Violence Detection Overview

Convolutional Neural Networks (CNN) are great for image data and Long-Short Term Memory (LSTM) networks are great when working with sequence data but when you combine both of them, you get the best of both worlds, and you solve difficult computer vision problems like video classification.

Our approach involves a Convolutional Neural Network Bidirectional LSTM model (CNN-Bi-LSTM) architecture to predict violence in the sequential flow of frames. Firstly, we breakdown a video into several frames. We pass each frame through a convolutional neural network, to extract the information present in that current frame. Then we use a Bidirectional LSTM layer to compare the information of the current frame once with the previous frames and once with the upcoming frames to identify any sequential flow of events. Finally, the classifier is used to identify whether an action is violent or not.

Understanding the Dataset

This dataset contains ONLY two directories: **Non-Violence** (which contains 1000 real life situations videos like eating, sports activity, singing, etc. and this directory doesn't have any violence situations) and the other directory **Violence** (contains 1000 videos with severe violence in various situations).

Real Life Violence Situations Dataset | Kaggle

Preprocessing

All what we need to do here is to extract the frames from all the videos.

We created a function frames_extraction() that will create a list containing the resized and normalized frames of a video whose path is passed to it as an argument. The function will read the video file frame by frame, although not all frames are added to the list as we will only need an evenly distributed sequence length of frames.

```
def frames_extraction(video_path):
    '''
    This function will extract the required frames from a video after resizing
and normalizing them.
    Args:
        video_path: The path of the video in the disk, whose frames are to be
extracted.
    Returns:
        frames_list: A list containing the resized and normalized frames of the
video.
    '''

    # Declare a list to store video frames.
    frames_list = []

    # Read the Video File using the VideoCapture object.
    video_reader = cv2.VideoCapture(video_path)

    # Get the total number of frames in the video.
    video_frames_count = int(video_reader.get(cv2.CAP_PROP_FRAME_COUNT))

    # Calculate the the interval after which frames will be added to the list.
    skip_frames_window = max(int(video_frames_count/SEQUENCE_LENGTH), 1)

    # Iterate through the Video Frames.
    for frame_counter in range(SEQUENCE_LENGTH):

        # Set the current frame position of the video.
        video_reader.set(cv2.CAP_PROP_POS_FRAMES, frame_counter *
skip_frames_window)

        # Reading the frame from the video.
        success, frame = video_reader.read()

        # Check if Video frame is not successfully read then break the loop
        if not success:
            break

        # Resize the Frame to fixed height and width.
        resized_frame = cv2.resize(frame, (IMAGE_HEIGHT, IMAGE_WIDTH))
```

```
# Normalize the resized frame by dividing it with 255 so that each pixel
value then lies between 0 and 1
        normalized_frame = resized_frame / 255

        # Append the normalized frame into the frames list
        frames_list.append(normalized_frame)

    # Release the VideoCapture object.
    video_reader.release()

    # Return the frames list.
    return frames_list
```

Then We created a function create_dataset() that will iterate through all the classes specified in the CLASSES_LIST constant and will call the function frame_extraction() on every video file of the selected classes and return the frames (features), class index (labels), and video file path (video_files_paths)

Model Building

Architectures

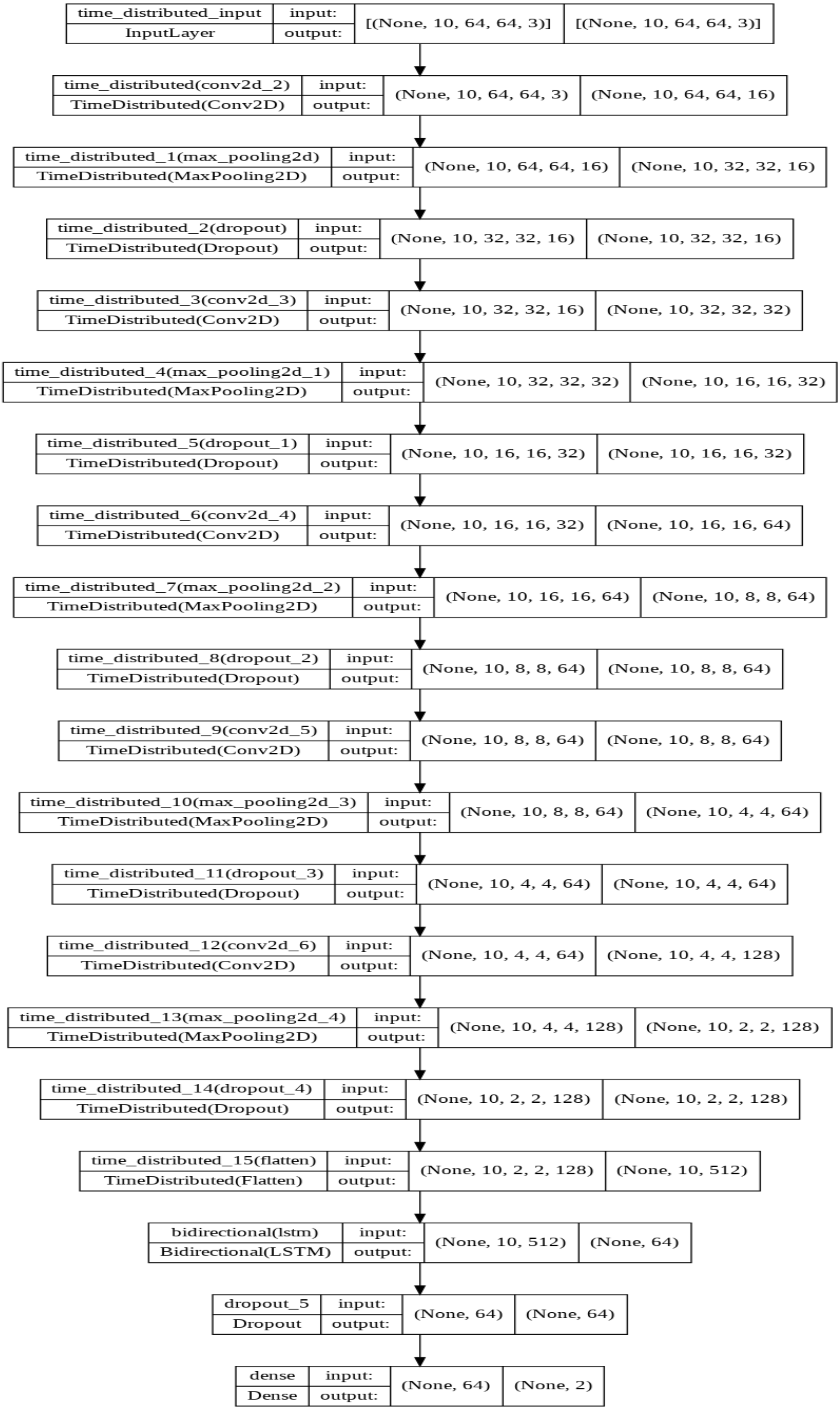
Here we built many models with different Architectures as we were searching for the best model.

