Breaking Bias Season – 2: Final Project Report

Dabeet, Sarthak, Kavin

25th March 2025

1 The Task

Given a dataset of peace (routine, non-violent activities) and brawl (physical altercations or fights) videos, the task was to develop a machine learning model to classify the videos. There were no criteria on the evaluation metrics, and a bonus task to determine the number of people involved in the brawl was also given. The dataset was labeled, making this a supervised learning problem.

2 The Data

The dataset consisted of 1,600 surveillance videos, classified into two categories:

2.1 Peace Videos

- Number of videos: 800
- Description: These footages feature individuals or groups engaging in routine actions such as walking or talking. Movements in these videos remain calm, with minimal sudden changes.

2.2 Brawl Videos

- Number of videos: 800
- Description: These videos capture instances of physical altercations or brawls in public spaces. The footage often shows rapid movement, chaotic patterns, and intense interactions.

3 A 3D ConvNet that Captures Spatio-Temporal Features

This approach is inspired by the 2012 research paper [1].

3.1 2D ConvNets and Image Classification

A 2D Convolutional Neural Network (CNN) is a deep learning architecture designed for processing grid-like data such as images. It applies 2D convolutional filters to extract spatial features like edges, textures, and patterns, making it highly effective for image-related tasks.

The following are the layers of a 2D ConvNet:

- **Convolutional Layers** – Apply filters (kernels) to learn spatial hierarchies of features.
- **Activation Functions** – Introduce non-linearity for better learning.
- **Pooling Layers** – Reduce spatial dimensions while retaining essential information.
- **Fully Connected Layers** – Flatten feature maps and classify images.
- **SoftMax Layer** – Converts outputs into probability distributions for classification.

3.2 3D ConvNets and Video Classification

A 3D Convolutional Neural Network (3D CNN) is an extension of 2D CNNs that processes video data by applying convolutional filters across spatial and temporal dimensions. Unlike 2D CNNs, which operate on single images, 3D CNNs capture motion by considering multiple consecutive frames as input.

A 3D CNN extracts spatial and temporal features, processes them through multiple convolutional and pooling layers, and classifies the video using fully connected layers. This makes 3D CNNs effective for action recognition, anomaly detection, and video-based surveillance.

4 The Model

Preprocessing involved loading, cleaning, and normalizing data to train the model. The dataset was loaded from Google Drive in Google Colab. Videos were split into 64×48 frames, and 13 consecutive frames [1] were fed into the first convolutional layer. Labels were defined as 0 (Brawl) and 1 (Peace). No grayscale conversion was done since TensorFlow handles RGB inputs.

A train-validation split of 80-20 was maintained. The 3D CNN model consisted of:

- **First Convolutional Layer**: 7 filters with a $(7, 7, 7)$ kernel size, followed by a MaxPooling3D layer.
- **Second Convolutional Layer**: 50 filters with a $(7, 7, 5)$ kernel size, followed by another MaxPooling3D layer.
- **Third Convolutional Layer**: 10 filters with a $(5, 5, 3)$ kernel size.

- **Fully Connected Layer**: Flattened feature map passed through a dense layer with 100 neurons, ReLU activation, and Dropout (0.55 probability) to prevent overfitting.
- **Output Layer**: SoftMax-activated dense layer for classification.

The model was trained using the Adam optimizer. Since the data was one-hot encoded, categorical cross-entropy was used as the loss function. A learning rate of 0.000095 was maintained throughout training. The model was trained for 18 epochs [1]. Validation loss fluctuated after approximately 8 epochs, indicating slight overfitting, but the validation accuracy steadily increased to about 80% by the end of training.

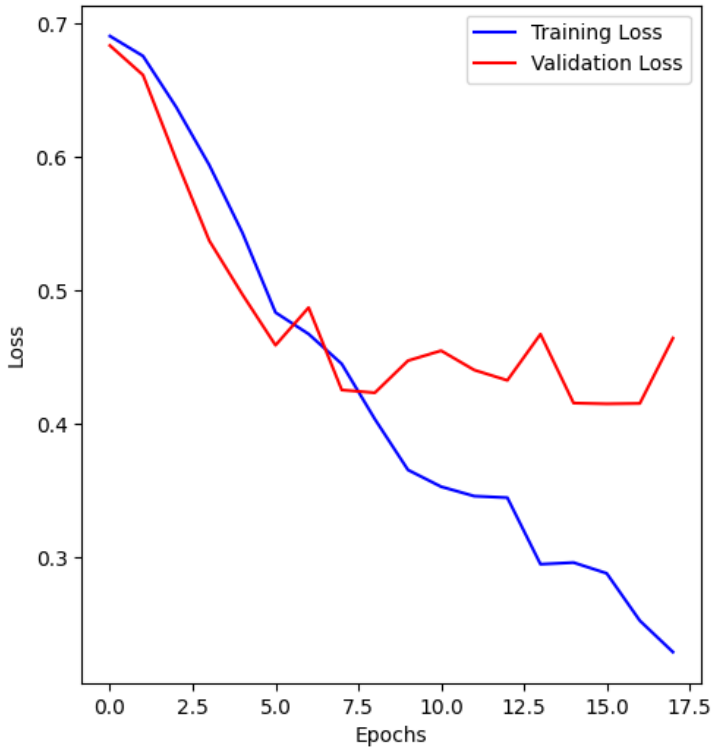

Figure 1: Training and Validation Loss

5 Evaluation

The confusion matrix was obtained as follows:

