# Deep learning methods and application

**Technical report of the project**

Hermes NDJENG, Ramses APPESE.

**Abstract**

*The paper is a summary of the methodology that was used to achieve our goals. We will present the concept used and also the reasons of our choices*

**Introduction**

The rapid advancement of video surveillance technologies has significantly influenced public safety and security systems worldwide. With the increasing prevalence of video streams in urban environments, the need for efficient and accurate detection of violent activities within these streams has become paramount. This project aims to address this critical issue by developing a sophisticated deep learning model capable of automatically identifying violent activities in video streams. Leveraging the power of convolutional neural networks (CNNs) and recurrent neural networks (RNNs), our model integrates spatial and temporal data analysis to accurately pinpoint instances of violence among individuals or groups within video footage.

The core of our approach lies in the integration of spatial and temporal information, a novel concept that has been gaining traction in the field of violent activity detection. This integration allows our model to analyze the complex patterns and sequential information inherent in video data, thereby enhancing its ability to detect violent activities with high accuracy. Our methodology involves a three-staged end-to-end framework, starting with the detection of persons in the video stream, followed by the extraction of spatiotemporal features using a 3D CNN model, and finally, the classification of these features to predict violent activities.

By providing a comprehensive solution for the automated detection of violent events in video streams, we aim to contribute to the development of safer and more efficient surveillance systems. This technical documentation will detail the architecture of our model, the training process, dataset preparation, and evaluation results, offering a thorough understanding of our approach and its potential applications.

**Data Collection and Preprocessing**

The foundation of our model's performance lies in the quality and diversity of the data it is trained on. For this project, we have curated a dataset consisting of 1000 videos of violent events and 1000 videos of non-violent events, sourced from YouTube and other sources. This dataset is a critical asset, as it provides a broad spectrum of scenarios that our model will learn to differentiate between. The violent videos include real-life street fight situations recorded under various environmental conditions, while the non-violence videos encompass a wide range of human activities, such as sports, eating, walking, and more. This diversity is crucial for training a robust model that can generalize well across different contexts and conditions.

**Video Preprocessing**

The preprocessing of video data is a critical step in preparing it for input into our deep learning model. The function “*preprocess\_video”* is designed to handle this task. It takes a video file path and optional parameters for frame interval and target frame size. The frame interval parameter allows us to reduce the frame rate of the video, which can significantly decrease the computational load and the size of the dataset without losing essential information. The target frame size is set to ensure that all frames are of a consistent size, which is a requirement for input into most deep learning models.

The preprocessing function reads the video frame by frame using OpenCV's VideoCapture class. It then checks if the current frame number is a multiple of the specified frame interval. If it is, the frame is resized to the target size and added to a list of frames. This process continues until the end of the video is reached. The function returns a list of preprocessed frames, ready for further processing or model training.

**Data Augmentation**

Data augmentation is a technique used to artificially increase the size of the training dataset by creating modified versions of the existing data. This can help improve the model's ability to generalize and reduce overfitting. The function data\_augmentation demonstrates a simple form of data augmentation by flipping frames horizontally. This is just one example of the many possible augmentation techniques that could be applied, such as rotation, scaling, or adding noise.

In the context of our project, data augmentation is particularly important because it can help the model become more robust to variations in the orientation of violent events within the video footage. By training the model on both original and augmented frames, we can enhance its ability to detect violent activities in a wider range of scenarios.

**SPATIAL FEATURE EXTRACTION:**

**InceptionV3: A Deep Dive**

InceptionV3, part of the Inception family of models, is a convolutional neural network (CNN) architecture that has been widely recognized for its effectiveness in image recognition tasks. Introduced by Google in 2015, InceptionV3 is designed to automatically learn spatial hierarchies of features from input images, making it highly efficient and accurate for identifying objects within images.

**Advantages of InceptionV3:**

**Efficiency:** InceptionV3 is designed to be computationally efficient while maintaining high accuracy. It achieves this by using a network architecture that reduces the number of parameters and computational cost.

**Accuracy:** The model has been proven to be highly accurate in image classification tasks, often outperforming other models in terms of both speed and accuracy.

**Flexibility:** InceptionV3 can be easily adapted for different image recognition tasks by fine-tuning the last few layers of the network.

**Pre-trained Weights:** The model comes with pre-trained weights on the ImageNet dataset, which can be fine-tuned on a specific task, significantly reducing the training time and improving performance.

**Why InceptionV3?**

The choice of InceptionV3 for our project is justified by its proven effectiveness in image recognition tasks, its efficiency, and the availability of pre-trained weights. By leveraging the pre-trained InceptionV3 model, we can significantly reduce the training time and improve the accuracy of our model. The model's ability to learn spatial hierarchies of features makes it particularly suited for extracting spatial features from video frames, which is crucial for our task of detecting violent events.