- **VGG-16 + Bidirectional-LSTM** accuracy: 84% - 86%
- **Convnet + Bidirectional-LSTM** accuracy: 90% - 92%
- **Mobile Net V2 + Bidirectional-LSTM** accuracy: 94% - 97%

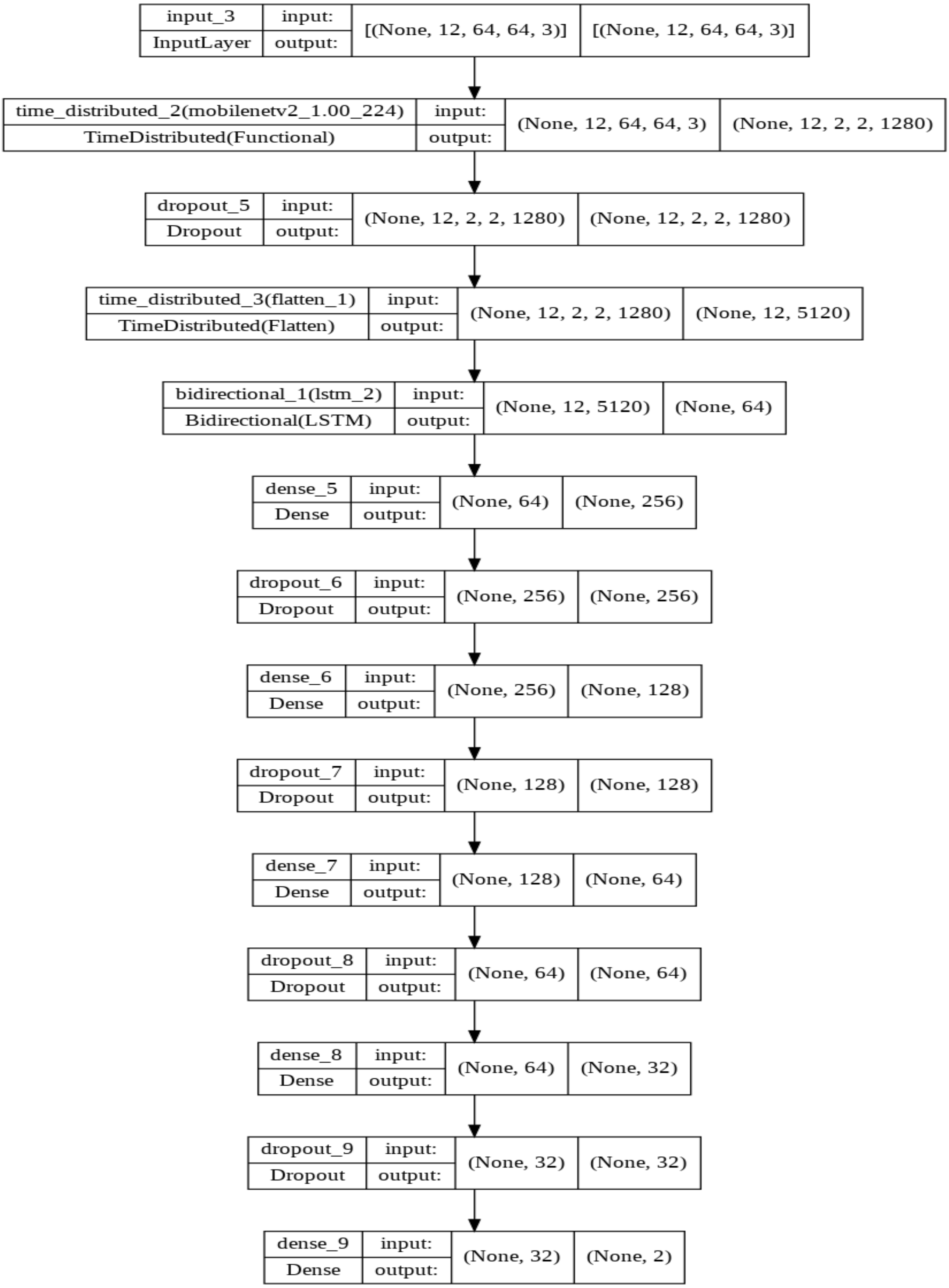
Our First Architecture: Convnet + Bidirectional-LSTM