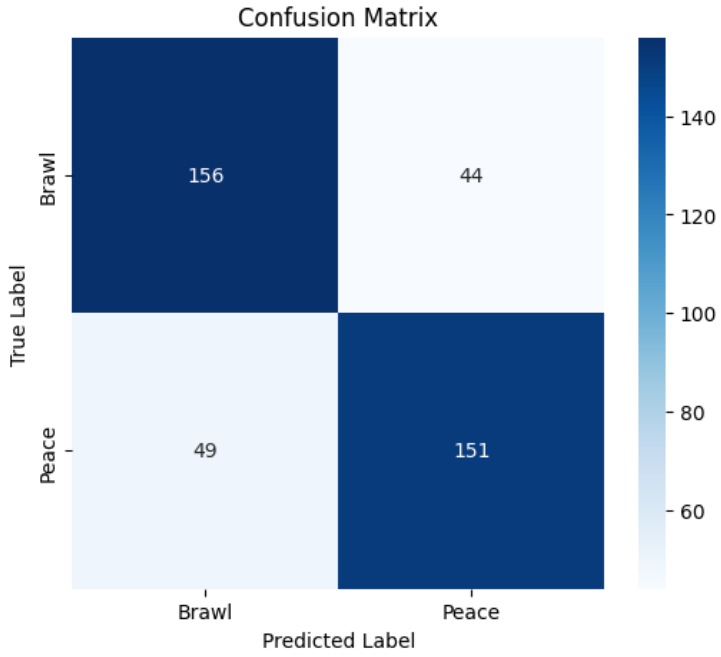

Figure 2: Confusion Matrix

Metric	Value
F1 Score	0.77037
Recall	0.76098
Precision	0.78000
Accuracy	0.76750

Table 1: Evaluation Metrics

6 Implementation and Running the Code

The entire code for the CNN was written in python using various libraries such as Tensorflow, matplotlib, seaborn, etc (the entire listing is done in the txt file). The comprehensive code has been uploaded with README file to github and serves as the final submission for the task. This repository consists of the following items. Please ensure that you follow the exact steps mentioned here to ensure seamless access to the program. In the text that follows, I have mentioned two ways to run the python scripts, on google colab as well as locally on your PC. Please be cautioned that Tensorflow 2.1+ is no longer supported for Windows, and this code will be able to run only on a Linux/Mac machine or a virtual machine / dual boot if you are using Windows.

```
project-root/
|-- models/
|   |-- breaking-bias-2012-balanced.h5
|   |-- model2-weights.h5
|-- breaking_bias_2_2012_research_paper.ipynb
|-- breaking_bias_2_2014_2d_paper.ipynb
|-- README.pdf
|-- requirements.txt
```

Running the Code Locally

- First up make sure that you have all the libraries mentioned in requirements.txt installed on your computer.
- Clone the repository to your PC and open it in jupyter notebook.
- The paths to the folders in the code have been left empty for you to fill according to the directory listing on your computer.
- Once done, run all cells on jupyter notebook and you are good to go! (Caveat: Use a GPU for best performance!)
- Alternatively, you may decide to use the models directly with the weights and biases. For such cases, I have provided the h5 files to both the models in "models/" directory. One can follow the standard tensorflow documentation to load these models and run them.

Running the Code on Google Colab

- You can directly clone the repository to your google colab environment, use !git clone https://github.com/DabeetDas/video-classification.git and then `%cd video-classification`.
- Fill the paths to the training and testing data directories.
- Change your runtime type from CPU to T4-GPU for better performance.(as above)
- Run all cells.

Data for the CNN

- Training Data consisting of 1,600 videos.
- Testing Data evaluate the model on this data.

7 Another Approach To The Task

This approach was inspired by the 2014 research paper [2].
The second approach implements a 2D Convolutional Neural Network (CNN) for video classification, distinguishing between "Brawl" and "Peace" scenarios using middle frames extracted from video clips. The dataset is stored in Google Drive, and each video is processed to extract a single representative frame, resized to (170,170), and normalized for input into the CNN. The dataset is then split into training and testing sets using an 80-20% ratio. This preprocessing approach significantly reduces computational costs by focusing on a single frame rather than full video sequences.

The model architecture is built using TensorFlow's Keras API, comprising multiple Conv2D layers for feature extraction, MaxPooling layers for downsampling, and Dense layers for classification. The network employs ReLU activation for non-linearity and a SoftMax layer to classify inputs into one of the two categories. The Adam optimizer is used with categorical cross-entropy loss to train the model effectively. Dropout regularization is applied to prevent overfitting, ensuring better generalization on unseen data.

The results show steady improvement in validation accuracy, reaching approximately 80% by the end of training. However, some overfitting is observed as validation loss fluctuates after a few epochs. The final confusion matrix reveals a well-balanced classification performance, with both classes exhibiting relatively low misclassification rates.The F1 score of 0.77037 suggests a strong balance between precision (0.78) and recall (0.76098), confirming the model's ability to correctly classify both categories without excessive false positives or false negatives. The final accuracy stands at 0.76.

References

[1] Deepak Pathak, Kaustubh Tapi, *Human Action Classification Using 3D Convolutional Neural Netowrk*, 2012.

[2] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, Li Fei-Fei, *Large-scale Video Classification with Convolutional Neural Networks*, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014, pp. 1725-1732,2014.