REPUBLIC OF TÜRKİYE
YILDIZ TECHNICAL UNIVERSITY
DEPARTMENT OF COMPUTER ENGINEERING

VIOLENCE DETECTION IN VIDEOS

20011620 — Esma Nur Ekmekci
19011904 — Muhammad Nasir Sabır

SENIOR PROJECT

Advisor
Asst. Prof. Hafiza İrem TÜRKMEN ÇİLİNGİR

April, 2024

ACKNOWLEDGEMENTS

We would like to express our sincere thanks to Asst. Prof. Hafiza İrem TÜRKMEN ÇİLİNGİR for her valuable support during the development of our project.

Esma Nur Ekmekci
Muhammad Nasir Sabır

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

CNN	Convolutional Neural Network
LSTM	Long Short-Term Memory
SVM	Support Vector Machine
MFCC	Mel-Frequency Cepstral Coefficients

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

VIOLENCE DETECTION IN VIDEOS

Esma Nur Ekmekci
Muhammad Nasir Sabır

Department of Computer Engineering
Senior Project

Advisor: Asst. Prof. Hafiza İrem TÜRKMEN ÇİLİNGİR

The aim of this work is to improve the detection of violent scenes from security camera footage or fight scenes during hockey matches. For this purpose, techniques based on deep learning are utilized. The approach first focuses on combining CNN and LSTM networks to extract temporal and spatial information from video frames. At the same time, pre-trained models such as VGG16 and ResNet are used to measure and improve the robustness of the methods.

Since a complete and high-quality dataset is not available, we decided to use two different datasets to examine how various data sources affect our model building process and to ensure that our models generalize to a wide range of violence scenarios. These are the Hockey Fights dataset and the SCVD dataset.

The system is as follows: The video data needs to go through a pre-processing stage before being used in the model. The processed video frames are then sent to CNN layers to extract key features. These features are then fed into LSTM networks that help analyze the data in sequences and classify the video. Experimental results show that these approaches achieve reasonable to good accuracy in recognizing violent content.

Keywords: Violence Detection, Convolutional Neural Networks, Long Short-Term Memory,Video Analysis, Video Surveillance, Deep Learning, VGG16, ResNet

ÖZET

VİDEODAN ŞİDDET TESPİTİ

Esma Nur Ekmekci
Muhammad Nasir Sabır

Bilgisayar Mühendisliği Bölümü
Bitirme Projesi

Danışman: Yardımcı Doçent Hafiza İrem TÜRKMEN ÇİLİNGİR

Bu çalışmadaki amaç, güvenlik kamerası görüntülerinden şiddet içeren sahneleri veya hokey maçları sırasındaki dövüş sahnelerini tespit etme konusunda iyileştirmeler yapabilmektir. Bunun için derin öğremeye dayalı tekniklerden yaralanılmaktadır. Yaklaşımda ilk olarak video karelerinden zamansal ve mekansal bilgi çıkarmak için CNN ve LSTM ağlarını birleştirmeye odaklanılmıştır. Aynı zamanda yöntemlerin dayanıklılığını ölçmek ve artırmak için VGG16 ve ResNet gibi önceden eğitilmiş modeller kullanılmıştır.

Eksiksiz ve yüksek kaliteli bir veri seti mevcut olmadığından, çeşitli veri kaynaklarının model oluşturma sürecimizi nasıl etkilediğini incelemek ve modellerimizin geniş bir yelpazedeki şiddet senaryolarına genellenmesini sağlamak için iki farklı veri seti kullanılmaya karar verilmiştir. Bunlar Hockey Fights veri seti ve SCVD veri setleridir.

Kurulan sistem şu şekildedir: Video verilerinin model üzerinde kullanılmadan önce bir ön işleme aşamasından geçmesi gerekli. İşlenen video kareleri daha sonra temel özelliklerin çıkarılması için CNN katmanlarına gönderilir. Daha sonra bu özellikler, verilerin diziler halinde analiz edilmesine ve videonun sınıflandırılmasına yardımcı olan LSTM ağlarına besleniyor. Deneysel sonuçlar bu yaklaşımların şiddet içeren içeriği tanımada makul veya iyi doğruluk oranlarına ulaştığını gösteriyor.

Anahtar Kelimeler: Şiddet Tespiti, Evrişimsel Sinir Ağları, Uzun Kısa Süreli Bellek, Video Analizi, Video Gözetimi, VGG16, ResNet, Derin Öğrenme

1 Introduction

Violence recognition in videos is an important problem for a variety of applications, including video surveillance, content regulation, and child safety. An increasing amount of video data is being generated, which means that automated systems that can reliably identify violent content are needed more than before.

This research aims to create a system that can detect violent material in video recordings. When it comes to improving security and safety, this technology could be valuable in a variety of contexts. For example, an automated violence detection system may analyze the inputs in real time and flag potential scenarios that require human assistance. Thus, we may take action before the unwanted event happens. Moreover, we can control online videos that include violence and prevent them from spreading.

If a person uses physical force on someone to cause harm, that is considered violence. This action may include the use of weapons such as knives and guns. In this project, we will investigate circumstances in which individuals engage in fights.

2 Pre-review

In this chapter, we will present and discuss studies that have been conducted, along with a literature review and general explanations on the subject.

Since violence varies, researchers have focused on different aspects of it. Some may focus on blood detection, while others may use audio cues or violent actions such as kicking and slapping. There are many approaches in this case. SVM, Conv3D(3D CNN),

Audio and visual features are used in this case to detect violence. Audio information are derived from MFCC characteristics. The visual information come from SentiBank, Blood, and Motion features. To detect violence, binary SVM classifiers are trained on all of these features [1].

S. Amraee et al. made a study related to this subject. They wanted to use some particular descriptors. They firstly extracted particular attributes from the data and then removed items that are unneeded or repetitious. Then, they used HOG-LBP and HOF descriptors to see how things are seen and move in certain locations. These descriptors are powerful when it comes to getting an aspect of both visual appearance and motion patterns. They used a one-class SVM model based on these descriptors so that they can detect violent behavior. Finally, they assessed the performance of their technique using some metrics such as accuracy, AUC, and ROC [2].

In the project by T. Hassner et al., they used the Violent Flow, Hockey Fight, and Aslan datasets. They mainly focused on using flow vectors in a sequence of frames. To do this, they used an optical flow descriptor. Eventually, they used this data to show the difference between violent and non-violent scenes [3].

Some researchers make emphasis to use both CNN (Convolutional Neural Network) and LSTM (Long Short-Term Memory) since they have different abilities. CNN has the ability of getting spatial features like where things are in a video. LSTM has the ability of getting temporal aspects of a motion. So with both of them, we have a great tool to guess video content. They used Violence in Movies and Hockey Fight as datasets [4].

In the paper of Javad Mahmoodi and Afsane Salajeghe, they proposed a new method for violence detection systems. Their approach, named Histogram of Optical Flow Magnitude and Orientation (HOMO), observes motion between video frames. They convert frames to grayscale, calculate optical flow, and analyze changes in its magnitude and orientation. Significant changes are captured using binary indicators, and these indicators are combined into a HOMO descriptor. After that operations, they trained SVM to detect violent scenes. The system has promising performance compared to existing methods [5].

Qin Jin and his friends come up with a different proposal. He suggests analyzing both the image and the sound from the video. It takes the audio from the video and converts it into a 40-dimensional Mel Filter Bank (MFB). CNN learns models from these video features. These models are then given to SVM where prediction is performed. Each video is divided into segments and the specific decision for that segment is decided with the max-min pooling technique, and then the overall decision is made for that video based on these decisions. This study, which uses the MediaEval dataset, works better than the system that predicts only on audio and only on images [6].

3 Feasibility

A feasibility study was conducted to assess the project's practicality. Thanks to this assessment we know the required resources, our time plan, legal and economic concerns. This is a crucial step in our project.

3.1 Technical Feasibility

The project's technical feasibility has been assessed by describing its software and hardware needs. Assessing technological resources and infrastructure is crucial for successful project execution. The software and hardware feasibility of the system is listed in Section 3.1.1 and Section 3.1.2.

3.1.1 Software Feasibility

The software elements to be used for the creation of the system are listed below.

- MACOS
- Python
- Visual Studio Code
- Google Colab

Python is used for software feasibility due to its user-friendliness, effective environment, and well-designed libraries. We initially utilized Google Colab because it enables collaborative work on a shared code platform. However, we later switched back to Visual Studio Code for quicker results.