This Architecture give a reasonable accuracy with least amount of parameters

It was a very efficient model, but we were looking for more model performance.

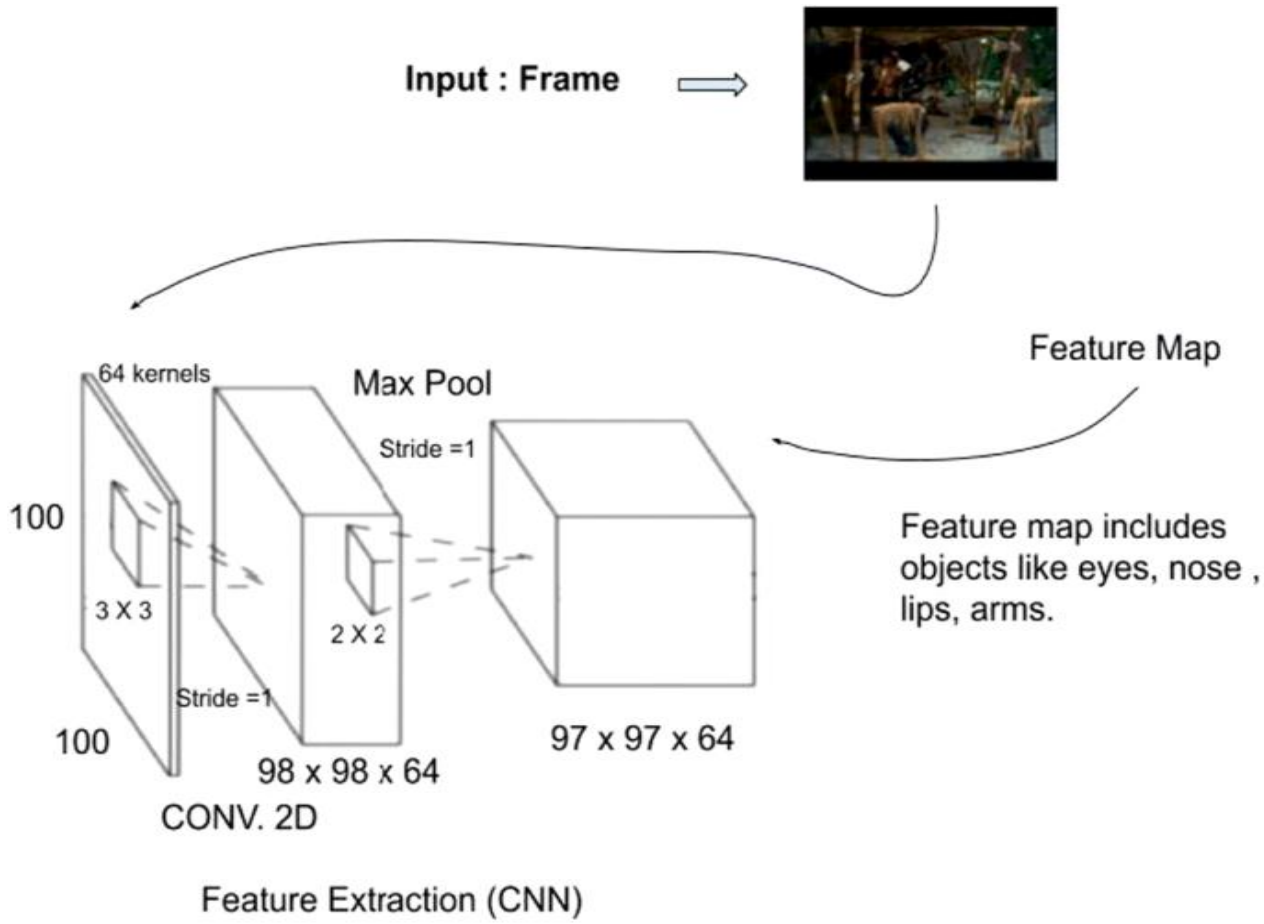

time_distributed_input
InputLayer
input:
output:
[(None, 10, 64, 64, 3)]
[(None, 10, 64, 64, 3)]
time_distributed(conv2d_2)
TimeDistributed(Conv2D)
input:
output:
(None, 10, 64, 64, 3)
(None, 10, 64, 64, 16)
time_distributed_1(max_pooling2d)
TimeDistributed(MaxPooling2D)
input:
output:
(None, 10, 64, 64, 16)
(None, 10, 32, 32, 16)
time_distributed_2(dropout)
TimeDistributed(Dropout)
input:
output:
(None, 10, 32, 32, 16)
(None, 10, 32, 32, 16)
time_distributed_3(conv2d_3)
TimeDistributed(Conv2D)
input:
output:
(None, 10, 32, 32, 16)
(None, 10, 32, 32, 32)
time_distributed_4(max_pooling2d_1)
TimeDistributed(MaxPooling2D)
input:
output:
(None, 10, 32, 32, 32)
(None, 10, 16, 16, 32)
time_distributed_5(dropout_1)
TimeDistributed(Dropout)
input:
output:
(None, 10, 16, 16, 32)
(None, 10, 16, 16, 32)
time_distributed_6(conv2d_4)
TimeDistributed(Conv2D)
input:
output:
(None, 10, 16, 16, 32)
(None, 10, 16, 16, 64)
time_distributed_7(max_pooling2d_2)
TimeDistributed(MaxPooling2D)
input:
output:
(None, 10, 16, 16, 64)
(None, 10, 8, 8, 64)
time_distributed_8(dropout_2)
TimeDistributed(Dropout)
input:
output:
(None, 10, 8, 8, 64)
(None, 10, 8, 8, 64)
time_distributed_9(conv2d_5)
TimeDistributed(Conv2D)
input:
output:
(None, 10, 8, 8, 64)
(None, 10, 8, 8, 64)
time_distributed_10(max_pooling2d_3)
TimeDistributed(MaxPooling2D)
input:
output:
(None, 10, 8, 8, 64)
(None, 10, 4, 4, 64)
time_distributed_11(dropout_3)
TimeDistributed(Dropout)
input:
output:
(None, 10, 4, 4, 64)
(None, 10, 4, 4, 64)
time_distributed_12(conv2d_6)
TimeDistributed(Conv2D)
input:
output:
(None, 10, 4, 4, 64)
(None, 10, 4, 4, 128)
time_distributed_13(max_pooling2d_4)
TimeDistributed(MaxPooling2D)
input:
output:
(None, 10, 4, 4, 128)
(None, 10, 2, 2, 128)
time_distributed_14(dropout_4)
TimeDistributed(Dropout)
input:
output:
(None, 10, 2, 2, 128)
(None, 10, 2, 2, 128)
time_distributed_15(flatten)
TimeDistributed(Flatten)
input:
output:
(None, 10, 2, 2, 128)
(None, 10, 512)
bidirectional(lstm)
Bidirectional(LSTM)
input:
output:
(None, 10, 512)
(None, 64)
dropout_5
Dropout
input:
output:
(None, 64)
(None, 64)
dense
Dense
input:
output:
(None, 64)
(None, 2)

