Developing an AI Model for Detecting Violent Behaviors

Seong-Ho Kim, Electrical Engineering, 2019060564 Hanyang University
Jong-Hyuk Hong, Electrical Engineering, 2019047665, Hanyang University

I. Introduction

With the increasing number of surveillance cameras in modern cities, it has become very difficult for humans to monitor all screens at once. Therefore, by developing an AI model that detect violent behavior by utilizing video understanding technology, it is expected to be able to provide fast alerts in time. This project aims to develop a deep learning model capable of classifying whether a given video scene depicts a violence or non-violence, thus constituting a binary classification task. To achieve this goal, the project will utilize the Lucas-Kanade algorithm for optical flow processing and the farneback algorithm for training the detection model. Moreover, all the tasks should be performed using Google Colab.

II. Optical Flow

A. Lucas–Kanade Algorithm

The Lucas-Kanade method is a widely used algorithm in computer vision for estimating the optical flow of a video sequence. It was introduced by Bruce D. Lucas and Takeo Kanade in their seminal paper "An Iterative Image Registration Technique with an Application to Stereo Vision" in 1981[1].

The Lucas-Kanade method assumes that the flow is essentially constant in a local neighborhood of the pixel under consideration. It estimates the motion of a small window of pixels by solving a set of linear equations. This method is particularly effective for estimating the optical flow in small neighborhood regions, making it suitable for tracking objects in video sequences.

Figure 1. Consecutive Images

In Figure 1, we can see the window of pixels are moved in the flow of time. The difference between those two images can be depicted as the following equation.

$$I(x, y, t) = I(x + \delta x, y + \delta y, t + \delta t) \quad (1)$$

We can apply Taylor's approximation, the equation (1) can be re-written as the following equation.

$$I(x, y, t) \approx I(x, y, t) + \frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta y + \frac{\partial I}{\partial t}\Delta t \quad (2)$$

$$\frac{\partial \mathrm{I}}{\partial \mathrm{x}}\Delta \mathrm{x} + \frac{\partial \mathrm{I}}{\partial \mathrm{y}}\Delta \mathrm{y} + \frac{\partial \mathrm{I}}{\partial \mathrm{t}}\Delta \mathrm{t} = 0 \quad (3)$$

$$\therefore I_x v_x + I_y v_y = -I_t \quad (4)$$

Here, I_x and I_y represent spatial components of the image, and I_t represents the temporal component. Therefore, there are two unknowns for each pixel. Consequently, solving this equation becomes challenging.

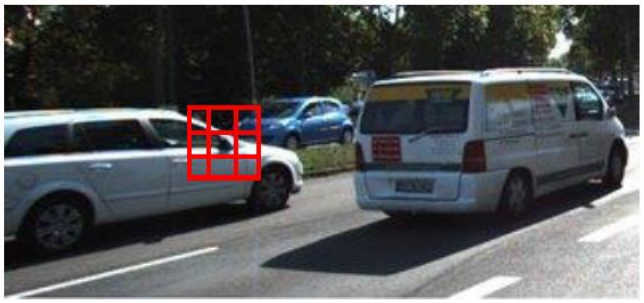

Figure 2. Patch of Interest

We assume that $v_x \approx v_y$ for nearby pixels. Let's consider a 3×3 patch with its center at (x, y).

$$\begin{bmatrix} I_x(x-1,y-1) & I_y(x-1,y-1) \\ \vdots & \vdots \\ I_x(x+1,y+1) & I_y(x+1,y+1) \end{bmatrix} \begin{bmatrix} v_x \\ v_y \end{bmatrix} = \begin{bmatrix} -I_t(x-1,y-1) \\ \vdots \\ -I_t(x+1,y+1) \end{bmatrix} \quad (5)$$

We can now solve the problem by linear least square method.