3.1.2 Hardware Feasibility

The technical specifications of the cloud and personal computers used in the development of the project are shown in the Table 3.1 and Table 3.2.

Table 3.1 Esma's Computer

Component	Specification
CPU	8-core CPU M2
GPU	8-core GPU
RAM	8 GB
Storage	256 GB SSD
Operating System	MacOS

Table 3.2 Nasir's Computer

Component	Specification
CPU	8-core CPU M1
GPU	8-core GPU
RAM	8 GB
Storage	512 GB SSD
Operating System	MacOS

After conducting some experiments on the project, we noticed that the training time is lengthy, causing a slowdown in our work. Therefore, we have opted to utilize Google Colab as our coding platform, which comes equipped with a default 16GB RAM.

These were the resources we had available for the project. Having access to more powerful CPUs and GPUs could have potentially improved the performance of our model.

3.2 Legal Feasibility

We used open source dataset, it was filmed on hockey games. The libraries used in the project were also open source. As a result of the examinations, no legal problems were detected.

3.3 Economic Feasibility

The dataset used for the project is accessible for free. Furthermore, there are no expenses associated with the software and dataset stages because the software utilized is free source and no additional licensed software is employed. There is nothing purchased for this project so there was no economic concern.

3.4 Time Feasibility

Our project timeline began with a consultation meeting where we defined project objectives. We planned out our strategic approach. After that, we discussed which approach would work best for the problem and started installing the necessary software. After installing the software, we got to work on our data processing and implying the methods.

Detailed time planning is given in Figure 3.1.

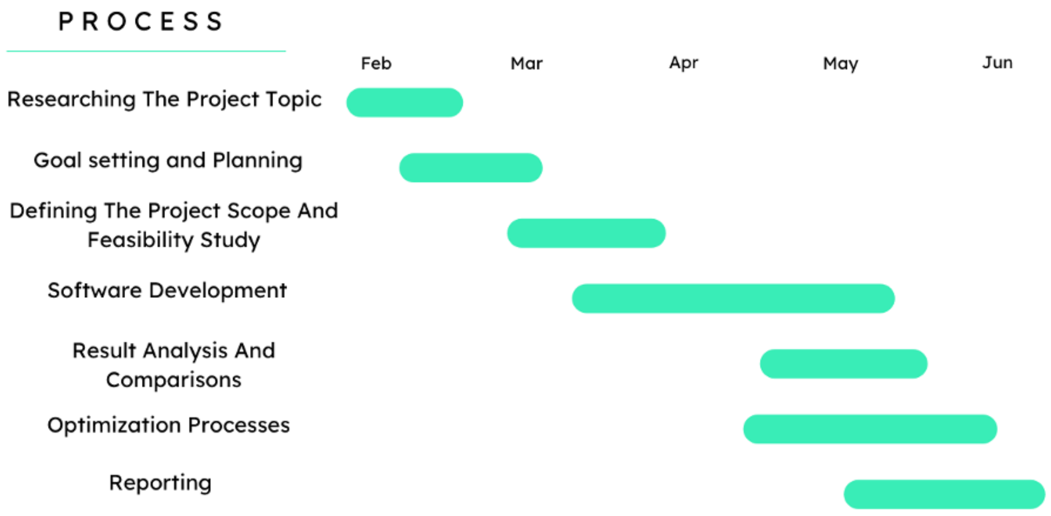


Figure 3.1 Gantt Diagram For Our Project

4 System Analysis

In this chapter, we will provide a comprehensive overview of the project's objectives. Selected dataset and the explanations, requirements for our software, system components, preferred models and performance metrics will be explained.

4.1 Goals

The study addresses video data management in safety tracking by introducing a deep learning and computer vision-based system for violence detection and categorization. The project's purpose is to overcome the limitations of manual video monitoring by developing an automated system capable of efficiently analyzing video data and detecting aggressive behavior. Automation could greatly improve security procedures.

This study aims to create a system where it can analyze a video and classify it into a class within 2 classes. It addresses video data management in safety tracking. To do that, the system will be using deep learning and computer vision-based techniques. This project has potential as a system which overcomes the limitations of manual video monitoring.

4.2 System Requirements

To do this project we require some tools. The system requirements for the violence detection project include hardware, software, data, model, testing, deployment, and documentation. Hardware should have adequate processing and storage capacity. Software needs include programming languages, frameworks, and libraries. For our project, they are summarized as **Python**, **TensorFlow**, **scikit-learn**, **NumPy**, **Matplotlib**, **OpenCV**, and **Pandas**.

We have used pre-trained models such as VGG19 and strategies to improve the performance of system.

4.3 Roadmap

The roadmap consists of preprocessing datasets, extracting features from video frames using a pre-trained model, building and training an LSTM-based classification model, evaluating its performance on a separate validation set, and analyzing the results to identify areas for improvement. Data preparation assures high-quality training data, whereas feature extraction saves time and resources by using pre-trained models. Training the LSTM model allows it to learn temporal dependencies in video data, and evaluation ensures unbiased performance assessment. Analyzing performance guides iterative improvements for developing a robust violence detection system.

Our general roadmap is given in Figure 4.1.

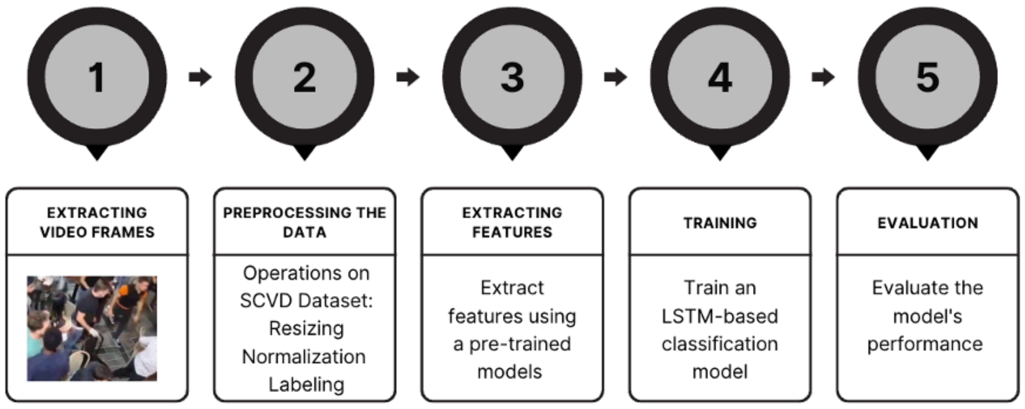

1
2
3
4
5
EXTRACTING VIDEO FRAMES
PREPROCESSING THE DATA
Operations on SCVD Dataset: Resizing Normalization Labeling
EXTRACTING FEATURES
Extract features using a pre-trained models
TRAINING
Train an LSTM-based classification model
EVALUATION
Evaluate the model's performance

Figure 4.1 Roadmap

4.4 Dataset Analysis

After searching around the literature, we saw there is a few well-established datasets accessible for this subject. Notably, two prominent datasets appeared from the research: one generated from cinematic sources, which had high-quality video content, and another derived from hockey matches, which included instances of player fights. In addition to these, we looked for alternative datasets on Kaggle and found one sourced from surveillance cameras. However, as compared to the previous datasets, the quality of this one was considerably lower. Despite initial efforts to use this dataset, we discovered significant challenges due to its complexity; for example, the beginning of fights often happened beyond the opening frame, resulting in the combination of nonviolent frames during model training.

After some experiments, we still have not obtained satisfactory results from the SCVD dataset, so we moved on to the Hockey Fight Dataset. We measured different parameters based on this dataset since it provided us with reliable results.

4.5 Method Analysis

There are different ways to solve this problem as we have seen in the literature review part. Researchers tried various methods to find the most effective solution to this problem. Support Vector Machines can be useful at detecting violence by learning decision boundaries between violent and non-violent variables; however, their linear nature may limit their ability to handle complex video data. On the other hand 3D CNNs provide a different aspect. It brings an advantage by directly evaluating video frames as 3D volumes. However, their computational cost can be high and this prevents us chosing this method. Combining CNNs for spatial feature extraction with LSTMs for temporal analysis offers the most promising approach, achieving high accuracy like 3D CNNs, but with better efficiency due to sequential frame processing. Also it should not be forgotten that CNN-LSTMs require significant training data and this can be challenging.