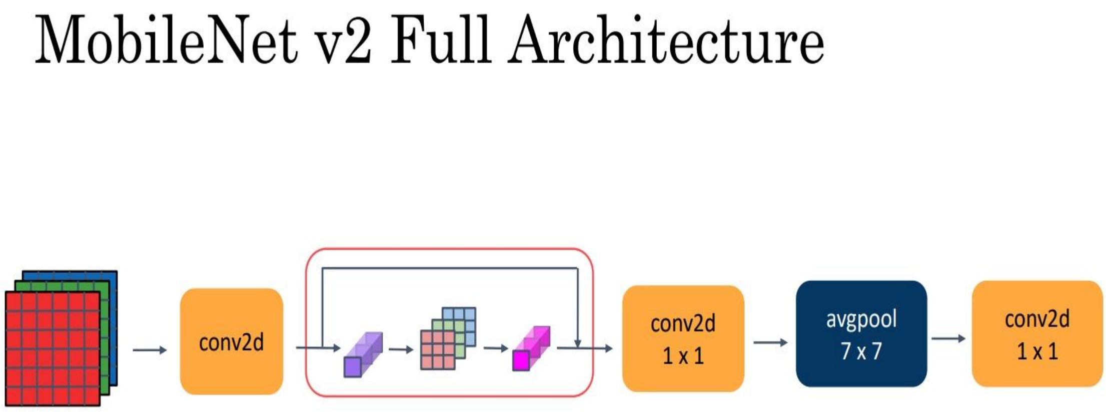
Our Final Architecture



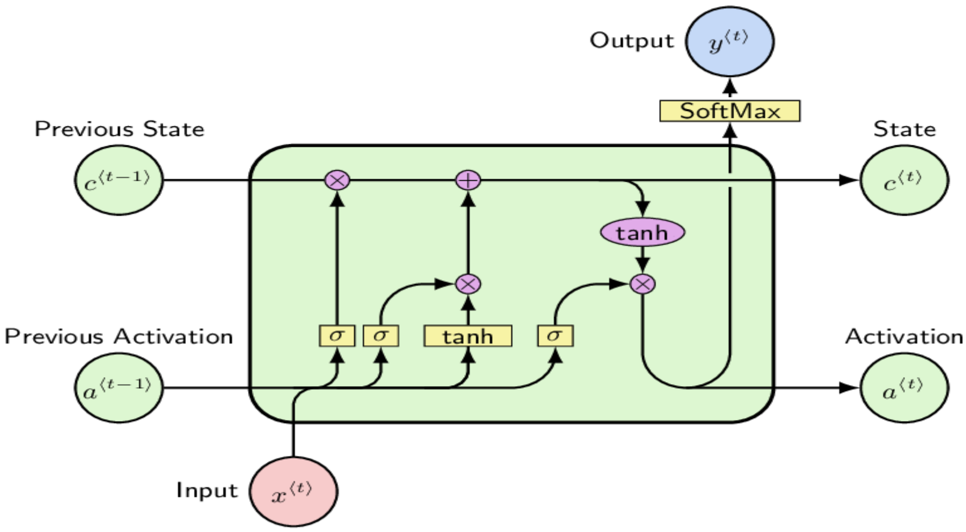
Methodology

1) MobileNetV2 : Feature Extraction from the frames

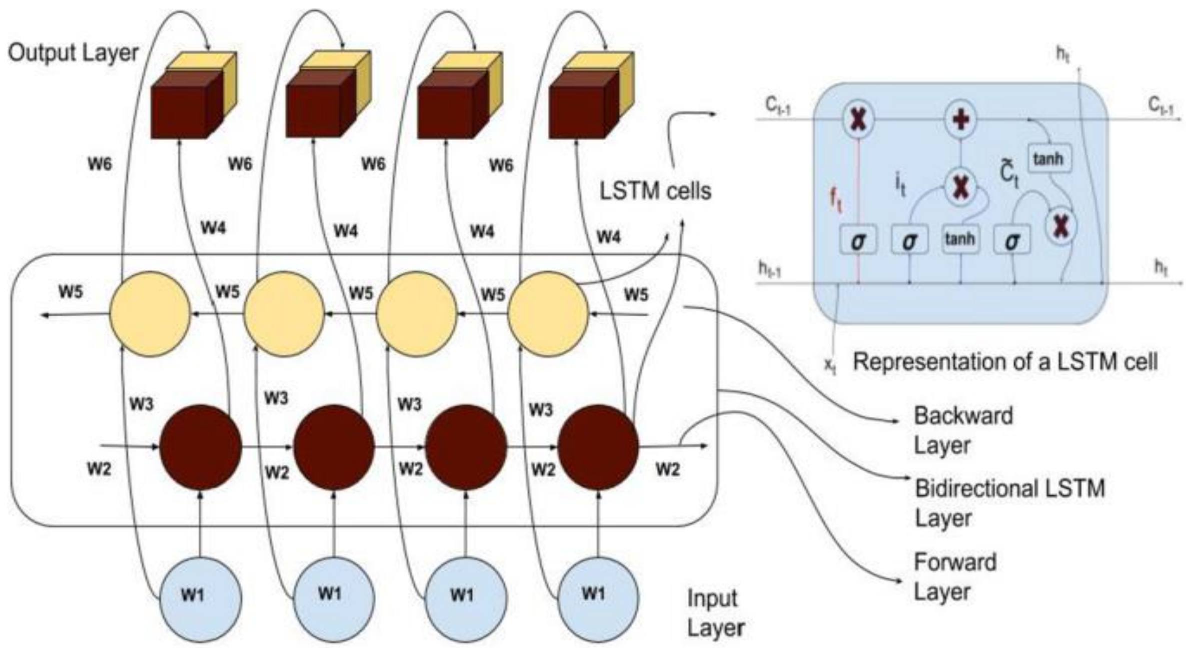



MobileNet v2 Full Architecture
conv2d
conv2d
1 x 1
avgpool
7 x 7
conv2d
1 x 1

