CNN-Based Threat Detection for Smart Surveillance in Nepal

Tika Datta Gautam
Navo Cloud Solutions Pvt Ltd
Balkot, Bhaktapur, Nepal

***Abstract*—The detection of threats using surveillance systems is an effective way to ensure the security of people, since it provides an opportunity to detect suspicious activity at an early stage. Despite the fact that the use of deep learning models gives high performance in detection, most of them are built based on complicated architectures, which are hard to implement on limited-resource devices. This paper studies the use of lightweight CNN architecture in detecting threats using surveillance in an experiment with controlled conditions. A unified three-class dataset containing images of *Armed*, *Fight*, and *Other* categories is collected from different public sources and used to compare the performance of eight CNN architectures by changing their depth, data augmentation, learning rate, and optimization algorithms. Experiment results showed that the combination of deeper architectures, data augmentation, smaller learning rates, and momentum allows for improving feature extraction, learning process stability, and generalization ability. The selected model has shown the following results: accuracy - 85.0%, precision - 85.1%, recall - 85.3%, F1-score - 84.9%.**

***Index Terms*—Threat Detection, Surveillance Systems, Convolutional Neural Networks (CNN), Deep Learning, Image Classification, Lightweight CNN, Human Activity Recognition, Weapon Detection.**

I. Introduction

Multiple criminal cases were reported during the fiscal year 2081/82 in Nepal. According to Nepal Police, a total of 6,110 cases were reported, out of which 1,602 cases were recorded in Kathmandu alone [1]. These cases included 1,042 murder cases, 92 attempted murder cases, and 947 social offense cases [1]. Another fact sheet report from Nepal Police reported 2,253 rape cases [2]. This indicates a serious public safety concern nationwide, affecting stakeholders including Nepal Police, the Government of Nepal, local governments, and citizens. The increasing number of cases places additional pressure on law enforcement agencies, challenges for public safety management, and raises concerns regarding the security of public spaces.

Manual CCTV surveillance systems are being deployed to monitor markets, heritage sites, city centers and other public spaces. According to Nepal Police records more than 3,000 CCTV surveillance system are present nationwide out of which 1,249 were present in the Kathmandu Valley alone [3]. These systems are monitored by the operators from Ranipokhari office and various other police stations [2]. Manual monitoring by operators have several limitations:

- Multiple surveillance frames to be monitored by a single operator.
- Operator fatigue with constant monitoring.
- Prone to miss critical events due to human limitation.
- Expansion of surveillance system increases the demand for operator increasing overall costs.

Deep learning techniques have demonstrated significant performance on classification of images and videos due to their ability to automatically extract and discriminate features from visual data. This signifies the possible use of Deep Learning on classifying the unusual threat activity shown by the surveillance camera. Existing approaches for classification for visual data in Deep Learning like Vision Transformer and Video Transformer are very powerful but requires high computational power and memory which can increase deployment cost for Government of Nepal. Multilayer Perceptron (MLP) can be used as Deep Learning method for low computation, but it does not preserve spatial relationships between pixels unless features are carefully engineered leading to accuracy drop and may not be optimal for CCTV frames with varying lighting, viewpoint changing, resolution differences and background noise. Convolutional Neural Network (CNN) provides a strong alternative by extracting spatial features while maintaining a balance between performance and computation complexities.

The proposed system is intended to help stakeholders including *Nepal Police*, *Metropolitan Police* and *Public Safety Agencies* by reducing operator workload through automated threat detection, enabling faster response time to critical events, and improving public safety. Therefore, this study presents a comparative analysis of multiple CNN architectures that use a curated "three-class" surveillance dataset to identify a lightweight model that achieves a balance between classification performance and computational efficiency.

The following are the key contributions of this study: (1) design and experimental analysis of a lightweight CNN model for threat detection in surveillance images; (2) controlled experiment involving the performance evaluation of eight different CNN models under various factors such as depth of the network, data augmentation, learning rate, and momentum, keeping all other parameters constant; (3) development of a consolidated three-class dataset of surveillance images from existing publicly available datasets; and (4) real-time prototype implementation of the proposed approach.

II. Literature Review

This section reviews the existing studies related to Deep learning for surveillance and threat detection. Existing re-

search has explored different approaches regarding detection for violent activities, weapons and suspicious behavior from surveillance footage. Different architectures has been proposed for with different classification performance and computational requirements.

A. Deep Learning for Violence Detection

Zahidul Islam and his research team has proposed the deep learning model for identifying violent activities from surveillance videos using the combination of CNN and Sep-ConvLSTM [4], [5]. The proposed architecture extracts spatio-temporal from video sequences improving performance and reducing the computational cost compared to ConvLSTM-based approaches [4], [6]. Experimental results on violence detection datasets demonstrated high performance, suggesting that the model is suitable for real-time surveillance application and low-end devices [4]. The authors also suggested pre-training on large-scale action recognition datasets might help achieve better generalization [4].

Similarly Sanskar Singh and his research team has proposed end-to-end deep learning-based video vision transformer (ViViT) [7] for detecting violent incidents in surveillance systems [8]. The model converts video into frames, add augmentations and learns spatio-temporal features from video clips using a transformer encoder to classify videos as violent or non-violent. Experimental results demonstrated that the proposed model out performed previous state-of-art approaches on challenging benchmark datasets [8]. However, the authors have concluded that the model requires a large quantity of training data for better training [8]. They have also suggested using generative adversarial networks (GANs) [9] and different transformer variants to further enhance its performance [8].

B. Deep Learning for Weapon Detection

Erssa Arif and her research team has proposed an automatic weapon detection system from surveillance footage using EfficientNet [10], [11]. Real time datasets, from local surveillance department's test sessions are used for model training and testing. Datasets consist of local environment images and videos from different type and resolution cameras that minimize the idealism [11]. Experimental results demonstrated that using EfficientNet algorithms, the accuracy achieved 98.12% when epochs increase as compared to other algorithms [11]. For future work author suggests to cover move objects more efficiently [11].

Similarly, P. Shanthi and V. Manjula proposed a model using ConViDeTR, a hybrid deep learning framework that integrates CNN, Vision Transformer (ViT), and Detection Transformer (DETR) architectures into a unified framework for weapon detection and face recognition [12]. The proposed framework introduced deep feature fusion layer, which integrates local spatial features, global context features, and object query features in one shared feature space [12]. Experiment results demonstrated that using existing benchmark datasets validate the performance, achieving 98.9% accuracy in weapon detection and 97.34% accuracy in face recognition and outperforming the existing techniques in both tasks. [12].

C. CNN Architectures for Intelligent Surveillance

Mohanarangan Kanniappan and his research team has proposed a high performance architecture using CNN-based architectures for real-time video classification for diverse theft related scenarios [13]. The proposed model was trained on real-world data and tested with naturally captured footage with industry grade surveillance cameras. [13]. Experimental results demonstrated that the model performed 92.91% classification accuracy and performed well on previously unseen backgrounds and locations [13]. The authors had also compared model size, processing speed, and Floating Point Operations (FLOPs) to select suitable architectures for high-performance surveillance systems [13]. The authors suggested further optimization of computational efficiency to improve scalability [13].