Distribution of violence detection methods is summarized in a graph by Omarov B. et al[7] and is given in Figure 4.2.

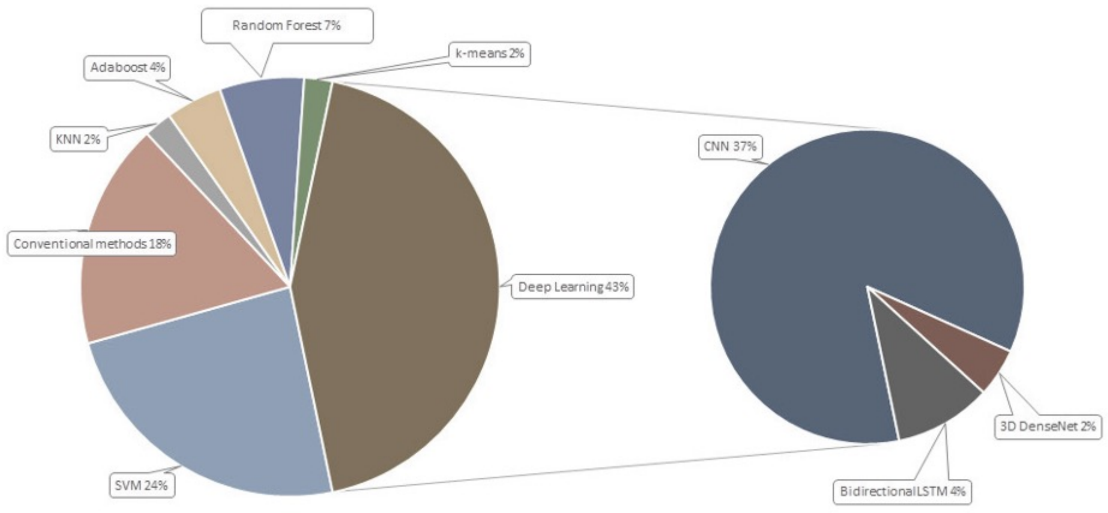


Figure 4.2 Distribution of Violence Detection Methods

4.6 Evaluation Parameters

This section discusses evaluation parameters for physical violence detection systems. Before explaining the metrics, we should know some terms.

In the context of our violence analysis project, we have used these metrics since they give insightful observations. They are important in our assessment. Accuracy reflects the proportion of correctly identified instances of each classes so it provides an overall aspect of our model's correctness in classification. The F1 score combines accuracy and recall. It offers a balanced evaluation. Precision indicates the ratio of accurately predicted instances of violence among all instances classified as violent, highlighting the importance of precise identification of violent cases. Recall measures the proportion of accurately predicted violent instances among all actual violent instances so it shows effectiveness in capturing all relevant instances of violence.

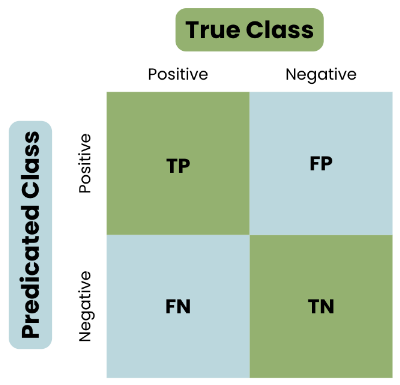


Figure 4.3 Confusion Matrix Explanation

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$
$$\text{Precision} = \frac{TP}{TP + FP}$$

In addition to these metrics, we use confusion matrix and it gives us a general idea of how the model produces results. In the given Figure 4.3 there is an explanation about what confusion matrix tells us [8]. As a result, these metrics guide us in improving the performance of our violence classification model.

4.7 Optimization Techniques

Various methodologies can be employed to optimize violence detection algorithms. To enhance accuracy, speed, and efficiency, different techniques have been tried. Refining deep learning models and fine-tuning parameters are considered essential strategies. Data augmentation techniques, such as image rotation, flipping, adjusting brightness and contrast, and inverting colors, can be utilized to increase the diversity of the training data.

In the experiments conducted, various types of changes have been tested. Image sizes have been adjusted based on specific needs, the number of frames fed into the LSTM has been varied, epoch sizes have been increased, and different pretrained models have been used. This topic will be explored further in later chapters.

5
System Design

5.1 Dataset Design

One of the datasets used in this project is taken from the publicly available (SCVD) Smart City CCTV Violence Detection Dataset on Kaggle and consists of three classes: “Normal”, “Violent” and “Weaponized” with approximately 500 videos. These videos include non-violent scenes, violent scenes (especially fights), and armed scenes involving harmful objects such as guns and knives. Due to the different scenarios and limited data specific to the weaponized category, we decided to simplify our focus by narrowing the categories to just ’Normal’ and ’Violent’.

Because the SCVD dataset has low pixel quality and is not particularly clean in terms of scene interpretation even for humans, and we want to ensure that our models work well with similar content-based data, we also used the HockeyFights dataset, which has considerably better image quality. This dataset is also available to the public on Kaggle. This dataset contains 1000 clips, 500 of which have fight scenes and the remaining 500 showing no fights. This balanced and high-quality dataset helped us with model evaluation.

Deep learning models for video tasks often require fixed-length input sequences. For this reason, we extracted exactly the same frame numbers from each video. We utilized a constant named SEQUENCE-LENGTH to ensure that the same amount of frames were taken from each movie. In order to use the most frames from each video, we performed another preprocessing step that divided videos into one-second slices. In this situation, we can extract 20 frames from each second of video, which is made up of at least 24 frames. This technique also expands our dataset which is what we have needed.

You can see some images of our data sets from the Figure 5.2 and Figure 5.1.

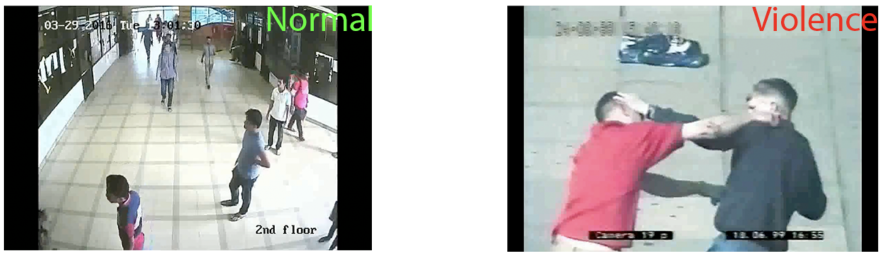


Figure 5.1 is an instance of SCVD dataset. Normal - Violence

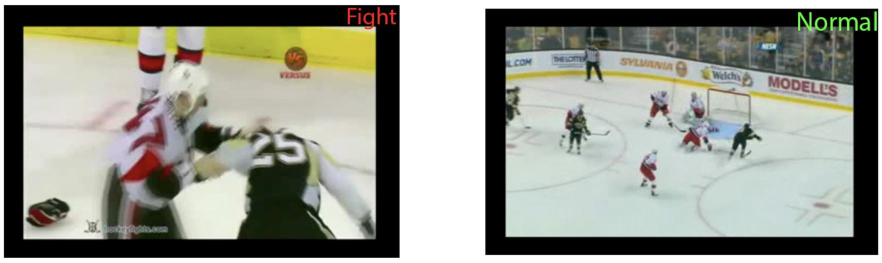


Figure 5.2 A sample instance of HockeyFights dataset. Fight - Normal

5.2 Preprocessing Data

In this part, we are slicing videos into seconds and then using them in a frame extraction method to prepare them for our CNN model or our pretrained models for feature extraction step. The image size is differ from our feature extraction model. For example, our CNN model using 64x64 RGB image and our pretrained models both VGG16 and ResNet using 224x224 RGB. The size of these frames is a hyperparameter that directly influences the model's performance.

Even with a small dataset, extracting frames for each training iteration is inefficient in terms of time. In order to resolve this issue, we developed alternate solutions for both models. In CNN, we stored the extracted frames as.npz files, whereas in VGG16 and ResNet, we saved them as.h5 files. Each iteration, we check to see if we have these files to avoid repeating the procedure. With these changes, we save a significant amount of time and can concentrate on model finetuning.

