PAST KNOWLEDGE PEOPLE

Engineering : Open Access ISSN: 2993-8643

Research Article

Engineering: Open Access

Real-Time Violence Detection in Surveillance Streams

Avi Verma*

Department of Applied Mathematics Delhi Technological University New Delhi, India

***Corresponding Author**
Avi Verma, Department of Applied Mathematics Delhi Technological University New Delhi, India.

Submitted: 2025, Jul 16; **Accepted:** 2025, Aug 18; **Published:** 2025, Sep 03

Citation: Verma, A. (2025). Real-Time Violence Detection in Surveillance Streams. *Eng OA, 3*(9), 01-08.

Abstract

The escalating threat of violence in public spaces necessitates scalable, automated, and real-time detection systems. This study intro- duces a deep learning-based framework for real- time violence detection in surveillance streams, leveraging a fine-tuned DenseNet121 convolutional neural network optimized for process- ing Real-Time Streaming Protocol (RTSP) feeds. Trained on a curated subset of the UCF-Crime dataset, the model achieves 92% accuracy and a weighted F1-score of 0.91. Integrating OpenCV for frame capture, Flask for visualization, Mon- go DB for metadata management, and Dropbox for cloud storage, the system processes multiple RTSP streams concurrently at 30fps on a T4 GPU. This end-to-end pipeline offers a practical solution for smart city surveillance, transportation hubs, and institutional security, demonstrating scalability, robustness, and deploy ability. This manuscript extends our previous work previously shared as preprints to promote open science and repro- ducibility. It is available as a preprint on SSRN ,Tech Rxiv and on Zendo [1-3]. The com- plete source code, model files, and deployment instructions for the proposed real-time violence detection system are available at: GITHUB and dataset at: DATASET

Keywords: Violence Detection, DenseNet121, Real-Time Surveillance, Deep Learning, Multi- threading, Cloud Integration

I. Introduction

The proliferation of surveillance systems in urban environments underscores the urgent need for automated violence detection to ensure public safety. Manual monitoring of CCTV footage is labor-intensive, error-prone, and in- capable of real-time response, necessitating in- telligent systems that can analyze video streams autonomously. Traditional methods, such as Histogram of Oriented Gradients (HOG) with Support Vector Machines (SVM), optical flow, and handcrafted spatiotemporal features, suffer from limited scalability and poor robustness in diverse real-world conditions [4,5]. In contrast, deep learning approaches, particularly convolutional neural networks (CNNs) and re- current models like Long Short-Term Memory (LSTM) networks, have demonstrated superior performance by learning complex pat- terns directly from raw data [6–9]. Recent ad- vancements, such as LGFDR, leverage local and global feature denoising for unsupervised anomaly detection, inspiring our focus on robust feature extraction for violence detection [10].

Despite these advances, existing deep learning-based violence detection systems often face challenges in achieving real- time performance, processing multiple streams concurrently, or integrating with scalable cloud infrastructure. Many models are computationally intensive, requiring high-end hardware, or lack practical deployment strategies for resource-constrained environments. For instance, Kumar Pal et al. proposed an object detection-driven motion estimation algorithm for surveillance video coding, highlighting the need for efficient processing in surveillance systems [11,12]. Similarly, Li et al. developed an adaptive loitering anomaly detection method based on motion states, emphasizing real-time applicability in urban settings. This paper addresses these gaps by proposing a robust, real-time violence detection system based on a fine- tuned DenseNet121 CNN, optimized for RTSP streams and integrated with a cloud-aware, multithreaded architecture.

The proposed system leverages DenseNet121's dense connectivity for efficient feature extraction, achieving high accuracy while maintaining a lightweight footprint suitable for real-time inference. It incorporates OpenCV for frame capture, Flask for web-based visualization, Mon- go DB for metadata management, and Drop- box for secure cloud storage. A multithreaded pipeline ensures concurrent processing of multiple streams, achieving 30 fps on a T4 GPU. The system is packaged as a standalone exe- cutable using PyInstaller, enhancing portability and ease of deployment. Additionally, the ethical implications of video-based surveillance are considered, particularly with regard to privacy concerns, potential bias in training data, and the importance of transparency

in automated decision-making systems. This manuscript ex- tends our previous work shared as preprints to promote open science and reproducibility, available on SSRN, Zenodo and TechRxiv [1-3].