Existing study for deep learning approaches demonstrate it's effectiveness on the surveillance-based threat detection, violence detection and weapon detection. However, most of the existing methods utilizes the complex architectures such as Vision Transformers, Video Vision Transformers [8], ConvLSTM, SEP-ConvLSTM [5], or hybrid approaches that require higher computational resources and a large scale dataset. Fewer studies have focused on light-weight CNN utilizing small unified dataset for surveillance system. Therefore, this study experimentally verifies an optimal architecture with limited dataset balancing computation cost and performance for surveillance system.

III. Methodology

This section presents the methodology of controlled environment used in this study. It describes the task definition, dataset description, architecture design, experimental setups and training configurations.

A. Task Definition

The study treats threat detection as supervised three-class image classification problem where the objective is to classify input images into one of three classes: *Armed*, *Fight* and *Other*. The *Armed* class contains images representing weapons such as firearms and knives. The *Fight* class contains images representing fight and violent activities between two or more people. Finally the *Other* class contains images representing normal situations such as sitting, standing, walking, and more.

The performance of the CNN architecture is evaluated using accuracy, precision, recall, and F1-score. Accuracy provides an overall evaluation of the performance of the classification problem, while precision, recall, and F1-score provide detailed insights into the model's performance. These metrics are selected because threat detection systems require reliable identification, where precision evaluates false detections, recall evaluates the ability to detect actual threats, and F1-score evaluates the balance between precision and recall scores.

B. Data Collection

Eight different datasets were utilized to formulate a single dataset. All the datasets were publicly available on Kaggle [14]–[21].

These datasets contained data of different formats. Some datasets were available in image format, while some were in video format. Each dataset had different directory names, folder structures, and distributed classes.

The Weapon Detection dataset [16] contained images of purse, credit cards, mobile phones and currencies which were manually removed from directory before further filtration methods were applied.

Manual examination of image-based datasets revealed an extreme number of identical or duplicate images which looked exactly the same. A single image had more than 50 copies, which could make the model overfit. Therefore, duplicate images were filtered.

C. Dataset Preparation and Class Formulation

The datasets required different filtering approaches due to their different folder structures and class distributions. The approach was simple: instead of deleting images from the device, images were filtered into a separate directory in an organized way.

The datasets were organized into three directories including *raw*, *classified_data*, and *final_classes*. The *raw* directory contained all datasets mentioned above in their respective directory structures. The *classified_data* directory contained multiple datasets combined into a unified structure. This dataset included the first phase of filtration but still contained multiple classes that could be excluded or merged.

The *final_classes* directory contained the final three classes, where some classes were merged or excluded from the *classified_data*.

The *classified_data* dataset contained multiple classes including *dance*, *fight*, *gun*, *hit*, *hug*, *kick*, *knife*, *punch*, *sit*, *sleep*, *stand*, and *wave*.

Only three classes were required for this study: *Armed*, *Fight*, and *Other*. Therefore, the following approaches were applied for reducing the dataset to three classes:

- *gun* and *knife* classes were merged and saved into the *Armed* class, containing 952 images.
- *stand*, *sit*, *sleep*, and *hug* classes were merged and saved into the *Other* class, containing 1025 images.
- *fight* class was saved into the *Fight* class, containing 1049 images.

The remaining classes were discarded due to multiple reasons. The *Armed* class contained fewer samples compared to *Fight* and *Other* classes. Increasing the number of samples in *Fight* and *Other* classes would lead to uncontrollable class imbalance. The *wave* class contained multiple images where guns and knives were shown in hand while waving; however, filtering this class was not possible because the data was mixed. The *punch* class contained images with people wearing boxing gloves, punching bags, and some actual punching activities, mostly with the same background of boxing matches. The *hit* class contained images with people raising bats or hitting objects, mostly without interaction between two people.

D. Train, Validation and Test Split

The *final_classes* dataset contained the required three-class structure with 952 images inside the *Armed* class, 1025 images inside the *Other* class, and 1049 images inside the *Fight* class. This dataset was split into training, validation, and testing sets with a 0.7:0.15:0.15 split ratio, respectively, and saved to *final_dataset*. The final dataset distribution across training, validation, and testing subsets is shown in Fig. 1. Sample images from the training set representing each class are shown in Fig. 2.

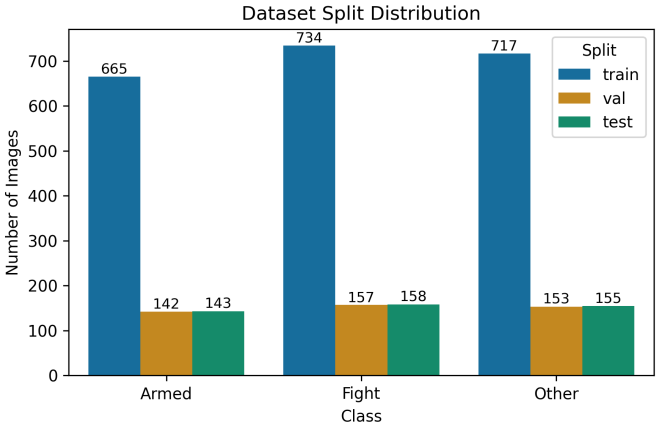

Fig. 1. Distribution of the final dataset across training, validation, and testing subsets.

The dataset was split while maintaining the original class distribution. The Fight class contained the highest number of samples with 734 training, 157 validation, and 158 testing images. The Other class contained 717 training, 153 validation, and 155 testing images. Finally, the Armed class contained 666 training, 142 validation, and 144 testing images. The distribution demonstrates slight variance in class samples, which was considered during model evaluation.

The sample images contain three rows and three columns. The rows represent the classes, and the columns represent the samples. The first row contains samples of the *Armed* class. The second row contains samples of the *Fight* class. Finally, the last row contains samples of the *Other* class.

The complete preprocessing pipeline and code implementation are available in the public GitHub repository for reproducibility for the dataset preparation process [22].

E. Architecture Design

For this study, controlled environment is created in which all experiments have to be done under similar settings. This helps in analyzing the modifications introduced to the CNN structure independently without any interference from outside factors. Therefore, all experiments are done with similar training

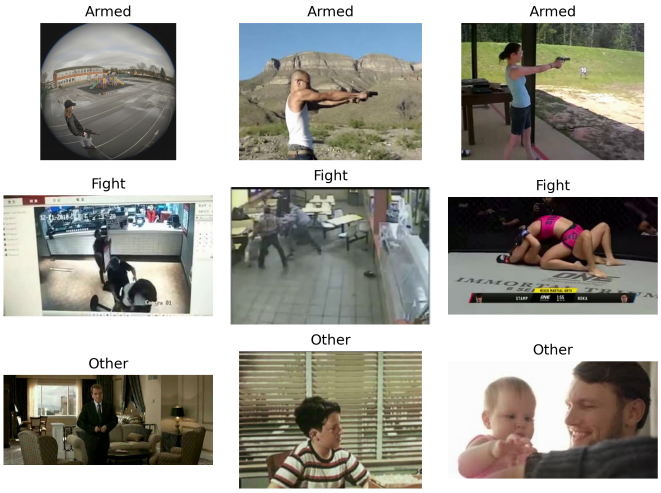

Fig. 2. Dataset samples across classes of training set.

setups and constrains. Following are the constrains for the controlled environment:

- Batch size was set to 32.
- Number of epochs was set to 100.
- Input image size was set to 64×64×3.
- A constant random seed value 40 was chosen to shuffle the data in *DataLoader* to make sure that all experiments have the same shuffled training data and provide a fair comparison of the architectures.
- Cross-Entropy Loss was used as the loss function throughout the study.
- Stochastic Gradient Descent (SGD) was used as an optimization algorithm.
- Every convolutional layer used a 3×3 kernel size, padding of 1 and stride of 1.
- Every pooling layer used MaxPooling with a 2×2 kernel size and stride of 2.
- Batch normalization was used after every convolutional layer.
- The first convolutional layer always produced eight output features.
- The number of output features is doubled after every following convolutional layer.