After computing the image gradients, spatial and temporal, we can take every $(2N+1) \times (2N+1)$ patch centered around (x, y). The solution vector $\mathbf{v}$ can be obtained by solving the linear equation $\mathbf{A^T A v = A^T b}$, $\mathbf{v = (A^T A)^{-1} A^T b}$.

$$\mathbf{v} = \begin{bmatrix} \sum_{i=-N}^{N}\sum_{j=-N}^{N} I_x^2(x-i,y-j) & \sum_{i=-N}^{N}\sum_{j=-N}^{N} I_x(x-i,y-j)I_y(x-i,y-j) \\ \sum_{i=-N}^{N}\sum_{j=-N}^{N} I_x(x-i,y-j)I_y(x-i,y-j) & \sum_{i=-N}^{N}\sum_{j=-N}^{N} I_y^2(x-i,y-j) \end{bmatrix}^{-1} \begin{bmatrix} \sum_{i=-N}^{N}\sum_{j=-N}^{N} I_x(x-i,y-j)I_t(x-i,y-j) \\ \sum_{i=-N}^{N}\sum_{j=-N}^{N} I_x(x-i,y-j)I_t(x-i,y-j) \end{bmatrix} \quad (6)$$

Note that $\mathbf{A^T A}$ is similar to the matrix in Harris corner detector[2]. Therefore, by letting $\mathbf{H} = \mathbf{A^T A}$, we can apply thresholding to filter out corners same as in the Harris corner detector.

B. Farneback Algorithm

Farneback algorithm[3] can be applied to other types of image sequences with varying and difficult to predict displacements between successive frames. The algorithm's ability to handle large and varying displacements makes it suitable for a wide range of applications, including those with challenging motion characteristics.

The algorithm has three approaches to solve the problem. The first is to utilize the polynomial expansion transform to approximate each neighborhood of the frames by quadratic polynomials. It can be depicted as the following equation.

$$f(x) \approx \mathbf{x}^T \boldsymbol{A}\mathbf{x} + \mathbf{b}^T\mathbf{x} + c \qquad (7)$$

This step allows the algorithm to capture local motion characteristics and variations, which is essential for handling the effects of high-frequency vibrations on the captured images. Moreover, the algorithm employs a multi-scale approach, starting at a coarse scale to obtain a rough but reasonable displacement estimate and then propagating this through finer scales to obtain increasingly more accurate estimates. The algorithm also refines the displacement estimates. After obtaining the initial displacement estimates, the algorithm applies a series of refinements to the estimated displacement fields to improve the accuracy and robustness of the motion estimation.

However, it is important to note that the algorithm's performance may vary depending on the specific characteristics of the image sequences being analyzed. To address these challenges, the algorithm can be combined with other techniques, such as simultaneous segmentation procedures, to improve its performance.

C. Similarities and Differences

The Lucas-Kanade algorithm and the Farneback algorithm are similar and different. In this section, we will get to know the similarities first.

First, both algorithms track pixel in images to calculate optical flow. It contains the procedure of computing gradients. Moreover, both algorithms may face difficulties in scenarios involving rapid movements or rotations. It can cause some error in computing spatial gradients.

The difference between the two algorithms occur in scale and area. Lucas-Kanade works well in small regions, whereas Farneback can estimate large movements spanning the entire image. Moreover, the differ in the area of accuracy and complexity. Lucas-Kanade provides quick results through relatively simple calculations, but its accuracy may suffer in the presence of large motions or complex textures. On the other hand, Farneback requires more complex computations but offers a more accurate optical flow, especially in challenging scenarios. Lastly, they differ in the way of approximation. Farneback algorithms uses a second-degree polynomial approximation for calculating optical flow, while Lucas-Kanade relies on a first-degree approximation.

III. RWF 2000-Detection Model

A. Data Class

Create a custom dataset class utilizing the 'Dataset' class from 'torch.utils.data' for handling data in PyTorch. This class should encompass fundamental functions such as 'init', 'len', and 'getitem'. The 'init' function initializes the dataset, 'len' returns the total number of samples in the dataset, and 'getitem' retrieves a specific sample.