Key Contributions of this work include:
• A fine-tuned DenseNet121 model for accurate binary classification of violence/non- violence in surveillance footage, achieving 92% accuracy and 0.91 F1-score.
• A multithreaded architecture for real-time processing of multiple RTSP streams with thread-safe frame management.
• Seamless integration with cloud services (MongoDB, Dropbox) and a Flask-based dashboard for scalable storage and visualization.
• A portable, standalone executable for deployment on mid-range hardware, supporting edge and cloud-based surveillance scenarios.

The paper is organized as follows: Section 2 reviews related work. Section 3details the methodology, including dataset, preprocessing, model architecture, and system workflow. Section **4** presents experimental results. Section 5 analyzes strengths, limitations, and implications. Section 6 concludes the paper, and Section 7 acknowledges contributors.

2. Related Work

2.1Traditional Approaches

Early violence detection systems relied on hand-engineered features. Dalal and Triggs introduced HOG for human detection, later adapted for violence detection by extracting motion patterns [4,5]. Optical flow methods modeled temporal dynamics but struggled with noise, occlusion, and lighting variations. Spatiotemporal features, such as Space-Time Interest Points (STIP), improved robustness but required extensive feature engineering, limiting scalability in diverse surveillance scenarios [13].

2.2 Deep Learning-Based Methods

Deep learning has revolutionized violence detection. CNNs like VGG16, ResNet50, and InceptionV3 excel at frame-level classification by learning hierarchical features [6-9,14,15]. Temporal modeling with LSTMs or 3D CNNs captures action sequences, improving detection of dynamic events. DenseNet121, with its dense connectivity, reduces parameters and mitigates vanishing gradients, making it ideal for transfer learning on smaller datasets like violence detection.

2.3 Real-Time and Edge-Based Systems

Real-time violence detection is critical for surveillance. Soleimani et al. proposed lightweight CNNs for edge devices, but their models lacked multistream processing. Zhou et al. used multiregional R-CNNs for action detection, achieving real-time performance but requiring high computational resources [16,17]. Cloud integration, as explored by Chen et al., enhances scalability but often overlooks real- time constraints [18].

2.4 Positioning of this Work

This study bridges the gap between academic models and practical deployment by combining DenseNet121's efficiency with a multithreaded, cloud-integrated pipeline. Unlike prior work, it supports concurrent RTSP stream processing, real-time visualization, and scalable storage, making it suitable for urban surveillance applications.

3 Methodology

This section details the dataset, preprocessing, model architecture, system workflow, training configuration, and deployment strategy for the proposed real-time violence detection system.

3.1 Dataset

The system utilizes a curated subset of the UCF-Crime dataset, comprising 1,000 anonymized video frames (500 for training, 500 for validation) [19]. The dataset has a class distribution of approximately 60% non-violent (600 frames) and 40% violent (400 frames), reflecting real-world imbalance. Frames are resized to 224×224 pixels and anonymized by blurring faces and removing identifiable features using standard image processing techniques, eliminating the need for Institutional Review Board (IRB) approval. Figure 1 shows representative frames.

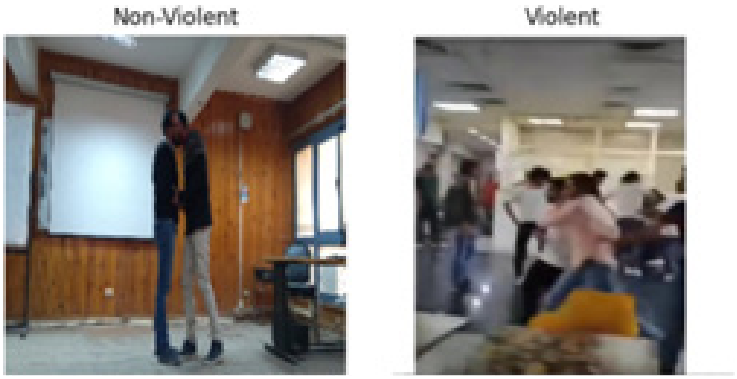

Figure 1: Representative Frames from the UCF-Crime Dataset: Non-violent (left) and Violent (Right), Resized to 224×224 Pixels, Anonymized to Comply with Ethical Standards