The goal of this research is to explore the lightweight CNN architecture, which will show a good balance between the performance of the threat detection and the computational efficiency. For this purpose, a custom architecture has been developed. It allows modifying the architecture of CNN while maintaining the controlled environment. This architecture has a modular design. Different experiments modify only components of architectures under investigation, the remaining controlled parameters are constant throughout the study.

F. Experimental Setup

Multiple experiments were conducted and evaluated for analyzing the performance of different CNN architectures under a controlled environment. Each experiment was developed by analyzing the performance and behavior of previous experiments. Each experiment introduced controlled changes to observe the effect on the performance of the model.

1) First Experiment - Baseline Architecture: The first experiment uses the baseline CNN architecture. It consists of 4 convolutional layers, each followed by a pooling layer. The experiment uses a learning rate of 2e-3. The total trainable parameters were calculated to be 131,587. The model has been visually represented in Fig. 3.

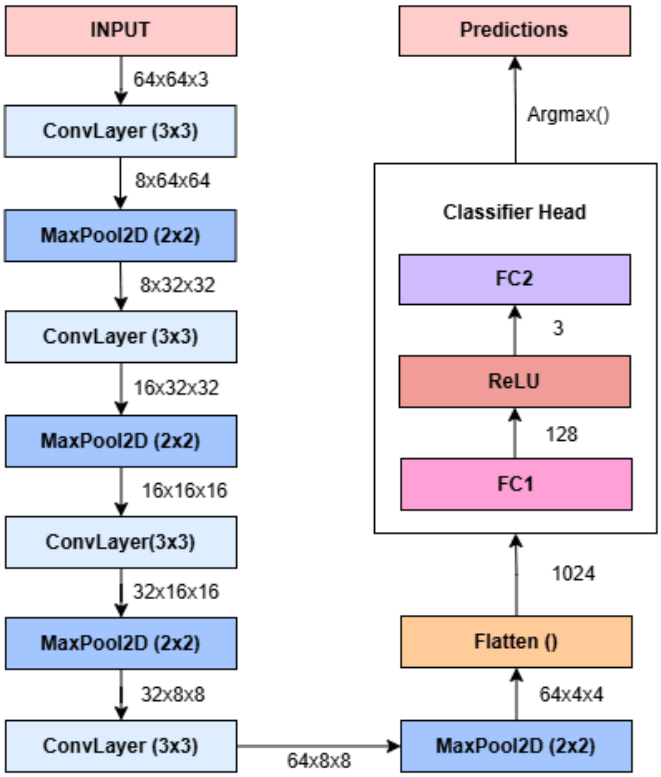

Fig. 3. Baseline CNN Architecture

In the given Fig. 3, a 'ConvLayer(3x3)' represents a sequence of convolution layer, batch normalization, and ReLU activation where (3x3) represents the kernel size of the convolutional layer. MaxPool2D (2x2) represents a pooling layer using maxpool with 2x2 kernel size. Finally, FC represents a fully connected layer.

2) Second Experiment - Baseline Architecture with Augmented Dataset: The second experiment uses the baseline architecture. The only difference is that this experiment uses augmented data. The augmentation includes random horizontal flips with p=0.5, random rotation of 10 degrees, color jitter with brightness of 0.2, contrast of 0.2, saturation of 0.2, and hue of 0.5, and random affine transformations with translate of (0.12, 0.12) and scale from 0.8 to 1.2. This allows the model to learn image classification on different angles, lighting, resolution, and scale that mimics real-life surveillance footage.

Random erasing is not used during data augmentation as it may hide the weapon in Armed class.

3) Third Experiment - Eight Convolutional Layers with Augmented Dataset: The third experiment utilizes a CNN architecture that consists of eight convolutional layers. Pooling layer is applied after every two consecutive convolutional layers. This experiment uses the same augmented data as the second experiment. Similarly, it uses a learning rate of 2e-3. The visualization of the architecture is present in Fig. 4.

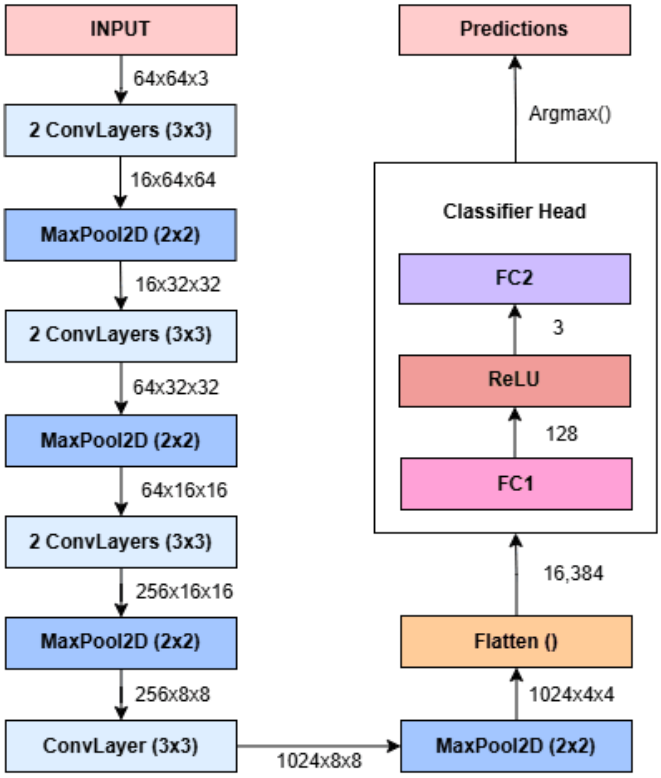

Fig. 4. 8 Layered CNN Architecture

In the given Fig. 4, a "2 ConvLayers (3x3)" represents two sequences, each containing a sequence of convolutional layer of 3x3 kernel size, batch normalization, and a ReLU activation.

4) Fourth Experiment - Eight Convolutional Layers without Augmented Dataset: The fourth experiment utilizes an eight-layered CNN architecture, same as used in the third experiment. This experiment uses the learning rate of 2e-3 which was used in previous experiments. This experiment was required to analyze how the model performs without the augmentation in training data and determine if the validation metrics are higher even without augmentation.