In addition to these base functions, include data loading functions named 'data_generation' and 'search_data'. The 'data_generation' function is responsible for loading processed data, which includes optical flow information. It plays a key role in generating the input data for training or evaluation. On the other hand, the 'search_data' function assists in efficiently locating and retrieving specific data within the dataset.

Overall, we used 160 train data to perform training and 40 datas for validation. Moreover, we used 20 datas to test the model's performance.

B. Neural Network Architecture

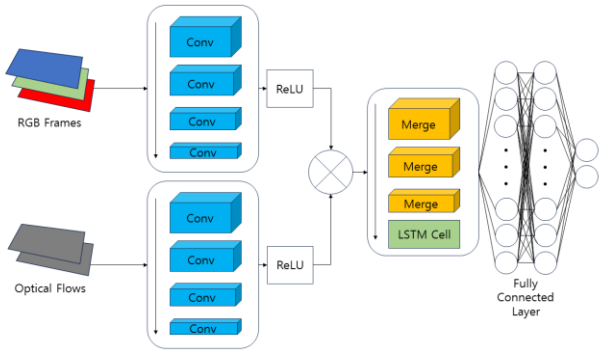

Figure 3. Model Architecture

To achieve our goal, we designed a neural network architecture. The original structure is suggested in Table 1. To improve the performance of classification, we added an LSTM cell to the original model, as in Figure 3, at the end of the Merging Block. The rationale behind utilizing

LSTM cells lies in their proficiency in handling time-series data. The choice was driven by the belief that LSTMs excel in processing temporal sequences, making them well-suited for effectively managing data variations across different frames. It is expected that using an LSTM cell to use a memory cell can provide some performance improvement as it helps address the vanishing gradient problem.

Block Name	Type	Filter Shape	T
RGB/Flow Channels	Conv3d	1×3×3@16	
	Conv3d	3×1×1@16	2
	MaxPool3d	1×2×2	
	Conv3d	1×3×3@32	
	Conv3d	3×1×1@32	2
	MaxPool3d	1×2×2	
Fusion and Pooling	Multiply	None	1
	MaxPool3d	8×1×1	1
Merging Block	Conv3d	1×3×3@64	
	Conv3d	3×1×1@64	2
	MaxPool3d	2×2×2	
	Conv3d	1×3×3@128	
	Conv3d	3×1×1@128	1
	MaxPool3d	2×2×2	
Fully-connected Layers	FC layer	128	2
	Softmax	2	1

Table 1. Original RWF-2000 Architecture[4]

C. Training Algorithm

The training algorithm initializes an SGD optimizer with a learning rate of 0.003, weight decay of 1e-6 for regularization, and Nesterov momentum of 0.9 for faster convergence. Additionally, it defines a cross-entropy loss function and a learning rate scheduler with a step size of 1 and a decay factor of 0.7. We didn't use batch, the reason will be provided in the result section. Accuracy and Loss will be also provided in the result section.

D. Usage of Farneback Algorithm

We visualize and save optical flow data calculated using the Farneback algorithm. By using OpenCV 'VideoWriter' object to save the visualization results of optical flow, we can transform pixel displacement data in the x and y directions from Cartesian coordinates to polar coordinates. In the HSV color space, Saturation is set to the maximum to enhance color clarity. The Magnitude is normalized to values between 0 and 255. Additional code is written to convert the HSV image to BGR format for writing to the video.

As a result, the Farneback algorithm models the movement of each pixel with a second-degree polynomial, estimating optical flow for the entire image based on this modeling, so we can visually represent optical flow data computed through the Farneback algorithm and save it in an easily understandable form as a video. By utilizing the HSV color space, it provides an intuitive representation of the direction and magnitude of pixel movements.

IV. Result

A. Optical Flow

Figure 4 .Lucas-Kanade Algorithm Output

Python implementation of the Lucas-Kanade algorithm demonstrates precise tracking of corner motion detected through the Harris Corner detector.