Example of creating extracted features file:

```
np.savez(os.path.join(save_path, 'dataset.npz'),
    features=features,
    labels=labels,
    video_files_paths=video_files_paths)
```

Both feature extracted files consists of these three information:

- **Features:** Extracted frames
- **File Paths:** Location of each video file
- **Labels:** Corresponding label for each frame, indicating whether it belongs to a non-violent or violent class

5.3 Software Design

Our project uses two different combined architectures to understand sequential data: a standard convolutional neural network (CNN) for feature extraction with an LSTM to understand sequential frames, and a pretrained model (VGG16 using ImageNet or ResNet) with an LSTM.

Like many other computer vision tasks, video classification requires an understanding of visual content. In the models we use, before LSTM extracts meaning from the sequential data, meaningful features are extracted from the image data we provide with our CNN and other Pre-Trained models. These models have ability to describe informations such as object detection, analyzing patterns, and finding edges. After this steps the combination of our model is now able to predict the video label for the entire video sequence.

The main motivation for using pre-trained models like VGG16 with ImageNet and also ResNet here is that we have a small data set. These pre-trained models are already trained on large datasets.

5.3.1 Convolution Neural Network

CNNs play an important role in video categorization by extracting spatial data from individual frames. CNNs consist of many convolutional filter layers and pooling layers. These layers can capture both low-level data, such as edges and textures, and larger elements, such as object parts and scenes. Also they are able to evaluate pixel values and patterns to detect important visual cues. In this way we extract The Spatial information that helps to interpret the visual content of each frame. Eventhough CNNs are excellent at feature extraction, they are not an only solution in capturing the sequential meanings of video data. Therefore, recurrent neural networks (RNNs) or Long Short-Term Memory (LSTM) networks are often used along with CNNs.

5.3.2 VGG16 Pretrained In ImageNet

Here is briefly explaination about how we utilize the VGG16 model. This pre-trained model have been pretrained on the ImageNet dataset for our feature extraction step in videos. Normally It is made up of 16 layers, 13 of which are convolutional and the remaining three are fully linked layers.

We employ VGG16 to extract high-level features from each video frame by leveraging its pretrained weights on ImageNet, which allows it to recognize a wide range of visual patterns. Specifically, we remove the final fully connected layers and use the output from the ‘fc2’ layer as the feature representation for each frame.

5.3.3 ResNet Pretrained In ImageNet

ResNet, like VGG16, is a pre-trained model. We wanted to experiment with ResNet in addition to CNN+LSTM. And we wanted to see if we could get better results from our data. These experiments helped us in comparisons of our models.

5.3.4 Long Short Term Memory

In this context, LSTM is the end point of our project workflow. Whether we are using CNN or a pre-trained model, we have to work with LSTM. Since we are dealing with videos, it is necessary to remember the information in the frames and determine the long-term relationships between the sequences. LSTMs derive their architecture from RNNs and belong to the RNN class. They have the ability to classify videos by motion and action by analyzing the relationships between frames.

6
Implementation

This project has two different legs as we have mentioned before. Our raw data passing through some preprocessing steps before being able to enter our CNN network in first leg, and Vgg16 network in second leg [9]. Lets check the implementation of models and analyze the results.

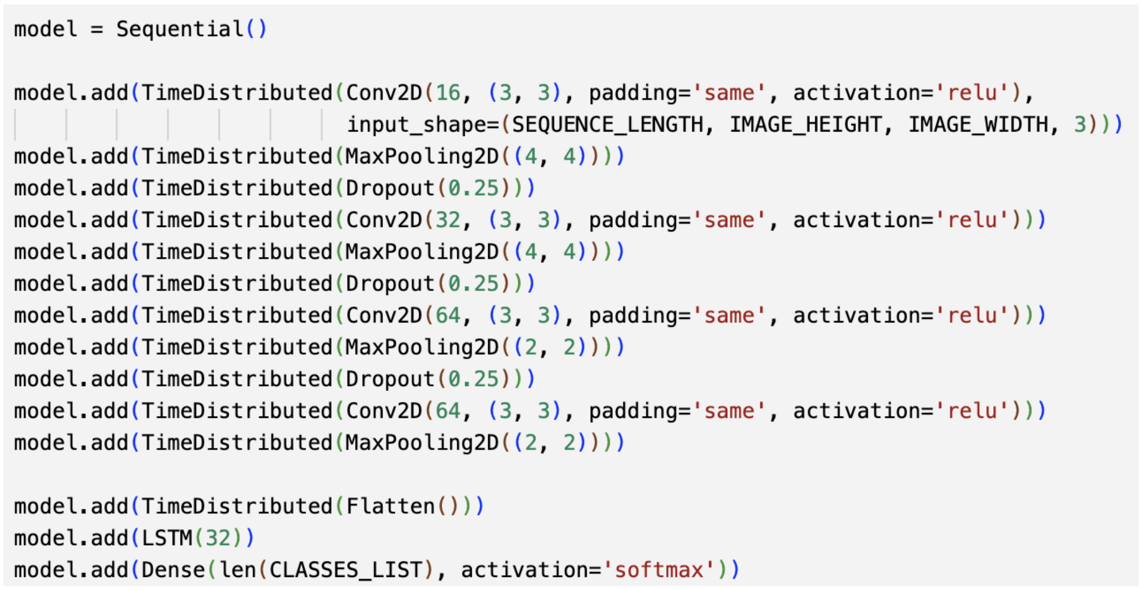
6.1 CNN and LSTM Architecture

Our custom-built CNN model has an architecture consisting of 4 convolutional layers. Each convolutional layer has max-pooling and dropout functionality to prevent overfitting. The duty of Max-pooling layers is to take the most important pieces of information from the image patches and decrease the overall size of the data. Dropout layers are randomly turning off some neurons during training to not rely too much on any single part of the data.

Our layers use 3x3 filters and the number of filters increases from 16 to 64 step by step. We use the activation function as Relu and the padding parameter is the same. Also, we used a method called padding to ensure the size of the images stays the same. The SCVD dataset lacks quality and rich color as it is captured by surveillance systems. For this reason, we implemented some augmentation methods from GitHub - okankop/vidaug to increase accuracy and decrease the overfitting where the model learns and memorizes too much training data.

After all these steps now the flattening layer takes all the 2D data from the feature maps and turns it into a long 1D vector. This step is necessary to prepare the data for the LSTM layer. And output layer makes the final prediction.

You can see the architecture of custom build CNN+LSTM model in Figure 6.1.



```
model = Sequential()

model.add(TimeDistributed(Conv2D(16, (3, 3), padding='same', activation='relu'),
                          input_shape=(SEQUENCE_LENGTH, IMAGE_HEIGHT, IMAGE_WIDTH, 3)))
model.add(TimeDistributed(MaxPooling2D((4, 4))))
model.add(TimeDistributed(Dropout(0.25)))
model.add(TimeDistributed(Conv2D(32, (3, 3), padding='same', activation='relu')))
model.add(TimeDistributed(MaxPooling2D((4, 4))))
model.add(TimeDistributed(Dropout(0.25)))
model.add(TimeDistributed(Conv2D(64, (3, 3), padding='same', activation='relu')))
model.add(TimeDistributed(MaxPooling2D((2, 2))))
model.add(TimeDistributed(Dropout(0.25)))
model.add(TimeDistributed(Conv2D(64, (3, 3), padding='same', activation='relu')))
model.add(TimeDistributed(MaxPooling2D((2, 2))))

model.add(TimeDistributed(Flatten()))
model.add(LSTM(32))
model.add(Dense(len(CLASSES_LIST), activation='softmax'))
```

Figure 6.1 CNN+LSTM Model Layers

In our architecture, we defined some metrics in the compilation step before training the model. The model is compiled by specifying the loss function, optimizer and performance measure. We use binary cross-entropy for the two-class problem (normal and severity), Adam optimizer for automatic learning rate adjustment, and accuracy as the performance metric.

While we defined our metrics, we also wanted to implement an early stopping callback to prevent overfitting. With this, we can stop the epochs if we are not receiving any improvements on the selected monitor. For instance, in this mode, we are monitoring validation loss.

Model Training is performed for 50 epochs using a batch size of 4 for the SCVD dataset and these metrics change when we use the HockeyFights dataset [10]. That is why we keep changing the metrics to find the best tuned settings for better results.

6.2 Pretrained Model and LSTM Architecture

After trying this CNN+LSTM method and not achieving satisfactory results, we pursued an alternative approach. We added a pretrained model into our code. Integrating a pretrained model into our code allowed us to get better results.

We used an image size of 224x224 pixels. The reason is it is a common standard for pretrained models such as VGG16 and ResNet. This choice ensured compatibility with these models. Also it improved transfer learning performance. In our previous model which does not have a pretrained model, the image size was 64x64 pixels.