5) Fifth Experiment - Extension of Fourth Experiment with Lower Learning Rate: The fifth experiment is the extension of fourth experiment that uses a lower learning rate of 3e-4. This experiment was proposed to reduce extreme fluctuations observed in previous experiments.

6) Sixth Experiment - Extension of First Experiment with Lower Learning Rate: The sixth experiment is the extension of the first experiment that uses a lower learning rate of 3e-4. This experiment was proposed to analyze if it could reduce fluctuations seen in the first experiment as this approach showed significant improvements to fluctuations for CNN architecture with eight convolutional layers at the fifth experiment.

7) Seventh Experiment - Combination of Third and Fifth Experiments: The seventh experiment uses exactly the same CNN architecture as the third and fifth experiments. This experiment combines 8 convolutional layer architecture, augmentation on training set such that of third experiment, and lower learning rate of 3e-4 such that of fifth experiment. This combination was chosen because augmentation on third experiment worked well for overfitting and lower learning rate on fifth experiment showed significant reduction in fluctuations while the validation metrics were above 0.83 (83%).

8) Eighth Experiment - Extended Seventh Experiment with momentum: The Eighth experiment uses the CNN architecture same as that in seventh experiment. This experiment investigates if adding momentum of 0.9 to SGD improves stability and convergence compared to previous experiment as momentum remembers direction and magnitude from previous gradient updates, allowing faster convergence.

G. Training Configuration

The experiment uses Pytorch deep learning frame work. The models were trained with a fixed batch size of 32 for 100 epochs for same controlled environment as described above. Cross Entropy was used as the loss function and Stochastic Gradient Descent (SGD) was used as the optimizer. The learning rate was selected based on the experimental setup including either 2e-3 or 3e-4. In the eighth experiment, momentum of 0.9 was added to optimizer to analyze if it impacts on stability and convergence of training.

The training and validation losses were computed after each epoch to track the learning behavior of each architecture, along with accuracy, precision, recall and F1-score. The best model having best validation metrics throughout the study is selected and then tested on the unseen test sets. To ensure a fair comparison between different CNN architectures, all the experiments were trained with the same training, validation and testing datasets with same constant seed 40 for shuffle re-generation.

IV. Results and Discussion

This section presents the comparative analysis of results of experiments used in this study. It presents a discussion on the final best model that can be used for the actual detection in surveillance system.

A. Experimental Evaluation

Eight different experiments were conducted, where each experiment showed different training behavior based on loss, accuracy, precision, recall and f1-score curves. Detailed analysis of each experiment are presented as follows.

1) First Experiment - Baseline Architecture: This experiment used the Baseline Architecture visually represented in Fig. 3. The loss and metric curves of this experiment is present at Fig. 5. The loss curve shows a smooth decrease in training set tending to zero while it showed extreme fluctuations peaking up-to 6 in validation set. The accuracy curve shows, accuracy rising gradually towards one 1 (100%) for training set while the validation rises around 0.83 (83%) with extreme fluctuations. Similar patterns were observed for all the metrics including precision, recall, and f1-score. A clear gap is present between training and validation curves for every chart which indicates overfitting. Overall evaluation of the experiment indicates that model is learning effectively on training data but could not generalize consistently on unseen data.

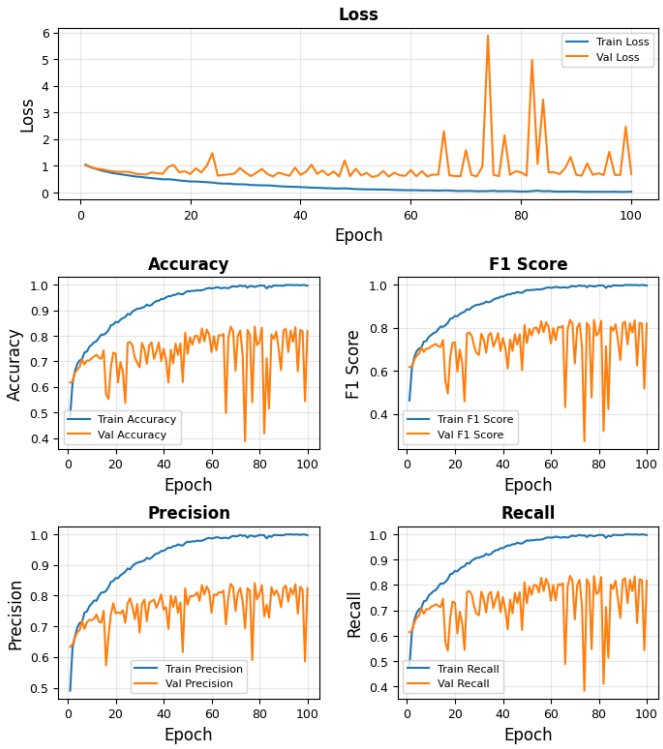

Fig. 5. Curves observed in the First Experiment

In the Fig. 5, blue line represents the line on training set and orange line represents the line on validation set.

2) Second Experiment - Baseline Architecture with Augmented Dataset: This experiment used the Baseline Architecture trained on Augmented training dataset. The loss and metric curves of this experiment is present at Fig. 6. The loss curve shows a smooth decrease in training set tending to zero while extreme fluctuations with peaks above 1.4 were recorded in the validation set. The accuracy curve shows, gradual rise on training set approaching 0.8 (80%) accuracy. It also shows that the accuracy on validation set tends to move together with accuracy on training set but with extreme fluctuations. Similar patterns can be observed in recall, and F1-score. The precision curve showed lower fluctuation rate as compared to other metrics. Training on augmented data performed reasonably well for overfitting but had severe fluctuations. Most of the metrics for training and validation sets remained around 0.8 (80%) representing slow convergence or mild underfitting.

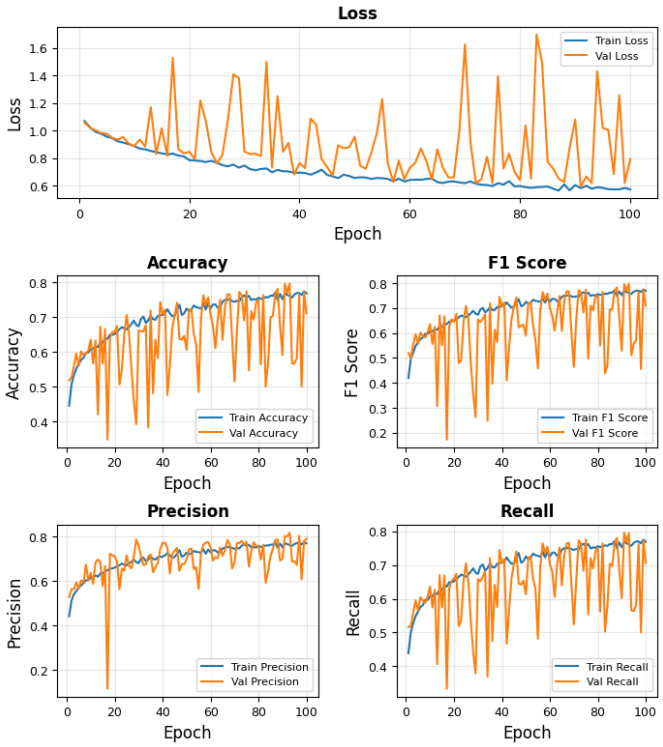

Fig. 6. Curves observed in the Second Experiment

3) Third Experiment - Eight Convolutional Layers with Augmented Dataset: This experiment used the CNN architecture with eight convolutional layers. It was trained on the augmented training dataset. The architecture can be visualized in Fig. 4. The loss and metric curves of this experiment are shown in Fig. 7. The loss curve shows a smooth decrease in the training set tending towards zero, while extreme fluctuations were visible in the validation set. The gap between the training and validation loss widened during the later epochs, suggesting some overfitting. The accuracy curve shows rise to approximately 0.65 (65%) by the 10th epoch and gradually approached 0.85 (85%) accuracy on the training set. It also shows that the validation accuracy tends to move together with the training accuracy but with extreme fluctuations. Similar patterns can be observed in recall, precision, and F1-score. The addition of convolutional layers pushed the metric scores with an approximately 5% increment suggesting better feature extraction.