To facilitate the training of the neural network, **the Farneback algorithm will be employed** during the data preprocessing stage. If there is an absence of applying optical flow, the model tends to overlook dynamic relationships between frames, making it susceptible to missing movements or changes. Consequently, the accuracy of anomaly detection models is generally diminished in videos featuring motion or unusual patterns. **Hence, applying optical flow is expected to significantly enhance the model's performance.**

B. Neural Network

In this section, we will compare three different training methods that we employed for the neural network while explaining the rationale behind configuring our neural network model as depicted in Figure 3.

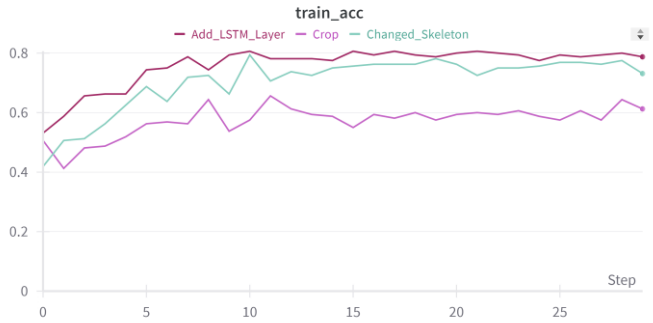

Figure 5 .Train Accuracy

Figure 6. Train Loss

In "Add_LSTM_Layer", we utilized a structure identical to that depicted in Figure 3. For "Crop", a preprocessing step was undertaken to eliminate unnecessary black borders in the data, following a structure consistent with Table 1. The term "Changed_Skeleton" denotes instances where the structure remained unchanged, aligning with the configuration outlined in Table 1.

Training the dataset using the model presented in Figure 3 yielded promising results. The training loss was reduced to 0.4491, and the training accuracy demonstrated a peak performance of 80.625%.

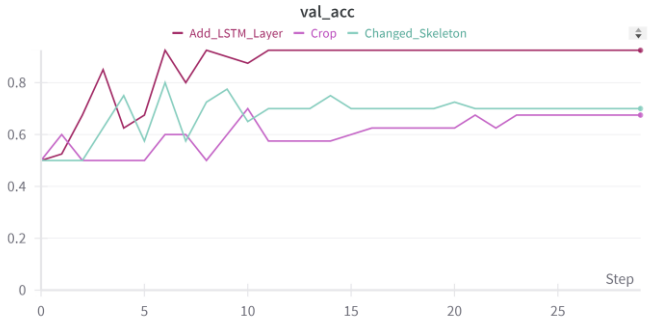

Figure 7. Validation Accuracy

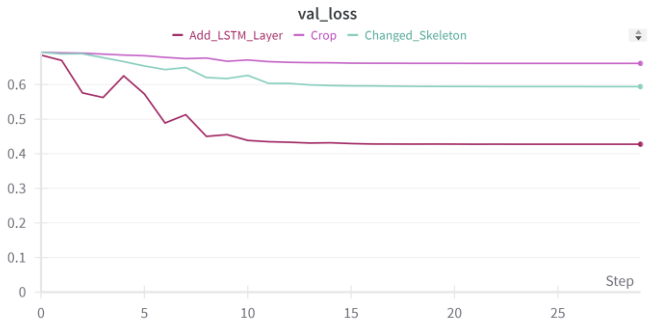

Figure 8. Validation Loss

To assess the model's performance and address concerns related to overfitting, validation was conducted. The validation loss decreased to 0.42757, and the validation accuracy reached a peak performance of 92.5%. This indicates that the model exhibited robust performance and **worked well for different datasets.**

To enhance performance, experiments were conducted by adjusting the batch size; however, the results were not superior to those obtained without using batches. Therefore, the training algorithm was executed as described in Section III - C. As observed in Figures 5 to 8, the customized model with **the addition of LSTM demonstrated an accuracy improvement of over 20% compared to the conventional methods**. Consequently, it was confirmed that the LSTM customized model outperformed the baseline model in Table 1.