**Spatial Feature Extraction:**

Spatial feature extraction is a critical process in computer vision and deep learning, particularly for tasks involving video streams. It involves using a model to identify and extract important spatial information from images or video frames. This information can include the location, shape, and orientation of objects within the image. In the context of our project, spatial feature extraction is used to extract features from individual video frames that capture the spatial characteristics of the scene.

These extracted features are then used as input for further analysis. By focusing on spatial features, our model can learn to recognize patterns and relationships between objects in the scene, which is crucial for accurately identifying violent activities.

In our implementation, we use the InceptionV3 model to extract spatial features from video frames. We configure the model to output features from the second-to-last layer, which is a common practice in transfer learning. This allows us to leverage the model's ability to learn complex spatial features while adapting it to our specific task of violent event detection.

**Experiment/Methodology:**

The methodology employed involves two main steps: frame preparation and feature extraction. Each video frame is prepared by converting it to the RGB color space, resizing it to a standard size (299x299 pixels), and normalizing the pixel values to a range between 0 and 1. This standardization ensures that the input data is consistent and compatible with the pre-trained model. Following the preparation, the frame is passed through a InceptionV3.

The feature extraction process is applied to a sequence of frames from each video, with the number of frames in each sequence determined by the SEQUENCE\_LENGTH parameter. To accommodate videos of different lengths, the code calculates a skip\_frames\_window, ensuring that the model processes a balanced number of frames from each video. If a video's sequence is shorter than the specified SEQUENCE\_LENGTH, the sequence is padded with zeros to match the required length. This ensures that all sequences are of the same length, facilitating batch processing in deep learning models.

**Results/Analysis/Discussion:**

The results of this methodology demonstrate the effectiveness of using a pre-trained model for spatial feature extraction from video frames. The extracted features serve as a compact representation of the spatial characteristics of each frame, enabling the model to learn patterns and relationships between objects in the scene. This approach has shown promising results in the detection of violent events within video streams, showcasing the potential of leveraging pre-trained models for video analysis tasks.

Before proceeding to temporal feature extraction, an initial step involves extracting features from a subset of videos, specifically 500 videos of non-violence and 500 videos of violence. This step is crucial for establishing a baseline dataset that will be used for training the model. The rationale behind this approach is to ensure that the model is trained on a balanced dataset, which is essential for its ability to generalize well to unseen data.

The feature extraction process is performed using the frames\_extraction function. The extracted features are then saved in a NumPy array, which is a compact and efficient format for storing large datasets.

The extracted features are saved to disk using the *np.save* function, which serializes the NumPy array to a binary file with a *.npy* extension. This allows for efficient storage and retrieval of the data, as well as compatibility with other tools and libraries that can read .npy files. The paths provided to the np.save function specify where the files should be saved, in this case.

Once the features have been saved, they can be loaded back into memory using the np.load function. This function deserializes the binary file back into a NumPy array, allowing for further processing or analysis. The paths provided to the np.load function specify the location of the saved files.

This process of feature extraction and data handling is a critical step in the project, as it sets the foundation for the subsequent steps of model training and evaluation. By extracting and saving the features from a representative sample of videos, the project ensures that the model is trained on a dataset that is representative of the types of videos it will encounter in real-world applications.

**Temporal Feature Extraction with LSTM**

Temporal feature extraction is a critical step in video analysis, especially when dealing with sequential data like video frames. Long Short-Term Memory (LSTM) networks are a type of recurrent neural network (RNN) that are particularly well-suited for this task. LSTMs can learn and remember patterns over time, making them ideal for analyzing sequences of data, such as video frames, where the order of the data points is important.

In this section, we define a Bidirectional LSTM model for video classification. Bidirectional LSTMs process the input sequence in both forward and backward directions, allowing the model to capture patterns that may be present in both directions. This can be particularly useful in video analysis, where the context of an event may be influenced by both past and future frames.

**Creating LSTM Model and Preparing Data**

The first step in preparing the data for training is to create labels for the videos. Since we have 500 videos of violence and 500 videos of non-violence, we create an array of zeros for the violence labels and an array of ones for the non-violence labels. These labels are then combined with the corresponding features to form the input data X and the target labels y.

The data is then split into training and testing sets using the *train\_test\_split* function from *sklearn.model\_selection*. This ensures that the model is trained on one portion of the data and tested on a separate portion, allowing for an unbiased evaluation of its performance.

The input data is reshaped to match the expected input shape of the LSTM model. In this case, each sample is a sequence of 16 frames, each with 2048 features extracted from the spatial feature extraction step.

**LSTM Model Definition using Keras Functional API**

The LSTM model is defined using the Keras Functional API, which allows for the creation of complex models with multiple inputs and outputs. The model consists of two Bidirectional LSTM layers, each followed by batch normalization and dropout for regularization. The output of the final LSTM layer is passed through a dense layer with a ReLU activation function, and finally, a dense layer with a sigmoid activation function to output the classification result.