4) Fourth Experiment - Eight Convolutional Layers without Augmented Dataset: This experiment used the CNN architecture with eight convolutional layers, the same as in the Third Experiment. It was trained on the training dataset without augmentation. The loss and metric curves of this experiment are shown in Fig. 8. The loss curve shows a smooth decrease in the

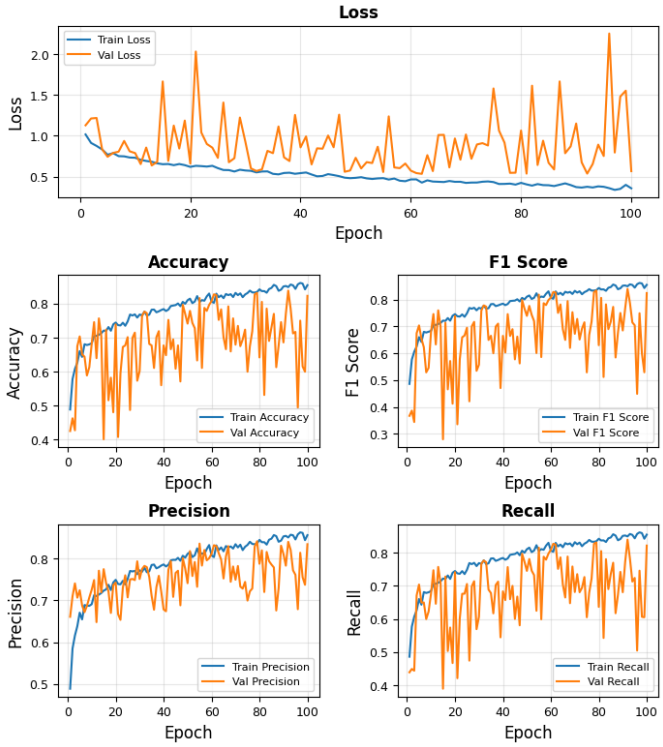

Fig. 7. Curves observed in the Third Experiment

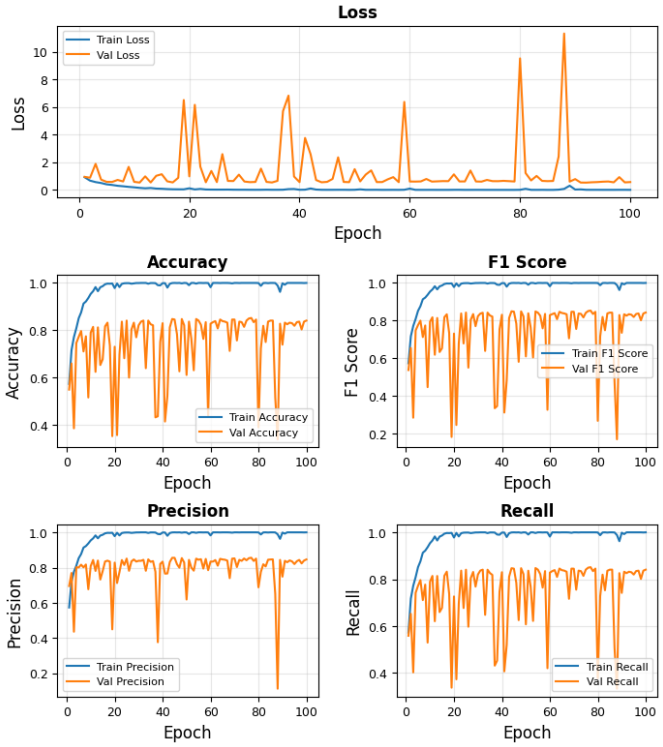

Fig. 8. Curves observed in the Fourth Experiment

training set, tending towards zero, while extreme fluctuations were visible in the validation set. A visible gap was present between the training and validation loss, suggesting overfitting. The accuracy curve shows a quick rise, reaching 1.00 (100%) by the 31st epoch, and then stabilized with only minor fluctuations. In contrast, the validation accuracy approached 80–85% by the 31st epoch and stabilized with extreme fluctuations. A wide gap of approximately 15–20% is present between the training and validation accuracy, suggesting severe overfitting. Similar patterns can be observed for recall, precision, and F1-score. The precision score showed comparatively lower fluctuations. The removal of augmentation resulted in severe overfitting, providing an insight that data augmentation helped the model prevent overfitting.

5) Fifth Experiment - Extension of Fourth Experiment with Lower Learning Rate: This experiment used the CNN architecture with eight convolutional layers, the same as in the Fourth Experiment, with a lower learning rate of 3e-4. The loss and metric curves of this experiment are shown in Fig. 9. The loss curve shows a smooth decrease in the training set, tending towards zero, while the validation loss decreased and stabilized at approximately 0.6 with minor fluctuations. A visible gap of approximately 0.6 was present between the training and validation loss, suggesting overfitting. The training accuracy curve shows a gradual and stable rise towards 1 (100%). In contrast, the validation accuracy approached 80–85% by the 40th epoch and stabilized with minor fluctuations. A gap of approximately 15–20% is present between the training and validation accuracy, suggesting overfitting. Similar patterns can be observed for recall, precision, and F1-score. The change in the learning rate from 2e-3 to 3e-4 drastically reduced the fluctuations and resulted in a more stable learning process.

6) Sixth Experiment - Extension of First Experiment with Lower Learning Rate: This experiment used the Base Architecture, the same as in the First Experiment, with a lower learning rate of 3e-4. The loss and metric curves of this experiment are shown in Fig. 10. The loss curve shows a smooth decrease in the training set, tending towards 0.4, while the validation loss decreased smoothly towards 0.6. A visible gap slowly widened with every epoch, suggesting overfitting. The training accuracy curve shows fast learning, reaching 0.6 (60%) by the 20th epoch and then gradually rising towards 0.9 (90%). Similarly, the validation accuracy reached 0.6 (60%) by the 20th epoch and then gradually rose towards 0.8 with very small fluctuations. A visible gap slowly widened after reaching 60% accuracy with every epoch, suggesting overfitting. Similar patterns can be observed for recall, precision, and F1-score. This experiment indicated that lowering the learning rate to 3e-4 reduced fluctuations, which was consistent with the observation from the fifth experiment.