Additionally, to assess the performance of the model, a task was undertaken to draw the AUROC curve, which is a common metric used to evaluate the performance of classification models. We used the model that exhibited the best performance among the 30 epochs to draw the AUROC curve.

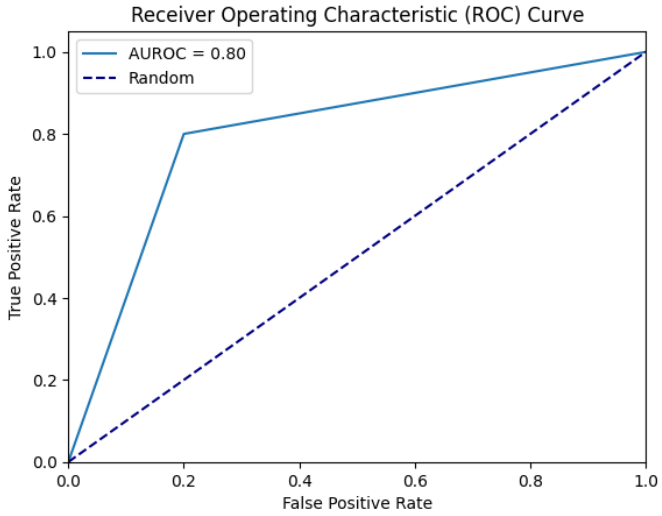

Figure 9. AUROC Curve

According to Figure 9, the Test AUROC of this customized model is 0.8, indicating a performance superior to the provided example of 0.6.

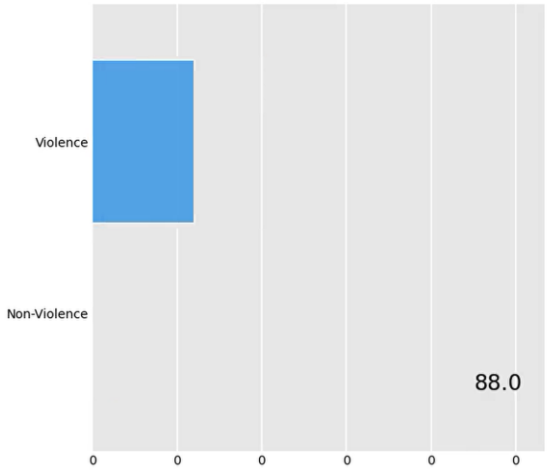

Figure 10. Visualization of Detection

The results of the visualization confirmed that the model is functioning properly.

V. Conclusion

In this project, we developed an AI model for detecting violent behavior in video. We utilized optical flow methods, the Lucas-Kanade algorithm for optical flow processing, and the farneback algorithm for training the detection model. We customized the given model with an LSTM cell to improve the performance of classification. The model demonstrated promising results, with a training accuracy of 80.625% and a validation accuracy of 92.5%. As a result, the model exhibited robust performance and resilience to overfitting issues. The Test AUROC of the model was 0.8, indicating superior performance compared to the provided example of 0.6. Overall, the developed AI model has the potential to provide fast alerts in time and improve surveillance in modern cities.

References

[1] Lucas, Bruce D., and Takeo Kanade. "An iterative image registration technique with an application to stereo vision." *IJCAI'81: 7th international joint conference on Artificial intelligence*. Vol. 2. 1981.

[2] Harris, Chris, and Mike Stephens. "A combined corner and edge detector." *Alvey vision conference*. Vol. 15. No. 50. 1988.

[3] Farnebäck, Gunnar. "Two-frame motion estimation based on polynomial expansion." *Image Analysis: 13th Scandinavian Conference, SCIA 2003 Halmstad, Sweden, June 29–July 2, 2003 Proceedings 13*. Springer Berlin Heidelberg, 2003.

[4] Cheng, Ming, Kunjing Cai, and Ming Li. "RWF-2000: an open large scale video database for violence detection." *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE, 2021.