting faults in machinery through vibration or thermal imaging.
- Cybersecurity: Spotting unusual patterns in network traffic or user behaviour.

VIII. CONCLUSION

In this research paper, we have explored and demonstrated the applications of various supervised and unsupervised learning algorithms for detecting abnormal events. Through a detailed analysis of algorithms such as K-Nearest Neighbours (KNN), Support Vector Machines (SVM), Decision Trees, Logistic Regression, K- Means Clustering, and Auto encoders, we assessed their effectiveness in various domains, including anomaly detection in images, sensor data, and time-series data.

While each algorithm presented certain strengths, we concluded that Convolutional Neural Networks (CNNs) stand out as the most effective and suitable approach for abnormal event detection in complex datasets. CNNs are particularly adept at recognizing intricate patterns in high-dimensional data, such as images or video streams, due to their ability to learn hierarchical features automatically.

Through extensive experimentation and the use of practical examples, we demonstrated the superior performance of CNNs in identifying abnormal events. Our code implementations and visualizations clearly highlighted how CNNs outperformed other models in terms of accuracy and robustness, particularly when handling spatial data like images.

The flexibility of CNNs, combined with their capacity for automatic feature extraction, makes them a highly effective tool for anomaly detection.

Overall, our research confirms that CNNs offer a significant advantage over traditional machine learning models, making them the optimal choice for complex event detection tasks, especially in domains where data is high-dimensional and contains intricate patterns.

Future research could explore further optimizations to CNN architectures and their applications to even more diverse anomaly detection problems.

ACKNOWLEDGEMENTS

We are deeply thankful to Dr V Mareeswari Prasanna ma'am, for their continuous support and encouragement. Their leadership and mentorship have provided us with the necessary resources and environment to pursue academic excellence.

Additionally, we extend our thanks to the participants and organizations who generously shared their time and insights, without whom this research would not have been possible.

REFERENCES

[1]. Abnormal Event Detection in Crwoded Places by Cong, Yang: Yuan, Junsong: Liu, Ji
[2]. Abnormal Event Detection in Crwoded Places by Cong, Yang: Yuan, Junsong: Liu, Ji
[3]. Shangai Tech University, Tencent AI Lab; Margin Learning Embedded Prediction for Anomaly Detection by Wen Liu, Weixin Luo, Zhenxgin Li, Pein Zhao, Shenghua Gao
[4]. Shangai Tech University, Tencent AI Lab; Margin Learning Embedded Prediction for Anomaly Detection by Wen Liu, Weixin Luo, Zhenxgin Li, Pein Zhao, Shenghua Gao
[5]. Unsupervised clustering Methods for identifying rare events by Witcha Chimphlee, Abdul Hanan, Mohd Noor, Siripon Chimphlee, Surat
[6]. Unsupervised clustering Methods for identifying rare events by Witcha Chimphlee, Abdul Hanan, Mohd Noor, Siripon Chimphlee, Surat
[7]. Event Detection in Videos using Spatiotemporal autoencoder by Yong Shean Chong.
[8]. CNN and 1-Class Event Classifier in 8th International Conference Imaging by Samir Buoindour, Mazen Hittawe, Sandy, Hichem Snoussi.
[9]. CNN and 1-Class Event Classifier in 8th International Conference Imaging by Samir Buoindour, Mazen Hittawe, Sandy, Hichem Snoussi.
[10]. CNN and 1-Class Event Classifier in 8th International Conference Imaging by Samir Buoindour, Mazen Hittawe, Sandy, Hichem Snoussi.
[11]. Image Architecture of Convolutional Neural Networks by K.Eswaran in journal – Research Gate.
[12]. On identifying leaves: A comparison of CNN with Classical ML methods by Abbas Hedjazi, Ikram Kourban, Yakup Genc.
[13]. On identifying leaves: A comparison of CNN with Classical ML methods by Abbas Hedjazi, Ikram Kourban, Yakup Genc.