7) Seventh Experiment - Combination of Third and Fifth Experiment: This experiment combines the third and fifth experiment that utilizes the the Eight layered CNN-architecture. The experiment uses learning rate of 3e-4 and trains on

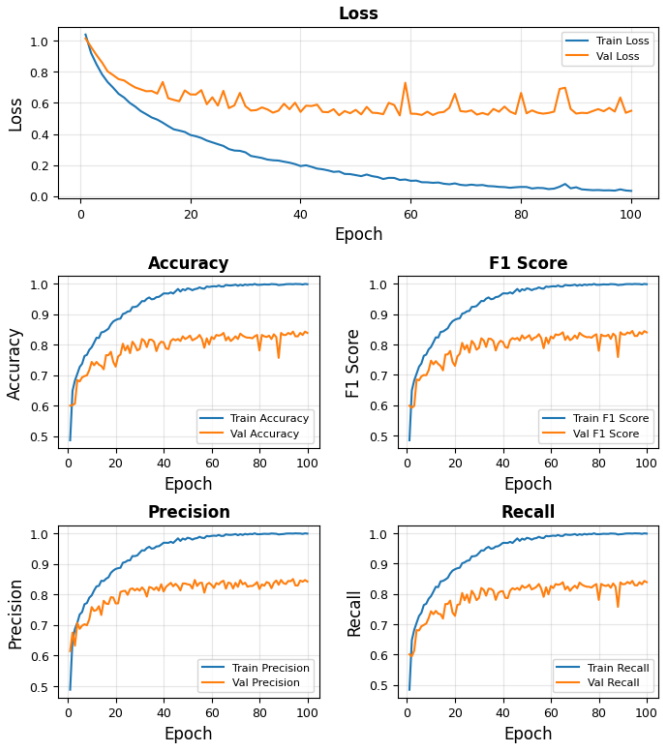

Fig. 9. Curves observed in the Fifth Experiment

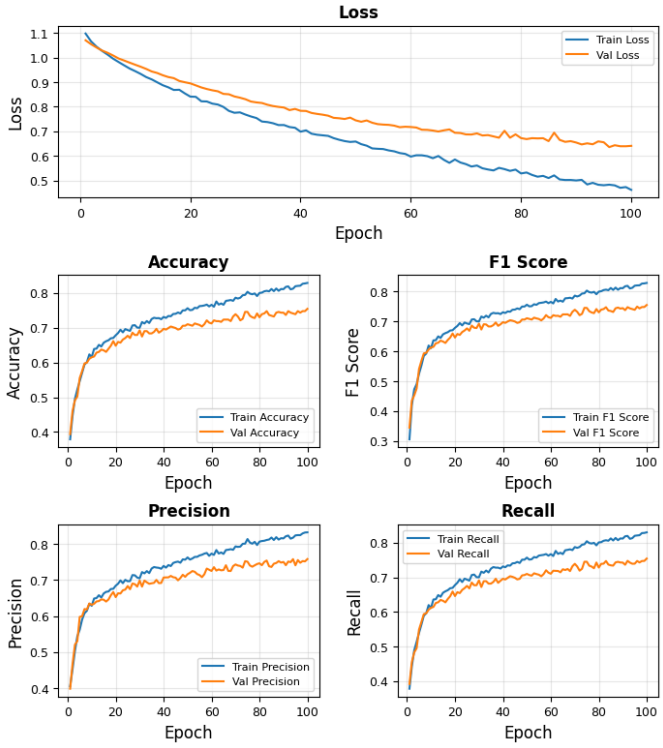

Fig. 10. Curves observed in the Sixth Experiment

the augmented dataset. The loss and metric curves of this experiment are shown in Fig. 11. The loss curve shows a smooth decrease in the training set, tending towards 0.5, where the validation loss follows with minor fluctuations. The accuracy on training set trends to rise towards 0.80 (80%) and the accuracy on validation set follows it with minor fluctuations. Similar pattern is observed for precision, recall and F1-score. Both training, and validation metrics tend to rise towards 80% score showing slow convergence. Overall the model had minimized fluctuation on validation sets and no significant overfitting was observed.

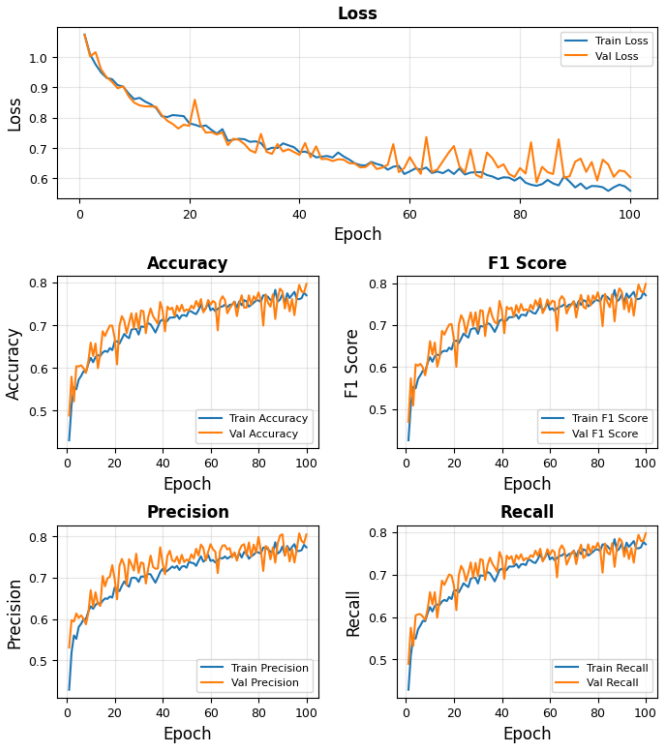

Fig. 11. Curves observed in the Seventh Experiment

8) Eighth Experiment - Extended Seventh Experiment with momentum: This experiment adds momentum to the optimizer of the Seventh Experiment. The loss and metric curves of this experiment are shown in Fig. 12. The loss curve decreases in the training set, tending towards 0, while the validation loss follows with minor fluctuations until around the 20th epoch. The validation loss tends to stabilize between 0.5 and 0.6 after the 20th epoch, which gradually widens the gap between the training and validation loss, indicating some overfitting during the later training epochs. The training accuracy rises to 0.70 (70%) in the initial epochs and then gradually rises towards 0.90 (90%), while the validation accuracy moves together with it. In the later epochs, the validation accuracy stabilizes around 0.85 (85%), while the training accuracy continues to rise, indicating minor overfitting. The same pattern was observed for the remaining metrics, including precision, recall, and F1-score. Overall, the model learned the training dataset effec-

tively and, although it showed minor overfitting in the later epochs, it generalized well based on the validation accuracy and F1-score.