**Compiling The Model**

The model is compiled with the Adam optimizer and binary cross-entropy loss function, which is suitable for binary classification tasks.

The choice of the Adam optimizer and binary cross-entropy loss function for compiling the model is rooted in their effectiveness and suitability for binary classification tasks.

**Adam Optimizer:**

The Adam optimizer is a popular choice for training deep learning models due to its efficiency and effectiveness. It combines the advantages of two other extensions of stochastic gradient descent: AdaGrad and RMSProp. Adam computes adaptive learning rates for different parameters, which makes it well-suited for problems with sparse gradients or noisy data. This adaptability allows the model to converge faster and achieve better performance, especially in complex models like LSTMs used for video classification.

**Binary Cross-Entropy Loss Function:**

Binary cross-entropy, also known as log loss, is specifically designed for binary classification problems. It measures the dissimilarity between the predicted probability distribution and the true binary labels of a dataset. This loss function is particularly useful for our task because it penalizes the model based on how well it predicts the probability of an event occurring (violence in this case) as opposed to not occurring. The binary cross-entropy loss function is the negative average of the log of corrected predicted probabilities, which means it rewards the model for making correct predictions and penalizes it for incorrect predictions. This makes it an ideal choice for training models to classify data into two distinct classes, such as violence and non-violence in video frames.

**Why These Functions?**

**Efficiency and Effectiveness:** The Adam optimizer is known for its efficiency and effectiveness in training deep learning models, making it a good choice for our LSTM model.

**Suitability for Binary Classification:** Binary cross-entropy is specifically designed for binary classification problems, making it the ideal loss function for our task of classifying videos as containing violence or not.

**Robustness to Noisy Data:** Both Adam and binary cross-entropy are robust to noisy data, which is common in video analysis tasks where the quality of the video frames can vary.

The model is then trained on the reshaped training data for 15 epochs, with a batch size of 32. The model's performance is evaluated on the testing data during training.

**Saving the Model:**

After training, the model is saved to disk using the torch.save function. This allows for the model to be loaded and used later for further analysis or deployment.

**Violence Detection Model Evaluation and Testing Report:**

This report outlines the evaluation and testing of a violence detection model, focusing on its performance metrics and the interpretation of these metrics to assess the model's effectiveness. The model was trained on a dataset of videos, classified into two categories: violence and non-violence. The evaluation process includes model accuracy, classification report, confusion matrix, ROC-AUC curve, and precision-recall curve.

**Model Evaluation:**

The model was evaluated on a test set to assess its accuracy. Accuracy is a fundamental metric that measures the proportion of correct predictions made by the model out of all predictions. It provides a straightforward measure of the model's performance but can be misleading if the classes are imbalanced.

**Testing with Unseen Videos:**

The model was tested on unseen videos, both containing violence and non-violence. This step is crucial for understanding how well the model generalizes to new, unseen data. The videos were processed to extract features, which were then reshaped to match the input shape expected by the model. This process is essential for preparing the data in a format that the model can process.

**Model Predictions:**

The model was used to predict the class of the unseen videos. The predictions were made for both violence and non-violence videos, and the results were compared to the actual classes. This step is crucial for understanding the model's ability to classify new data correctly.

**Classification Report:**

A classification report was generated to evaluate the model's performance in terms of precision, recall, f1-score, and support for each class. Precision measures the proportion of true positive predictions out of all positive predictions, while recall measures the proportion of true positive predictions out of all actual positive instances. The F1 score is the harmonic mean of precision and recall, providing a single metric that balances both. The support is the number of actual occurrences of the class in the test set.

**Confusion Matrix:**

A confusion matrix was created to visualize the performance of the model. It shows the number of true positives, true negatives, false positives, and false negatives. This matrix provides a clear picture of the model's performance, especially in terms of its ability to correctly classify positive and negative instances.

**ROC-AUC Curve:**

The Receiver Operating Characteristic (ROC) curve and the Area Under the Curve (AUC) were calculated to evaluate the model's ability to distinguish between the two classes. The ROC curve plots the true positive rate (sensitivity) against the false positive rate (1-specificity) at various threshold settings. The AUC-ROC score is the area under the ROC curve and ranges between 0 and 1, with 1 representing a perfect classifier and 0.5 representing a random guess. This metric is particularly useful for binary classification problems, especially when the classes are imbalanced.

**Precision-Recall Curve**

The precision-recall curve was plotted to evaluate the model's performance in terms of precision and recall, especially in scenarios where the positive class is imbalanced. This curve is particularly useful for assessing the model's performance in terms of precision and recall, providing a more nuanced view of the model's performance than the ROC curve, especially in cases where the positive class is rare.