We extracted 20 frames from each movie to find a balance between precise representation of the content and managable data load. This was also a common choice in violence detection systems.

Using the OpenCV library, we converted photos from BGR (OpenCV's default format) to RGB. Each frame was scaled to 224x224 pixels using the INTERCUBIC interpolation method. It ensured high-quality resizing. To save memory use, the photos were normalized to have pixel values between 0 and 1 and then transformed to the float16 data type.

The data was shuffled to improve the model's learning process. For classification, we used one-hot encoding to encode the labels: [1, 0] for violent movies and [0, 1] for nonviolent ones. This encoding format is required for many machine learning models. It enables the use of cross-entropy loss functions, which more correctly compute the difference between predicted and real labels in our system.

We mainly used the VGG16 model as a feature extractor. It is pre-trained model on the ImageNet dataset. There are several layers in models. By removing the model's last fully connected layer, we adapted it to our specific task. Transfer learning reduced training time and data needs in our project.

To optimize memory use and computation, we processed video frames in batches. The collected features and labels were stored in the .h5 file format. It is designed for rapid access and big data volumes.

Our customized model included an LSTM layer. The LSTM layer is needed for us to handle sequential data. It is ideal for video frame analysis. The model was trained using the Adam optimizer. The training data was divided between 80% training and 20% validation. The model was trained over different number of epochs and its performance was assessed using test data.

7 Performance Analysis

In this chapter, we will discuss the performance of our models in this project. Many approaches have been tried, and in the end, a model was chosen. We analyzed what is good and what is bad for our subject.

7.1 Performance of First Model

First we will give our analysis of the custom-built CNN LSTM model. In the beginning, to split our dataset randomly we used the shuffle metric. But in every iteration we run the program this randomization will be so different than the recent run. To prevent that we used seed constants which enable us to fit the randomization in every program run and we can be sure that our data is split randomly but the same in every iteration.

When we change the Seed Constant, there are quite different changes in the model. Test results decrease from 68 percent to 40 percent and change. In machine learning and scientific computing, where consistent results are required for debugging and verification, you ensure that the random numbers generated by libraries are consistent each time the code is run for reproducibility.

```
seed_constant = 27     #27 > 36 > 41
np.random.seed(seed_constant)
random.seed(seed_constant)
tf.random.set_seed (seed_constant)
```

Every time we change the seed constant above, our data changes in distribution as Table 7.1 ; Our main wish is for it to be more evenly distributed.

	Normal	**Violence**
Training	631	587
Testing	185	222

Table 7.1 Data Amount for Training and Testing when seed constant is 40

And here is the test result with data distribution of Table :

Accuracy: 0.4085 Loss: 0.6989

The data distribution is close to what we want in Table 7.2 Finally, we decided that we get the most successful results when the seed constant is 27, so we moved on in this way.

	Normal	**Violence**
Training	602	616
Testing	214	193

Table 7.2 Data Amount for Training and Testing when seed constant is 27

And here is the test result with data distribution of Table 7.2 :

Accuracy: 0.7033 - Loss: 0.5784

You can analyze the total loss and accuracy results of our CNN+LSTM model using SCVD data export and find the confusion matrix in Figure 7.3 and Figure 7.4 respectively.

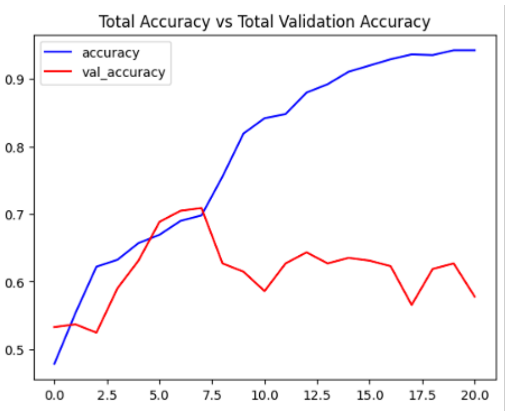


Figure 7.1 Accuracy of SCVD Dataset in CNN LSTM

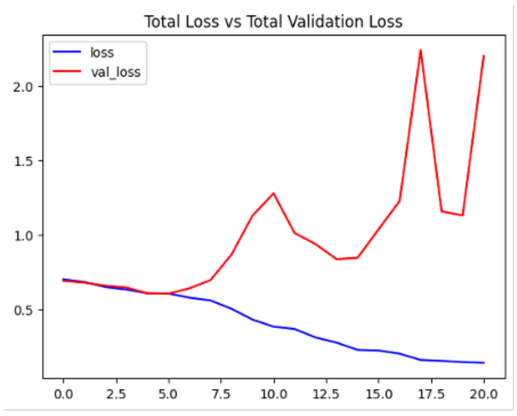


Figure 7.2 Loss of SCVD Dataset in CNN LSTM

Figure 7.3 Total Loss and Accuracy of SCVD Dataset in CNN LSTM

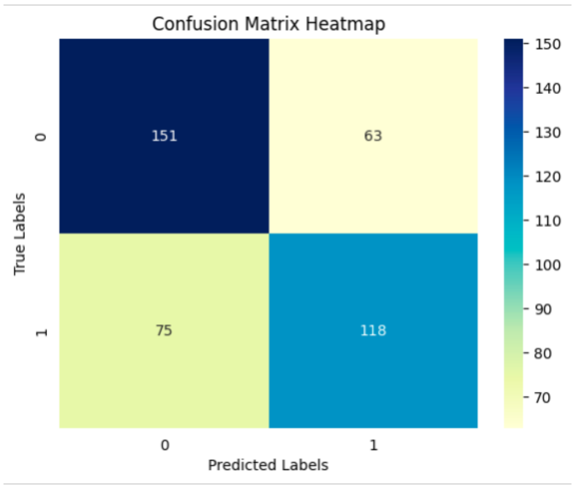
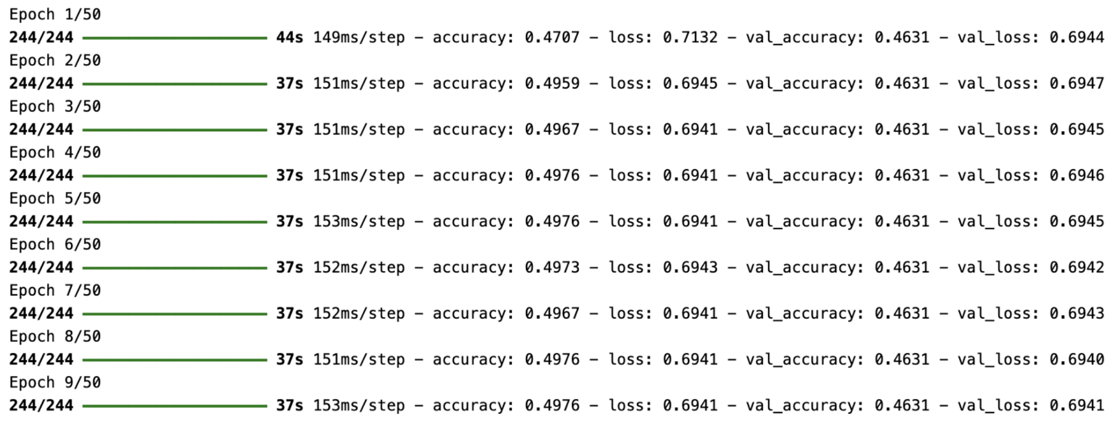


Figure 7.4 Confusion Matrix of SCVD Dataset in CNN LSTM model, with seed constant:27

After this improvement, we saw in our total accuracy and validation accuracy rates were not matching, This was indicating that our model is encountering with overfitting issues. To prevent this issue, we wanted to apply data augmentation techniques [11].

```
augmentor = va.Sequential([
    va.OneOf([va.HorizontalFlip(), va.VerticalFlip()]), # Randomly flip
    va.RandomTranslate(x=10, y=10),  # Randomly translate  10 pixels
    va.GaussianBlur(sigma=1),  # Randomly apply Gaussian blur
    va.Add(10),  # Randomly change brightness
    va.Multiply(0.8),  # Randomly change contrast
])
```