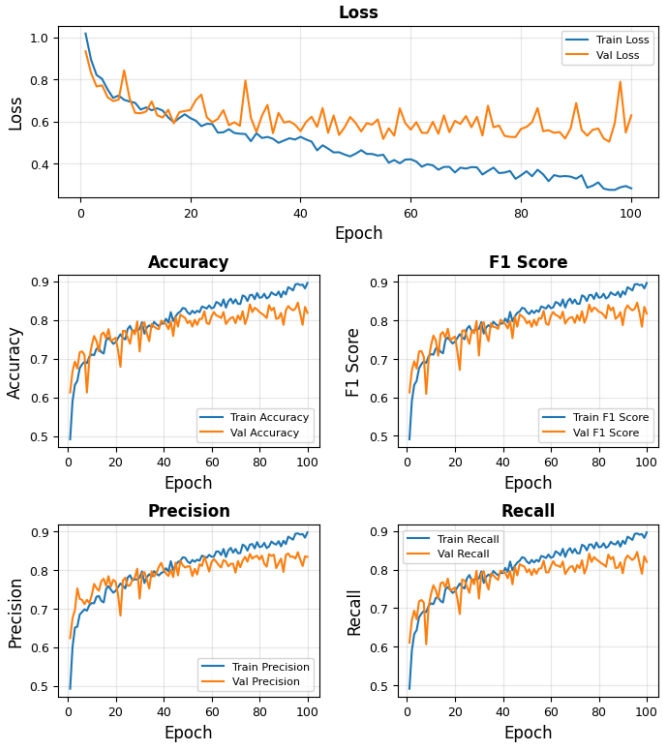

Fig. 12. Curves observed in the Eighth Experiment

B. Experimental Comparison

The eight experiments show how CNN efficiency depends on the interaction of its capacity, regularization, and optimization technique and not on the complexity of its architecture. Indeed, increasing the number of convolutional layers positively affected feature extraction since the network had an opportunity to extract higher order spatial structures from surveillance images. At the same time, the experiments proved that increase in model capacity along with insufficient model regularization leads to higher risk of memorizing training examples, evidenced by the growing difference between the training and validation metrics.

Data augmentation played an important role in improving the generalization capability of the CNN since it decreased the dependence on particular image features which appeared in the training dataset. Comparison of the third and fourth experiments proves that higher accuracy of the deeper models was achieved not only because of higher numbers of layers, but also because of preserving enough variability of data during the learning process.

In addition to this, lowering the learning rate helped increase optimization stability by ensuring smaller parameter changes during the optimization process, which ensured smooth convergence and less variance in validation. The use of momentum also helped increase the efficiency of the optimization process by ensuring consistency in the direction of updates, which helped in better convergence without any instability in the changes in parameters. The best results were obtained in the last experiment in terms of ability of feature extraction, generalization, and optimization stability, and hence, it was selected as the final model.

C. Final Model Evaluation

The eighth experiment was finally chosen as the selected model because of the combination of the three criteria mentioned: classification performance, generalization ability, and training stability of the model. While there were other experiments that had similar performance during the training process, the eighth experiment performed better by showing less variation during the validation process, as well as low overfitting without sacrificing the accuracy and F1 score.

The checkpoint at 94th epoch at which the validation accuracy score of 84.2% and F1 score of 84.3% were maximized was chosen as the final model. It was further tested in test dataset to avoid choosing a model based on the results from the training data alone since it might have poor generalization on unseen samples.

Further testing of the model at the 94th epoch of the eighth experiment showed good generalization with 85% accuracy, 85.1% precision, 85.3% recall, and 84.9% F1-score on the unseen test dataset. The confusion matrix of the test is shown in the Fig. 13.

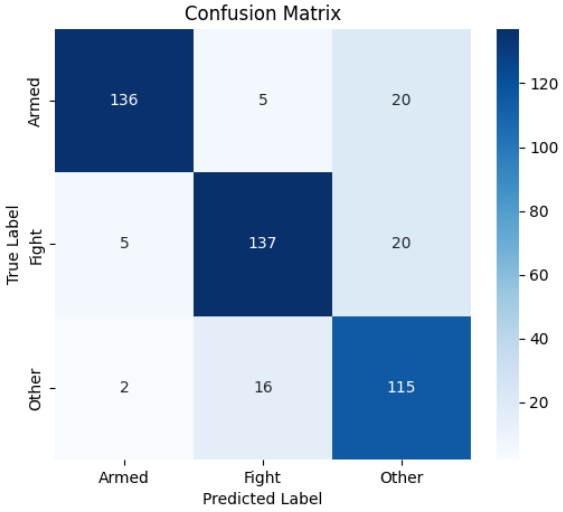

Fig. 13. Confusion Matrix observed in final model

The confusion matrix reveals that the model has strong overall performance, with about 85% accuracy in all three classes. In particular, the model was able to classify 136 examples of Armed, 137 of Fight, and 115 of Other, and shows good agreement with the main diagonal. The rate of classification error for the Armed and Fight classes is relatively

low, with five errors made in each of these classes. However, the most significant errors were observed in those cases where the Armed and Fight classes were confused with Other classes, since there were twenty errors in each of these cases. It can be assumed that some situations representing a threat but having poor distinguishing characteristics may be difficult for classification by the model.

D. Prototype Implementation

The chosen CNN was implemented into the prototype to show how possible threat recognition could work. The prototype was created using the frontend interface developed using Vite.js for accessing a camera and backend written using FastAPI for model inference. The model was loaded into the backend, capturing frames from a camera in frontend, preprocessing them, and predicting their class. After that, the predicted class was returned and visualized on the frontend interface.

The evaluation of the prototype revealed that it can perform inference based on a live camera feed. Images of weapon containing situations visualized via the laptop camera were successfully recognized by the model, proving the success of integration of the camera interface, backend, and CNN.

Unfortunately, the evaluation of live threat recognition wasn't done completely because of the impossibility to implement fight or weapon scenarios in the test environment. Also, the environment of the laptop camera is different from the one of the surveillance images used while training: surveillance images contained such characteristics as CCTV viewpoint, distant subjects, and low image quality. Therefore, additional evaluation with a real surveillance camera is needed.

V. Key Findings

In this work, the impact of various architectural modifications made to convolutional neural networks on the efficiency of the threat detection task was analyzed within the framework of controlled training conditions. The key observations obtained as a result of experiments are summarized as follows:

- Increase in the number of convolutional layers led to improved feature extraction and better classification results. However, deeper architectures also increased the likelihood of overfitting without proper regularization.
- Augmentation of the data improved the generalization capability of the CNN model due to overfitting avoidance caused by lack of variation in the training data set. Comparison of experiments with and without augmentation showed the importance of maintaining image variability for surveillance-based classification problems.
- The learning rate had a great impact on training stability. Decrease of the learning rate from 2e-3 to 3e-4 reduced validation variability and allowed for smoother convergence.
- The model chosen as a result of the eighth experiment had 85% accuracy and 84.9% F1-score on the unseen test data set. In addition, the model managed to classify three surveillance-related classes: *Armed*, *Fight*, and *Other*.

VI. Limitations

This study has several limitations that should be acknowledged.

First of all, the collection of the dataset used for this research is relatively small in terms of the number of surveillance images collected from public sources. Although some preprocessing and augmentation techniques were used to enhance the dataset's ability to generalize, there could be limitations in terms of variation in real life surveillance situations, such as the angle of the camera, lighting, background, and image quality.