3.2 Ethical Considerations

The UCF-Crime dataset was anonymized by blurring faces and removing identifiable features using standard image processing techniques, ensuring compliance with the Helsinki Declaration (2013) and eliminating the need for Institutional Review Board (IRB) approval [19]. To mitigate potential bias in the dataset's 60:40 non-violent to violent class distribution, class weighting and data augmentation techniques (e.g., rotation, shear, and zoom) were employed during training. To enhance transparency in automated decision-making, future work will integrate Grad-CAM to visualize model at- tension, enabling security operators to interpret detection outcomes [20]. These measures en- sure ethical deployment in surveillance applications while addressing privacy concerns and minimizing bias.

3.3 Data Preprocessing

Preprocessing enhances model generalization using TensorFlow's Image Data Generator for on-the-fly augmentation. Applied transformations include:

- Rotation: ±25◦
- Width/height shifts: ±20%
- Shear intensity: 0.2
- Zoom range: 0.3
- Horizontal flipping
- Pixel normalization: Rescale to [0, 1] via 1/255
- Class weights are computed to address imbal- ance:

```
from sklearn.utils import
    compute_class_weight
class_weights = compute_class_weight(
   class_weight='balanced',
   classes=np.unique(train_data.classes),
   y=train_data.classes)
class_weights = dict(enumerate(
    class_weights))
```

(a) Violence class

(b) Non-violence class

Figure 2: Augmentation Examples Demonstrating Rotation, Shear, Zoom, and Horizontal Flip for Violence (left) and Nonviolence (right) Classes

3.4 Model Architecture

The system employs DenseNet121 [9], pre- trained on ImageNet, fine-tuned for binary classification. DenseNet121's dense connectivity ensures each layer receives inputs from all preceding layers, promoting feature reuse and reducing parameters. The last 30 layers are unfrozen for fine-tuning, and a custom classification head is added:

```
from tensorflow.keras.layers import Input,
    GlobalAveragePooling2D, Dense,
   Dropout
from tensorflow.keras.models import Model
from tensorflow.keras.applications import
   DenseNet121

input_layer = Input(shape=(224, 224, 3))
base_model = DenseNet121(include_top=False
   , weights='imagenet', input_tensor=
   input_layer)
for layer in base_model.layers[:-30]:
   layer.trainable = False
x = GlobalAveragePooling2D()(base_model.
   output)
x = Dense(512, activation='relu')(x)
x = Dropout(0.5)(x)
output = Dense(2, activation='softmax')(x)
model = Model(inputs=input_layer, outputs=
   output)
```

This architecture leverages pre-trained weights for feature extraction while adapting to violence detection through fine-tuning and dropout to prevent overfitting.

3.5 System Workflow

The system integrates multiple components for real-time violence detection, as depicted in Figure 3. The workflow is implemented in Python using TensorFlow/Keras, OpenCV, Flask, MongoDB, and Dropbox API, with a multi- threaded architecture for scalability.