But when we added the augmentation techniques like above, our model started to show no improvements at all. You can check the accuracy and loss rates in each epoch in Figure 7.5.



```
Epoch 1/50
244/244 ━━━━━━━━━━━━━━━━━━━━ 44s 149ms/step - accuracy: 0.4707 - loss: 0.7132 - val_accuracy: 0.4631 - val_loss: 0.6944
Epoch 2/50
244/244 ━━━━━━━━━━━━━━━━━━━━ 37s 151ms/step - accuracy: 0.4959 - loss: 0.6945 - val_accuracy: 0.4631 - val_loss: 0.6947
Epoch 3/50
244/244 ━━━━━━━━━━━━━━━━━━━━ 37s 151ms/step - accuracy: 0.4967 - loss: 0.6941 - val_accuracy: 0.4631 - val_loss: 0.6945
Epoch 4/50
244/244 ━━━━━━━━━━━━━━━━━━━━ 37s 151ms/step - accuracy: 0.4976 - loss: 0.6941 - val_accuracy: 0.4631 - val_loss: 0.6946
Epoch 5/50
244/244 ━━━━━━━━━━━━━━━━━━━━ 37s 153ms/step - accuracy: 0.4976 - loss: 0.6941 - val_accuracy: 0.4631 - val_loss: 0.6945
Epoch 6/50
244/244 ━━━━━━━━━━━━━━━━━━━━ 37s 152ms/step - accuracy: 0.4973 - loss: 0.6943 - val_accuracy: 0.4631 - val_loss: 0.6942
Epoch 7/50
244/244 ━━━━━━━━━━━━━━━━━━━━ 37s 152ms/step - accuracy: 0.4967 - loss: 0.6941 - val_accuracy: 0.4631 - val_loss: 0.6943
Epoch 8/50
244/244 ━━━━━━━━━━━━━━━━━━━━ 37s 151ms/step - accuracy: 0.4976 - loss: 0.6941 - val_accuracy: 0.4631 - val_loss: 0.6940
Epoch 9/50
244/244 ━━━━━━━━━━━━━━━━━━━━ 37s 153ms/step - accuracy: 0.4976 - loss: 0.6941 - val_accuracy: 0.4631 - val_loss: 0.6941
```

Figure 7.5 Epochs after we applied augmentations

Later we decided to use only small augmentation techniques like "Fliplr" stands for "flip left-right" and "Affine(rotate=(-10, 10), shear=(-16, 16))" stands for rotation and shearing, which their effectiveness still is a mystery.

7.2 Performance of Second Model

Our new direction was changing the dataset. So we moved on to that dataset and made our implementations on it. We used pretrained models while implementing the model. We used both VGG16 and ResNet. To be able to do that, we changed some of our approaches. For instance, we changed the image size to 224, which was 160 before.

When using VGG16 we removed fc2 layer to use it in our specific purposes. And in ResNet we used avg_pool as a transfer layer. A new fully connected layer is added to replace the removed layer. It allowed the model to be trained for a new data set and a new task. We did our experiments with these two different models. one important finding we noticed: Resnet was faster at processing the given images. During inference, we found that ResNet is faster due to more efficient use of parameters and fewer fully connected layers.

Thanks to this new model, we don't need as many epochs. We need fewer trials thanks to the pretrained model. So we could speed up our work.

When experimenting with our data, we saved the processed videos in HDF5 File Format instead of constantly reprocessing the videos. This technique is used to store and manage large data sets. It made our work very easy in terms of access.

In the next chapter we will see the test results of our experiments.

8
Test Results

8.1 SCVD Dataset Experiments

Figure 8.3 and Figure 8.4 show the accuracy of our VGG16 model using the SCVD dataset.

Epoch 40: accuracy: 0.8185 - loss: 0.1293

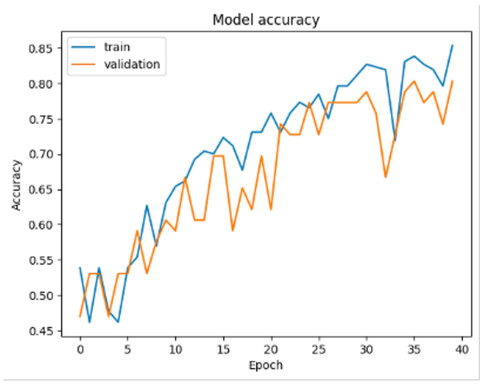


Figure 8.1 Accuracy of SCVD Dataset When Epoch is 40

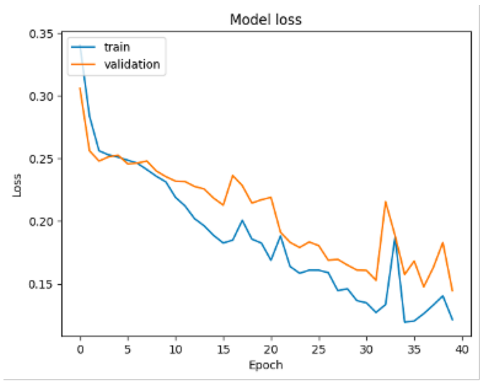


Figure 8.2 Loss of SCVD Dataset When Epoch is 40

Figure 8.3 Total Loss and Accuracy of SCVD Dataset When Epoch is 40 in VGG16

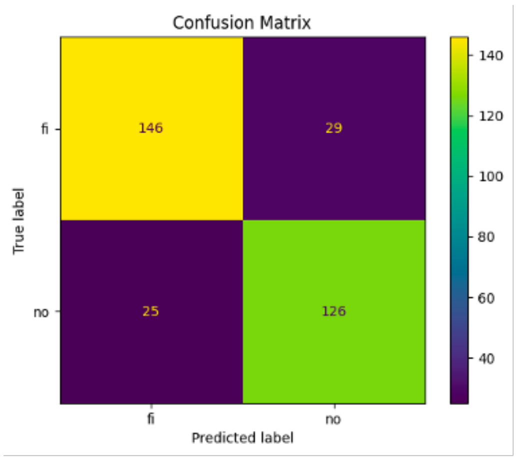


Figure 8.4 Confusion Matrix of SCVD Dataset When Epoch is 40 in VGG16

Figure 8.7 and Figure 8.8 show the accuracy of our VGG16 model using the SCVD dataset when epoch is 100.

Accuracy: 0.9416 - Loss: 0.0526

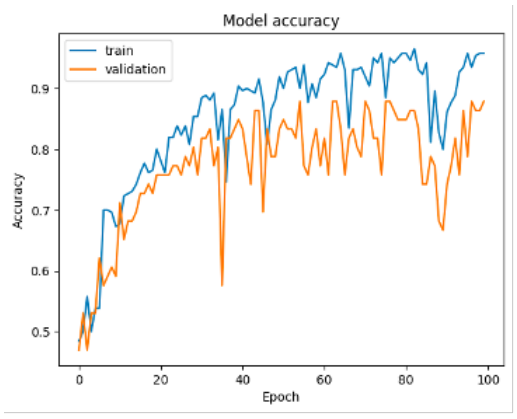


Figure 8.5 Accuracy of SCVD Dataset When Epoch is 100

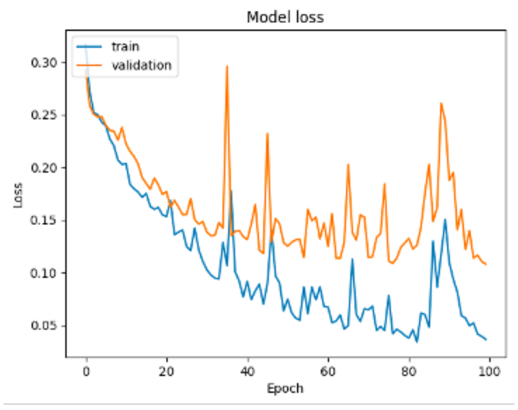


Figure 8.6 Loss of SCVD Dataset When Epoch is 100

Figure 8.7 Total Loss and Accuracy of SCVD Dataset When Epoch is 100 in VGG16

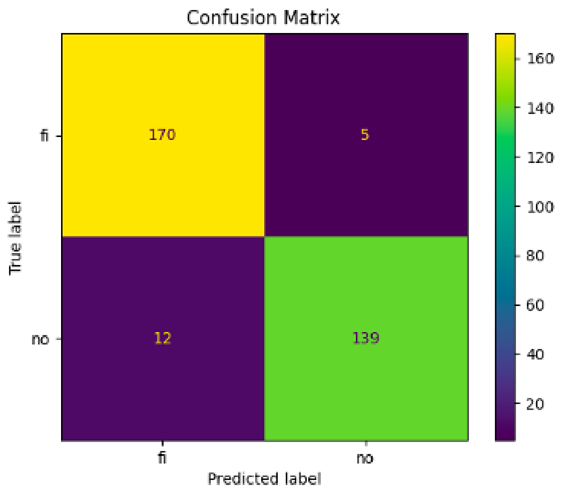


Figure 8.8 Confusion Matrix of SCVD Dataset When Epoch is 100 in VGG16

8.2 Hockey Fight Dataset Experiments

In this section, we analyzed based on scenarios. We used 80% trains for all scenarios. We used the Adam optimizer unless stated otherwise. Figure 8.11 and Figure 8.12 show the accuracy of our VGG16 model using the HockeyFight dataset.