The second point is that the proposed technique depends entirely on analyzing individual images in order to classify them. Given that the events of interest happen continuously over time, some situations might require the inclusion of some sort of motion or temporal information in their classification.

Thirdly, the dataset consists only of three broad categories: Armed, Fight, and Other. While this simple classification makes it easy to create a quick detection system, it also limits how functional the model can be in terms of getting details about threats.

Lastly, the prototype was tested in a laptop camera environment where the printed images of threats were shown which were correctly classified. However, the validation is incomplete because the model was not tested on the real-world live surveillance camera.

VII. Future Work

There are many opportunities to improve the effectiveness of the proposed threat detection system in the future. Collecting the larger and more diverse dataset based on real-life surveillance scenarios will allow it to achieve higher generalization capacity. The dataset should contain a number of different variations in terms of different camera angles, lighting conditions, distance to the object, and environmental factors.

Additionally, future studies may improve the current image-based classification system by using temporal information in such architectures as CNN-LSTM [23], 3D CNNs [24], or video vision transformers [7]. Such that, the system will be able to derive motion patterns.

Furthermore, classification categories could provide a hierarchical based threat detection model. Instead of only identifying simple categories like Armed, Fighting and Other, in the future, it will be possible to distinguish specific kinds of weapon such as firearm, knife, explosives, swords, etc and human activity such as walk, jump, fall, and more. This can provide more meaningful information for real-world security applications.

Finally, the future work can focus on the deployment and evaluation of model on real surveillance system with continuous video streams, multiple subjects, varying distance and challenging environmental conditions. Optimization techniques such as model compression, quantization [25], and edge-device deployment can also be investigated to improve real-time performance while maintaining detection accuracy.

VIII. Conclusion

This study presented a light weight CNN-based threat detection system for surveillance images. It investigated the effect of architectural and optimization modifications under a controlled environment. Total eight experiments were conducted by validating model depth, data augmentation, learning rate, and optimizer settings while consistence training environment were maintained.

The experimental analysis has shown that increasing convolutional depth improved feature extraction, while applying the augmentation reduced overfitting and improved generalization. Furthermore, reducing the learning rate and applying momentum contributed on reducing fluctuation leading to more stable training. On the basis of balance between classification performance, stability and generalization on unseen data, the model at 94th epoch of eighth experiment was selected as the final model.

The selected model had achieved 85% accuracy, 85.1% precision, 85.3% recall, and 84.9% F1-score on the unseen test dataset which demonstrates a feasibility of light-weight CNN architectures for surveillance-based threat detection. However, further improvements are required to support complex, real-world scenarios, including video-based analysis, hierarchical threat categories, and deployment under diverse surveillance conditions.

IX. Code Availability

The source code, including data processing, model training, backend inference engine, and prototype development is open-source to facilitate reproducibility. The source code for both training and the backend is available at github repository [26] whereas the frontend code, which includes camera input and prediction visualization is available at [27]

References

[1] The Himalayan Times, "Over 6,100 crime cases registered nationwide, kathmandu valley tops the list," *The Himalayan Times*, 2025.

[2] Nepal Police, "Annual crime infographics fiscal year 2081/82," Nepal Police, Tech. Rep., 2025.

[3] S. Dhungana, "Authorities are installing more cctv cameras to increase surveillance in the capital city," *The Kathmandu Post*, 2019.

[4] Z. Islam, M. Rukonuzzaman, R. Ahmed, M. H. Kabir, and M. Farazi, "Efficient two-stream network for violence detection using separable convolutional lstm," Islamic University of Technology, Tech. Rep., 2021.

[5] A. Pfeuffer and K. Dietmayer, "Separable convolutional lstms for faster video segmentation," arXiv, Tech. Rep., 2019.

[6] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W. kin Wong, and W. chun Woo, "Convolutional lstm network: A machine learning approach for precipitation nowcasting," in *Advances in Neural Information Processing Systems*, 2015.

[7] A. Arnab, M. Dehghani, G. Heigold, C. Sun, M. Lucic, and C. Schmid, "Vivit: A video vision transformer," arXiv, Tech. Rep., 2021.

[8] S. Singh, S. Dewangan, G. S. Krishna, V. Tyagi, S. Reddy, and P. R. Medi, "Video vision transformers for violence detection," IIIT Naya Raipur, Tech. Rep., 2021.

[9] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems*, 2014.

[10] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *Proceedings of the 36th International Conference on Machine Learning*, 2019.

[11] E. Arif, S. K. Shahzad, M. W. Iqbal, M. A. Jaffar, A. S. Alshahrani, and A. Alghamdi, "Automatic detection of weapons in surveillance cameras using efficient-net," *Computers, Materials & Continua*, 2022.

[12] P. Shanthi and V. Manjula, "An efficient hybrid cnn–transformer framework for real-time weapon detection and face recognition," *Frontiers in Artificial Intelligence*, vol. 9, 2026.

[13] M. Kanniappan, P. P, and A. Gadde, "Enhanced surveillance system through cnn video classifiers: An evaluation and architecture proposal," SSRN, Tech. Rep., 2024.

[14] "Cctv weapon dataset." [Online]. Available: https://www.kaggle.com/datasets/simuletic/cctv-weapon-dataset

[15] A. Sharma, "Weapon detection dataset." [Online]. Available: https://www.kaggle.com/datasets/ankan1998/weapon-detection-dataset

[16] M. Cubukcu, "Weapon detection." [Online]. Available: https://www.kaggle.com/datasets/mehmetcubukcu/weapon-detection

[17] K. Ashraf, "Human action recognition." [Online]. Available: https://www.kaggle.com/code/kirollosashraf/human-action-recognition-har

[18] "Cctv knife detection dataset." [Online]. Available: https://www.kaggle.com/datasets/simuletic/cctv-knife-detection-dataset

[19] "Weapon - gun detection & segmentation." [Online]. Available: https://www.kaggle.com/datasets/trainingdatapro/people-with-guns-segmentation-and-detection

[20] A. Saravanan, "Fight dataset." [Online]. Available: https://www.kaggle.com/datasets/anbumalar1991/fight-dataset

[21] S. Shekhar, "Knife dataset." [Online]. Available: https://www.kaggle.com/datasets/shank885/knife-dataset

[22] T. D. Gautam, "Crime and threat dataset cleaning." [Online]. Available: https://github.com/tikadatta2005/Crime-And-Threat-Dataset-Cleaning

[23] T. N. Sainath, O. Vinyals, A. Senior, and H. Sak, "Convolutional, long short-term memory, fully connected deep neural networks," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2015, pp. 4580–4584.

[24] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3d convolutional networks," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2015, pp. 4489–4497.

[25] B. Rokh, A. Azarpeyvand, and A. Khanteymoori, "A comprehensive survey on model quantization for deep neural networks in image classification," *IEEE Access*, 2023.

[26] T. D. Gautam, "Efficient surveillance system research." [Online]. Available: https://github.com/tikadatta2005/efficient-surveillance-system-research

[27] ——, "Weapon-violence detection frontend." [Online]. Available: https://github.com/tikadatta2005/weapon-violence-detection-frontend