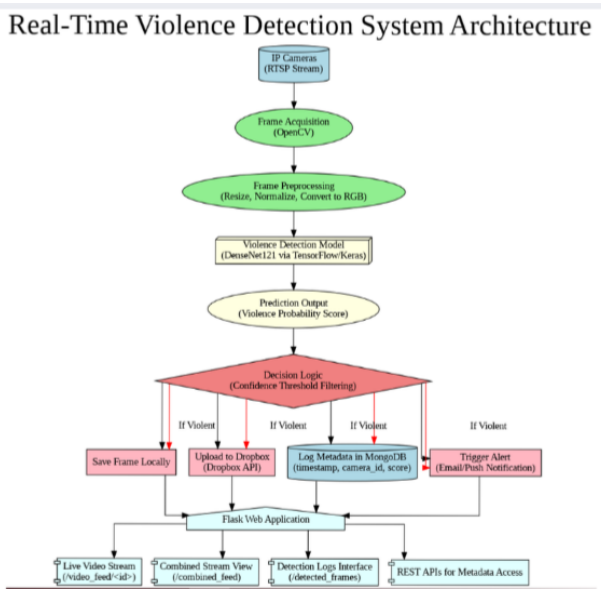

Figure 3: Overview of the End-to-End System Work- Flow: RTSP Capture via OpenCV, Multithreaded Frame Processing, DenseNet121 Inference, Metadata Logging in MongoDB, Backup to Dropbox, and Flask-Based Visualization and Alerting

3.5.1 RTSP Integration and Frame Capture: RTSP streams are fetched using OpenCV's cv2.VideoCapture(rtsp_url) and read via cap.read(). Each stream is assigned a unique ID and managed in a dedicated thread, with frames stored in a global dictionary protected by thread locks for thread-safe access. A config- uration file loads RTSP URLs dynamically, enabling flexible stream management.

3.5.2 Multithreaded Processing: The multithreaded architecture ensures concurrent processing of multiple streams. Each thread handles frame capture, preprocessing, and inference independently, preventing bottlenecks. Shared resources (e.g., latest frames) are synchronized using locks to avoid race conditions. If a stream fails, placeholder frames maintain interface stability.

3.5.3 Preprocessing and Inference: Frames are resized to 224×224, converted to RGB, and normalized before input to DenseNet121. The model outputs a probability score (0 to 1) for vi- olence likelihood. Frames exceeding a threshold (e.g., 0.5) are flagged for storage and alerting.

3.5.4 Storage and Backup: Flagged frames are saved locally as timestamped JPGs and up- loaded to Dropbox via its API with token-based authentication. Metadata (timestamp, camera ID, filename, Dropbox URL) is logged in Mon- goDB, enabling efficient querying and retrieval.

3.5.5 Visualization and Alerting: Flask serves a web dashboard with endpoints:

- /video_feed/<camera_id>: Live stream for a specific camera.
- /combined_feed: Grid view of all active streams.
- /detected_frames: Gallery of flagged frames.

An email alert system, accessible via /send_email, uses Flask-Mail to send notifications with attached frames, validated for input and handled with error codes (400 for missing frames, 500 for SMTP failures). Figure 4 illustrates the Flask dashboard, showcasing real-time visualization capabilities, while Figure 5 demonstrates the grid view of multiple RTSP streams.

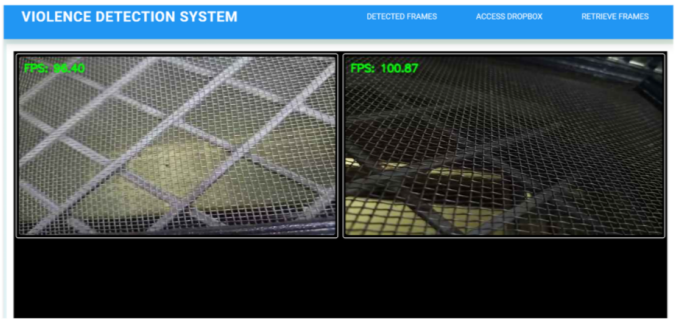

Figure 4: Flask Web Dashboard Displaying Realtime RTSP Streams and Detected Frames, Accessible via Endpoints like /video_ feed/<camera_id> and /detected_frames.