8.2.1 VGG16, 20 Epochs

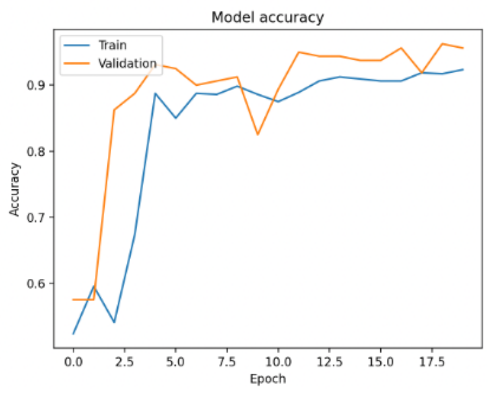


Figure 8.9 Accuracy of HF Dataset When Epoch is 20 Using VGG16

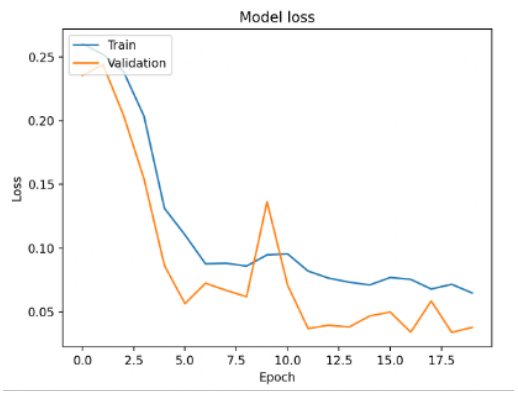


Figure 8.10 Loss of HF Dataset When Epoch is 20 Using VGG16

Figure 8.11 Total Loss and Accuracy of HF Dataset When Epoch is 20 Using VGG16

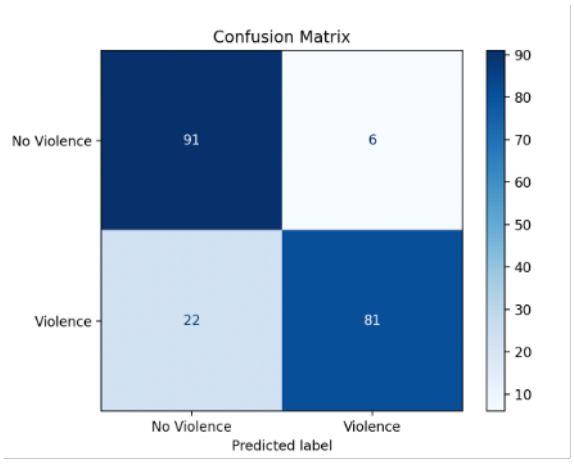


Figure 8.12 Confusion Matrix of HF Dataset When Epoch is 20 Using VGG16

8.2.2 VGG16, 20 Epochs, Added a Layer to LSTM Model

Adding another layer to lstm made the results slightly worse. After adding an extra layer, the model may be learning certain patterns and noise in the training data that do not generalize. We gave these results as Figure 8.15 and Figure 8.16

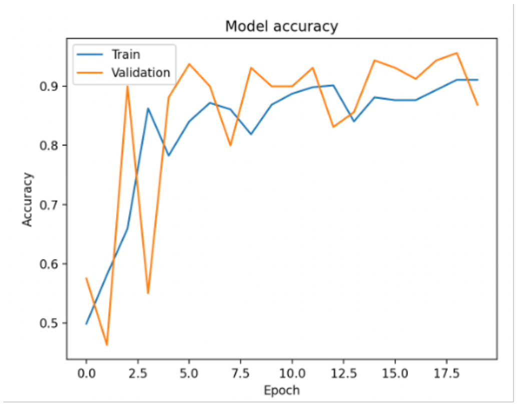


Figure 8.13 Accuracy of HF Dataset Using VGG16 and Added a Layer to LSTM

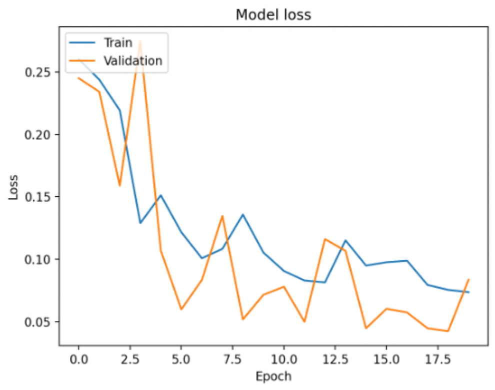


Figure 8.14 Loss of HF Dataset Using VGG16 and Added a Layer to LSTM

Figure 8.15 Total Loss and Accuracy of HF Dataset Using VGG16 and Added a Layer to LSTM

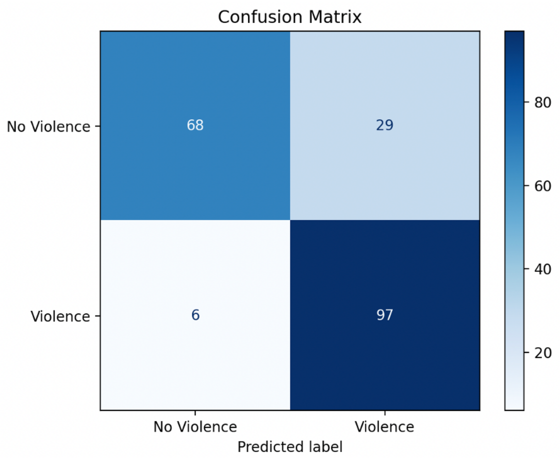


Figure 8.16 Confusion Matrix HF Dataset When adding new layer to LSTM

8.2.3 Optimizer='rmsprop', VGG16, 20 Epochs

Changing the optimizer worsened the result. Adam generally offered us higher accuracy and better overall performance because it combines both momentum and adaptive learning speed in his work. On Figure 8.19 and Figure 8.20 we gave related results to our above adjustments.

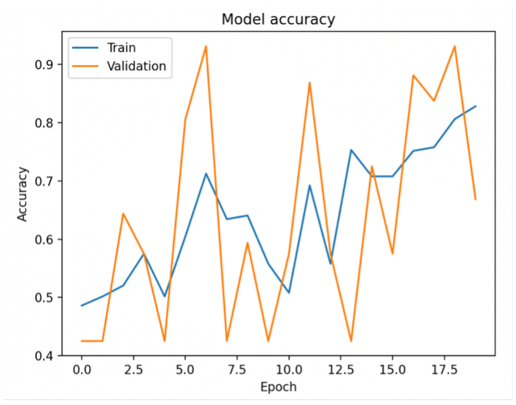


Figure 8.17 Accuracy of HF Dataset Using VGG16 and with 'rmsprop' Optimizer

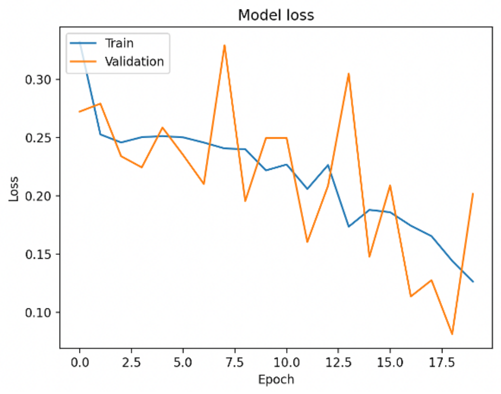


Figure 8.18 Loss of HF Dataset Using VGG16 and with 'rmsprop' Optimizer

Figure 8.19 Total Loss and Accuracy of HF Dataset Using VGG16 and with 'rmsprop' Optimizer

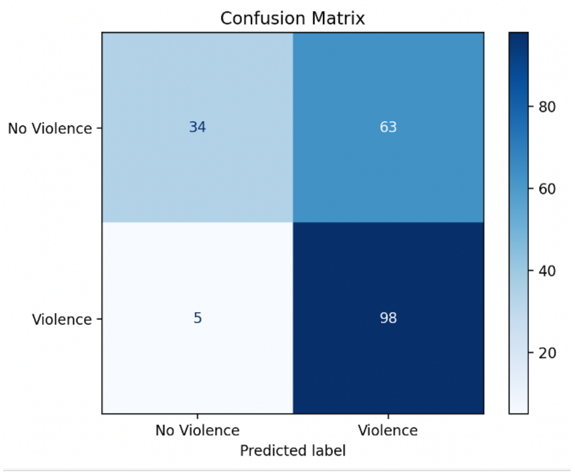


Figure 8.20 Confusion Matrix of HF Dataset Using VGG16 and with 'rmsprop' Optimizer

8.2.4 ResNet, 40 Epochs

ResNet was faster compared to VGG16. On Figure 8.23 and Figure 8.24 you can find ResNet results.