2) LSTM-Cell

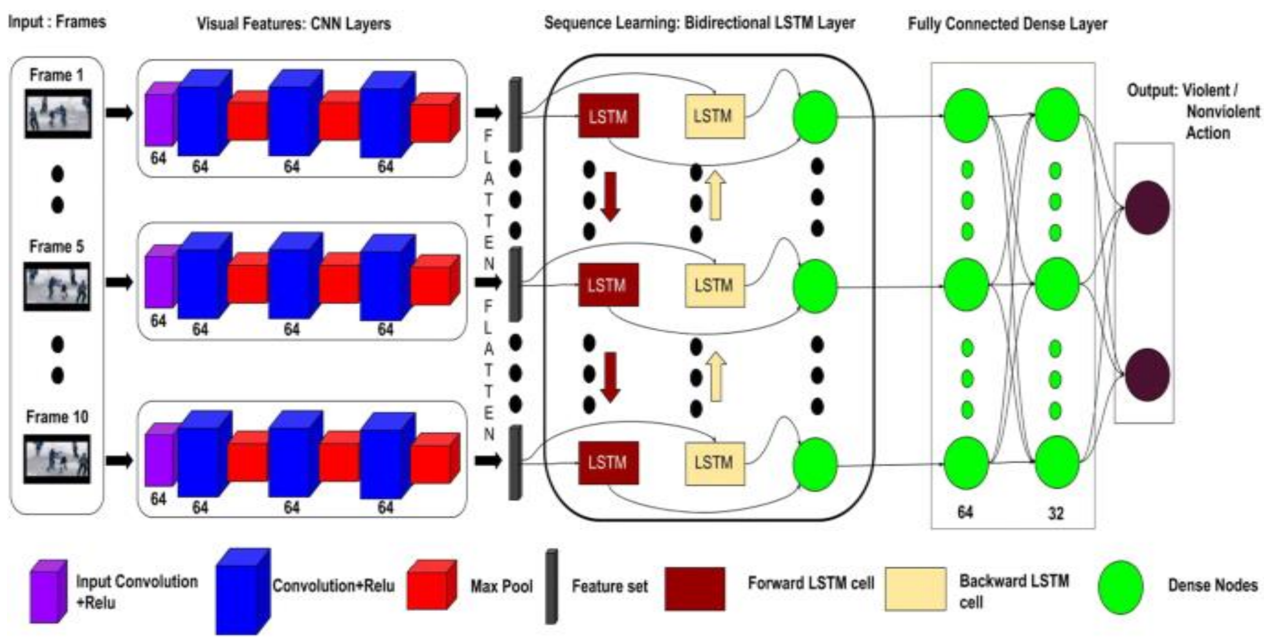

Output
$y^{\langle t \rangle}$
SoftMax
Previous State
$c^{\langle t-1 \rangle}$
State
$c^{\langle t \rangle}$
tanh
σ
σ
tanh
σ
Previous Activation
$a^{\langle t-1 \rangle}$
Activation
$a^{\langle t \rangle}$
Input
$x^{\langle t \rangle}$