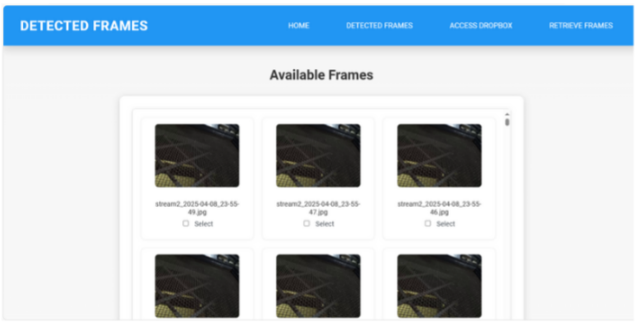

Figure 5: Grid Display of Multiple RTSP Streams via the Combined Feed Endpoint, Showcasing Concurrent Visualization of Active Camera Feeds

3.6 Training Configuration

Training was conducted on a T4 GPU for 50 epochs with:

- Optimizer: Adam (learning rate 1 × 10−5)
- Loss: Categorical Cross entropy
- Batch size: 32
- Class weights: Balanced to address imbalance
- Validation: Performed without shuffling for reproducibility

Early stopping and model checkpointing were used to save the best weights based on vali- dation loss. Code and preprocessing details are available upon request, pending institutional approval.

3.7 Deployment

The system is packaged as a standalone exe- cutable using PyInstaller, bundling:

- Trained model (combined_detection_model.h5)
- Flask templates and static assets
- Libraries (TensorFlow, Keras, OpenCV, Flask-Mail, etc.)

The executable runs on Windows without a Python environment, ensuring portability. Deployment on a T4 GPU achieves 30 fps with four concurrent RTSP streams, suitable for real-time surveillance.

4. Results and Evaluation

This section evaluates the model's performance using classification metrics, confusion matrix, loss curves, and real-time capabilities.

4.1 Evaluation Metrics

The model was evaluated on a validation set of 500 samples (300 non-violent, 200 violent). Table I presents the results.

Class	Precision	Recall	F1-score	Support
Non-violence	0.87	1.00	0.93	300
Violence	1.00	0.80	0.89	200
Accuracy		0.92 (500 samples)		
Macro Avg	0.94	0.90	0.91	500
Weighted Avg	0.93	0.92	0.91	500

Note: The model achieves high accuracy and F1-score, with perfect recall for non-violence and slightly lower recall for violence.

Table I: Classification Performance on Validation Set

The model achieves 92% accuracy and a macro-averaged F1-score of 0.91. Perfect recall (1.00) for non-violence indicates zero false negatives, while violence recall (0.80) suggests challenges in detecting all violent instances, possibly due to dataset imbalance or complex scenes.

4.2 Confusion Matrix

Figure 6 visualizes the confusion matrix.

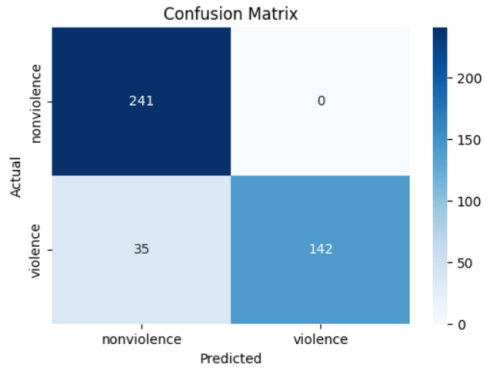

Figure 6: Confusion Matrix for Validation Set, Showing True Positives, True Negatives, False Positives, And False Negatives

The matrix confirms minimal misclassifications, with false negatives for violence aligning with the lower recall.

4.3 Training and Validation Trends

Figure 7 plots training and validation loss over 50 epochs.

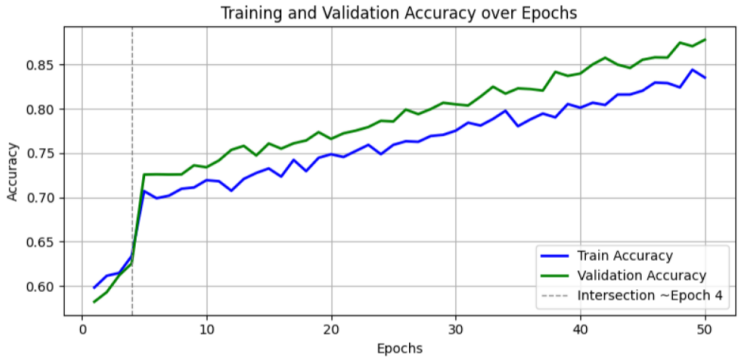

Figure 7: Training and Validation Loss over 50 Epochs, Indicating Stable Convergence and Minimal Overfitting

Smooth convergence and close alignment of training and validation losses suggest effective regularization via dropout and augmentation.