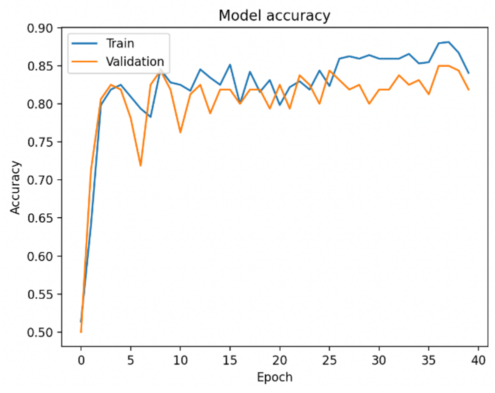


Figure 8.21 Accuracy of HF Dataset Using ResNet with 40 Epochs

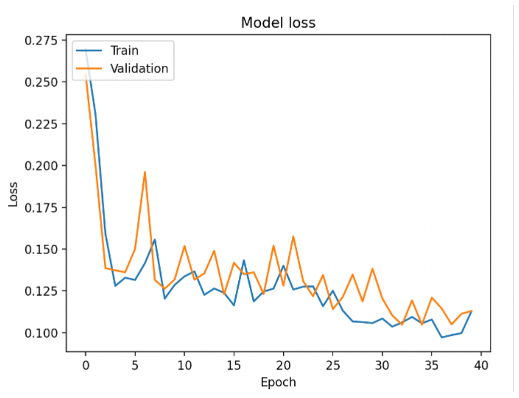


Figure 8.22 Loss of HF Dataset Using ResNet with 40 Epochs

Figure 8.23 Total Loss and Accuracy of HF Dataset Using ResNet with 40 Epochs

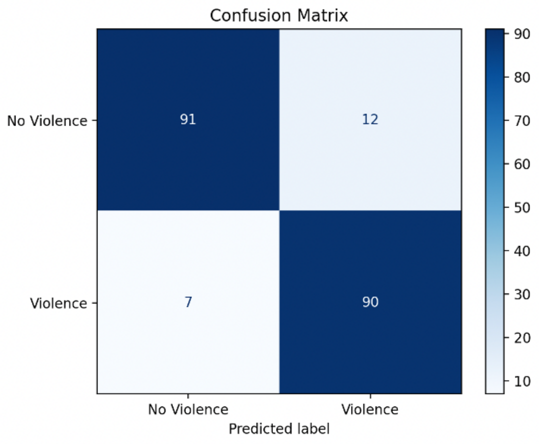


Figure 8.24 Confusion Matrix of HF Dataset Using ResNet with 40 Epoch

9 Conclusion

In addition to all the work done on the SCVD dataset, we started working in parallel with the HockeyFights dataset, which is a different and clean dataset, as we stated in the report. First, we applied this dataset to our existing CNN LSTM model and realized that the results were not due to the dataset. Therefore, we started to search for a different model that we could work on. Still, we tried to fine-tune the first model by experimenting with data augmentation to maximize the accuracy. Our tweaking attempts on the first model didn't work very well, and the model started returning different results each iteration, making it difficult to tell which direction we were going.

Hence, we could not make any progress in this area, we developed a model using VGG16 and upon the good results, we received there. we decided to change the model in the project and continue the work using a pre-trained model. Due to the limited data we had, we thought that using models prepared with large data sets was a good solution. We initially achieved 90 percent success on the Hockey Fights dataset. When we used the same model for the SCVD data set, we achieved a 75 percent success rate and our loss values were quite low. Our validation rates were parallel to training and showed no signs of overfitting. These developments made us happy. In the next step, we tried to achieve the best result by making the necessary fine-tuning operations on this model. The most successful results were obtained with 94% accuracy on the SCVD dataset trained with VGG16 with 100 epochs and 92% accuracy on the Hockey Fight dataset trained with ResNet. Considering other studies, the model we have prepared has responded to us faster with the help of pre-trained models compared to the existing systems. After all, we were able to see the effect of the data we used on the result, and we were able to do various experiments with our Hockey Fight data set to see what works well with the model and what doesn't.

References

[1] P. Tirupattur, C. Schulze, and A. Dengel, *Violence detection in videos*, 2021. arXiv: 2109.08941 [cs.CV].

[2] S. Amraee, A. Vafaei, K. Jamshidi, and P. Adibi, "Abnormal event detection in crowded scenes using one-class svm," *Signal, Image and Video Processing*, vol. 12, Sep. 2018. DOI: 10.1007/s11760-018-1267-z.

[3] T. Hassner, Y. Itcher, and O. Kliper-Gross, "Violent flows: Real-time detection of violent crowd behavior," Jun. 2012, pp. 1–6, ISBN: 978-1-4673-1611-8. DOI: 10.1109/CVPRW.2012.6239348.

[4] S. Sudhakaran and O. Lanz, "Learning to detect violent videos using convolutional long short-term memory," in *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, 2017, pp. 1–6. DOI: 10.1109/AVSS.2017.8078468.

[5] J. Mahmoodi and A. Salajeghe, "A classification method based on optical flow for violence detection," *Expert Systems with Applications*, vol. 127, pp. 121–127, 2019, ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2019.02.032. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417419301460.

[6] G. Mu, H. Cao, and Q. Jin, "Violent scene detection using convolutional neural networks and deep audio features," vol. 663, Nov. 2016, pp. 451–463, ISBN: 978-981-10-3004-8. DOI: 10.1007/978-981-10-3005-5_37.

[7] "Violence detection techniques in video surveillance security systems: A systematic review. peerj computer science 8:e920." (), [Online]. Available: https://doi.org/10.7717/peerj-cs.920 (visited on 06/04/2022).

[8] "What is a confusion matrix in machine learning? the model evaluation tool explained." (), [Online]. Available: https://www.datacamp.com/tutorial/what-is-a-confusion-matrix-in-machine-learning.

[9] G. Anitha, M. Siddharth Shanker, M. Durgam Muni, L. Soma Sukrith, K. Venkatraman, and S. M. Usman, "Spatio temporal feature-based weapon and violence detection from video surveillance using deep learning," in *Proceedings of the 7th International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC 2023)*, IEEE, Chennai, Tamil Nadu, India, 2023, pp. 597–602, ISBN: 979-8-3503-4148-5. DOI: 10.1109/I-SMAC58438.2023.10290433.

[10] "How does batch size impact your model learning." (), [Online]. Available: https://medium.com/geekculture/how-does-batch-size-impact-your-model-learning-2dd34d9fb1fa.

[11] “Video augmentation techniques for deep learning.” (), [Online]. Available: `https://github.com/okankop/vidaug`.

Curriculum Vitae

FIRST MEMBER

Name-Surname: Esma Nur Ekmekci
Birthdate and Place of Birth: 15.08.2001, Konya
E-mail: nur.ekmekci@std.yildiz.edu.tr
Phone: 05061783465
Practical Training: Akbank Data Office Department
Çimsa Software Department
Ibtech Mobil Banking Backend

SECOND MEMBER

Name-Surname: Muhammad Nasir Sabır
Birthdate and Place of Birth: 20.08.1999, Mezar-ı Şerif
E-mail: nasir.sabir@std.yildiz.edu.tr
Phone: 0541 744 91 49
Practical Training: Qlub, Site Relaibility Engineering

Project System Informations

System and Software: MAC0S, Python
Required RAM: 16GB
Required Disk: 256MB