3) Bi-LSTM Layer

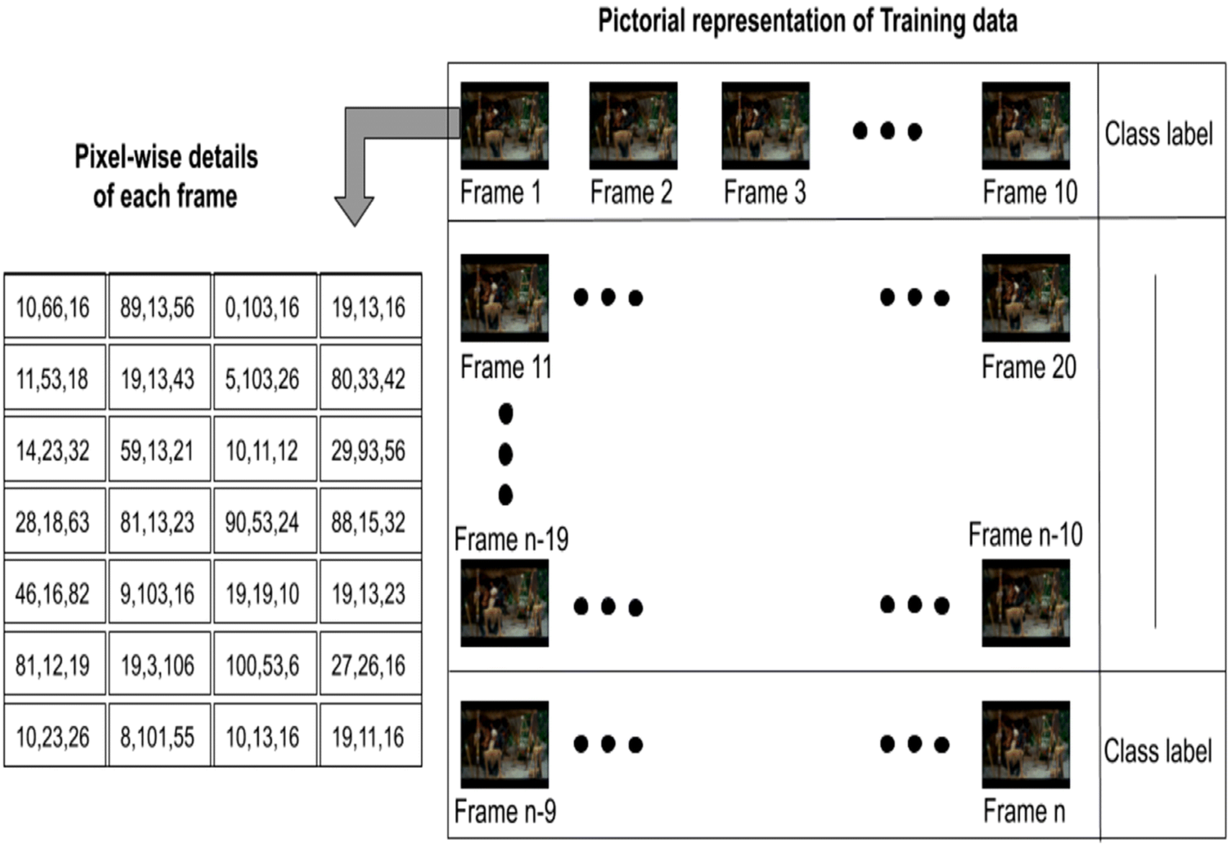


4) Dense Layers specified in the entire architecture

The Entire Architecture



Representation of Training data



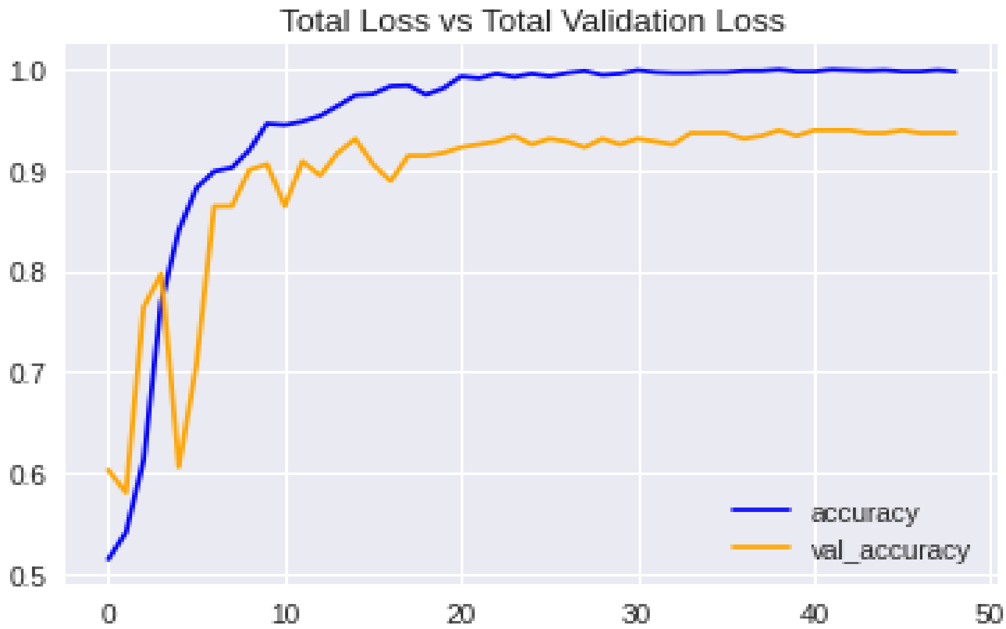
Metrics considered for Model Evaluation

Accuracy, Precision, Recall and F1 Score

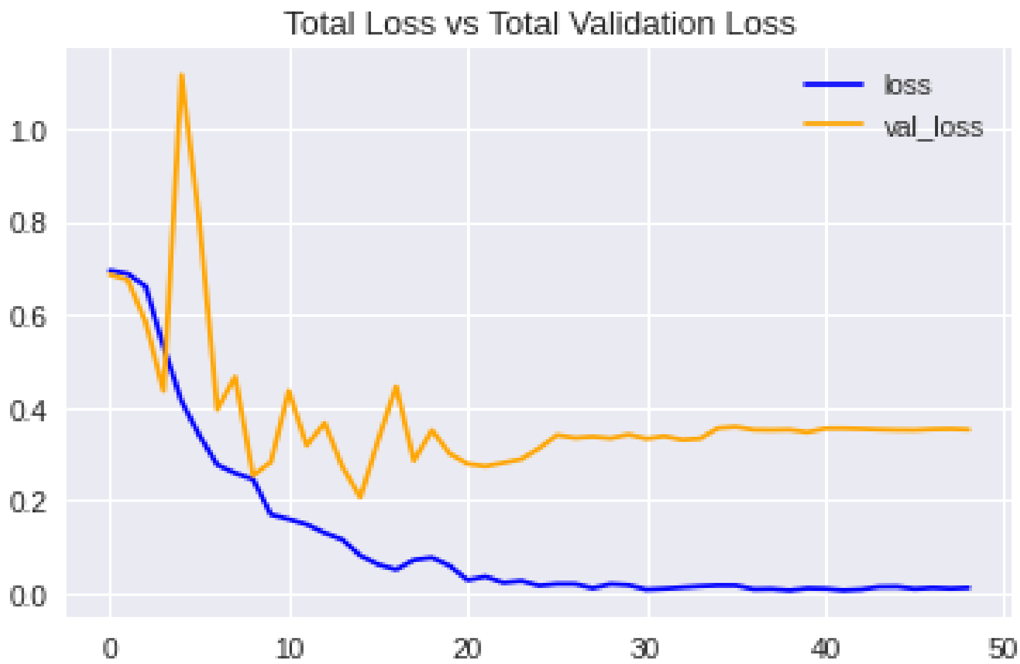
- Accuracy: What proportion of actual positives and negatives is correctly classified?
- Precision: What proportion of predicted positives are truly positive?
- Recall: What proportion of actual positives is correctly classified?

F1-Score: Harmonic mean of Precision and Recall

Training Accuracy (blue) vs Validation Accuracy (yellow)