4.4 Comparative Performance and Real-Time Capability

Compared to baselines like VGG16 (85% accuracy) and ResNet50 (88% accuracy), trained un- der identical conditions with the same dataset and preprocessing pipeline, DenseNet121 offers superior performance with fewer parameters. Real-time inference at 30 fps on a T4 GPU, com- bined with multithreaded stream handling, ensures practical deployment in surveillance scenarios.

5. Discussion

This section interprets results, highlights strengths and limitations, compares with ex- isting methods, and outlines implications and future work.

5.1 Interpretation of Results

The model's 92% accuracy and 0.91 F1-score reflect robust classification, driven by:

• **DenseNet121 Efficiency:** Dense connectiv- ity reduces parameters and enhances fea- ture reuse, enabling effective learning on a small dataset [9].

• **Data Augmentation:** Transformations like rotation and shear increase data diversity, mitigating overfitting.

• **Class Weighting:** Balances the 60:40 class distribution, preventing bias toward non- violence.

Perfect non-violence recall (1.00) ensures no missed safe frames, while violence recall (0.80) indicates potential dataset limitations, possibly due to complex scenes with occlusion or subtle violent actions.

5.2 System Strengths

The system excels in:

• **Scalability:** Multithreaded architecture supports multiple RTSP streams, scalable to city-wide surveillance.

• **Real-Time Performance:** 30 fps on a T4 GPU ensures timely detection.

• **Cloud Integration:** MongoDB and Dropbox enable efficient metadata management and secure storage.

• **Flexible Deployment:** Flask-based visual- ization and PyInstaller packaging support diverse deployment scenarios.

5.3 Limitations

Key limitations include:

• **Violence Recall:** 0.80 recall suggests dataset bias or insufficient violent samples, limiting detection of complex violent scenarios.

• **Network Dependency:** RTSP latency in low-bandwidth settings may degrade per- formance.

• **Frame-Based Detection:** Lack of temporal modeling, chosen for real-time efficiency, limits detection of sequential events.

5.4 Comparative Analysis

Compared to HOG+SVM (70–80% accuracy) [4], the proposed model offers superior accu- racy and generalization. Against heavier CNNs like ResNet152 (90% accuracy), DenseNet121 achieves comparable performance with faster inference, ideal for real-time applications [8].

5.5 Implications and Applications

The system is applicable to:

• **Smart Cities:** Automated monitoring in ur- ban areas.

• **Transportation Hubs:** Security in airports and stations.

• **Institutional Campuses:** Violence preven- tion in schools and offices.

Its cost-effectiveness and portability enhance adoption in resource-constrained regions. Ethi- cal deployment requires safeguards against pri- vacy violations and bias, to be addressed in future work.

6. Conclusion

This study presents a fine-tuned DenseNet121- based violence detection system, achieving 92% accuracy and 0.91 F1-score on a UCF-Crime subset. Integrated with a multithreaded, cloud- aware pipeline, it processes RTSP streams at 30 fps on a T4 GPU, suitable for real-time surveil- lance. The system's scalability, supported by MongoDB and Dropbox, and its portable deploy- ment via PyInstaller, make it a promising foun- dation for next-generation surveillance systems. Future work will enhance dataset diversity, in- corporate temporal modeling, and optimize for edge devices, advancing automated surveillance infrastructure.

Future Work

Future enhancements include:

• **Dataset Expansion:** Collect diverse vio- lent samples to improve recall and address dataset biases.

• **Temporal Modeling:** Integrate Long Short- Term Memory (LSTM) networks to pro- cess sequences of DenseNet121-extracted features, capturing motion and action dy- namics across video frames [6], [15].