Training Loss vs Validation Loss

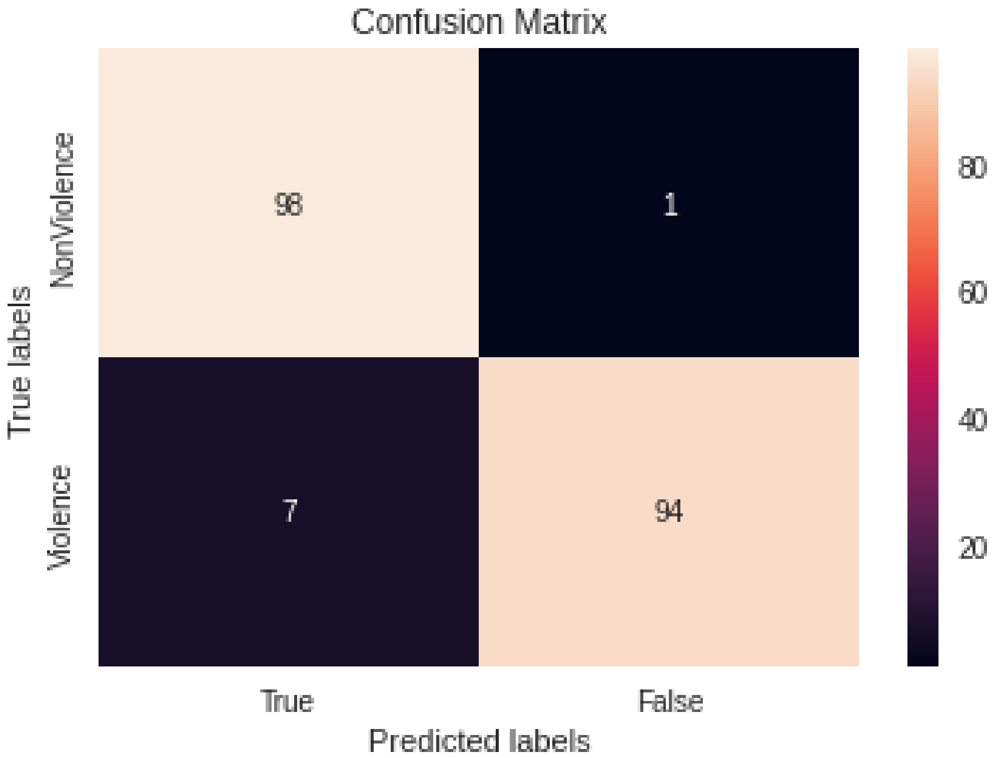


Confusion Matrix

For prediction:

True → NonViolence (1)

False → Violence (0)



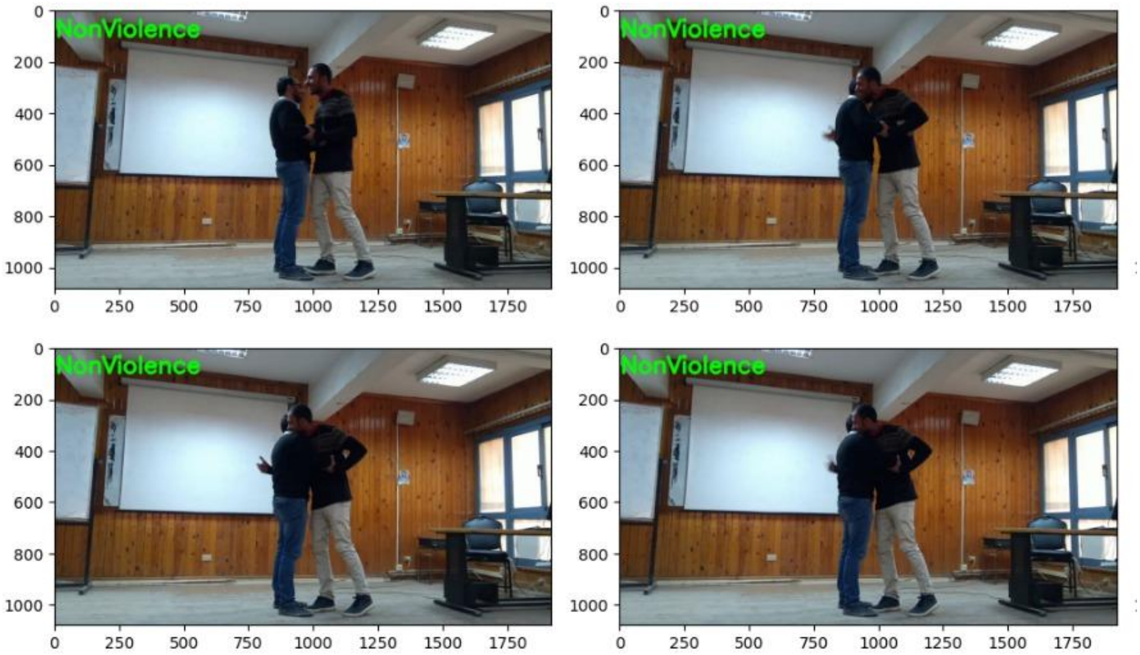
Classification Report

```
Classification Report is :
              precision    recall  f1-score   support

           0       0.93      0.99      0.96        99
           1       0.99      0.93      0.96       101

    accuracy                           0.96       200
   macro avg       0.96      0.96      0.96       200
weighted avg       0.96      0.96      0.96       200
```

Experiments

We Implemented two Prediction Functions :

- predict_frames : this one takes the video and predict each frame and then it generates a new video with the prediction output printed on the upper left of each frame then it saves the video.



- predict_video : this function takes the video and predict the action of the whole video.

```
Predicted: NonViolence
Confidence: 0.9999170303344727
```

```
Predicted: Violence
Confidence: 0.9999997615814209
```

References

[Convolutional Neural Networks | Coursera] Offered by DeepLearning.AI

Sandler et al. 2019, MobileNetV2: Inverted Residuals and Linear Bottlenecks

[CNN-BiLSTM Model for Violence Detection in Smart Surveillance | SpringerLink]