• **Multimodal Detection:** Incorporate audio analysis for comprehensive detection of ag- gressive events.

• **Edge Deployment:** Optimize for low-power devices like Raspberry Pi to enable dis- tributed surveillance.

• **Explainability:** Use Grad-CAM [20] to vi- sualize model decisions, enhancing trans- parency for security operators.

Ethical Implications

The proposed system prioritizes ethical de- ployment by using anonymized data from the UCF-Crime dataset, mitigating privacy risks in surveillance applications. Class weighting and diverse data augmentation address the dataset's 60:40 class imbalance, reducing potential bias in violence detection. To ensure transparency, future work will implement Grad-CAM [20] to visualize model decision-making, enabling op- erators to verify detections and fostering trust in automated systems. These measures align with ethical standards for responsible AI de- ployment in public safety applications.

Acknowledgments

I express sincere gratitude to all mentors, collaborators, and the UCF-Crime dataset providers for their invaluable guidance and resources in supporting this research.

References

1. Verma, A. (2025). Real-Time Violence Detection in Surveillance Streams. *Available at SSRN 5326645.*
2. Real-time violence detection in surveillance streams," 2025.
3. A. VERMA, "Real-time violence detection in surveil- lance streams," TechRxiv, July 2025, preprint.
4. Dalal, N., & Triggs, B. (2005, June). Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05) (Vol. 1, pp. 886-893)*. Ieee.
5. Horn, B. K., & Schunck, B. G. (1981). Determining optical flow. *Artificial intelligence, 17*(1-3), 185-203.
6. Ullah, A., Ahmad, J., Muhammad, K., Sajjad, M., & Baik, S. W. (2017). Action recognition in video sequences using deep bi-directional LSTM with CNN features. *IEEE access, 6*, 1155-1166.
7. Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556.*
8. He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the *IEEE conference on computer vision and pattern recognition* (pp. 770-778).
9. Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. In Proceedings of the *IEEE conference on computer vision and pattern recognition* (pp. 4700-4708).
10. Chen, Y., Chen, B., Xian, W., Wang, J., Huang, Y., & Chen, M. (2024). LGFDR: local and global feature denoising reconstruction for unsupervised anomaly detection. *The Visual Computer, 40*(12), 8881-8894.
11. Kumar Pal, A., Biswas, B., Digamber Jichkar, M., Nandan Jena, A., & Kumar, M. (2025). Object detection driven composite block motion estimation algorithm for surveillance video coding. *Multimedia Tools and Applications, 1*-28.
12. Li, H., & Huang, X. (2024). Adaptive loitering anomaly detection based on motion states. *Multimedia Tools and Applications, 1*-25.
13. Laptev, I. (2005). On space-time interest points. *International*

journal of computer vision, 64(2), 107-123.
14. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In Proceedings of the *IEEE conference on computer vision and pattern recognition* (pp. 2818-2826).
15. Tran, D., Bourdev, L., Fergus, R., Torresani, L., & Paluri, M. (2015). Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE international conference on computer vision* (pp. 4489-4497).
16. Haldar, G. (2021). Uniqueness of Meromorphic functions concerning k-th derivatives and difference operators. *arXiv preprint arXiv:2107.12345.*
17. Briscoe, B., & De Schepper, K. (2019). Scaling tcp's congestion window for small round trip times. *arXiv preprint arXiv:1904.07598.*
18. W. Chen, Y. Zhang, and J. Li, "Cloud-based real-time video surveillance system using deep learning," *IEEE Trans. Multimedia, vol. 22, no. 5, pp. 1254–1266, 2020.*
19. Sultani, W., Chen, C., & Shah, M. (2018). Real-world anomaly detection in surveillance videos. In Proceedings of the *IEEE conference on computer vision and pattern recognition* (pp. 6479-6488).
20. Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., & Batra, D. (2017). Grad-cam: Visual explanations from deep networks via gradient-based localization. In Proceedings of the *IEEE international conference on computer vision* (pp. 618-626).