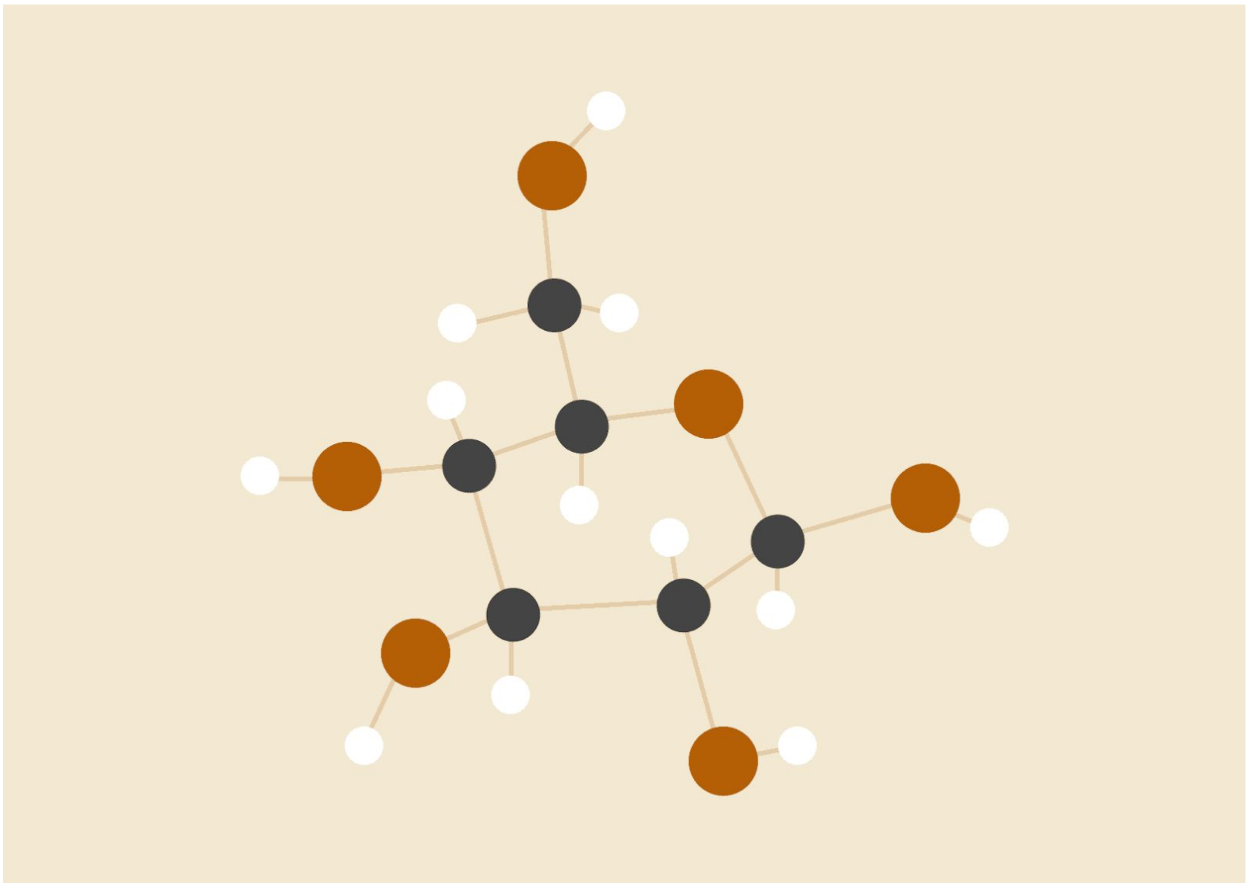
Violence Detection in Video Classification Individual Report

Utilizing the RWF-2000 Video Database



Spencer Staub

12/07/2020

Machine Learning II

Amir Jafari

1. Abstract

Since I wrote the final report for this assignment I will be utilizing the text in parts of this report. Below you will find text from the final report; the introduction, database description, preprocessing steps and three models that I implemented: Conv2D, ResNet50 and Time Distributed Conv2D with LSTM. I will also be utilizing parts of the final report conclusion in this individual report.

I found working with the other individuals in this group quite difficult as there was a serious language barrier and communication was at a minimum. Given this collaboration was quite difficult, resulting in this project more acting like a Kaggle competition with the best model winning in the end.

2. Introduction

With the increase in surveillance cameras throughout the world, a large amount of video data can now be collected in efforts to classify human action. Like many other sources of data there has become an excess of observations and a lack of human capability to observe them. For this project I will attempt to utilize the RWF-2000 video database which includes 2000 videos split evenly between violent and nonviolent actions in order to create a binary classification network to detect violence. Hopefully, this classification system will be quick and accurate enough to alert authorities in time to mitigate the damages caused by the violence captured.

3. Database

The RWF-2000 dataset comes from Ming Cheng, Kenjing Cai, and Ming Li from Duke Kunshan University located in the Jiangsu Province, China. The dataset includes 2000 video surveillance clips collected from youtube sliced into 5 second clips at 30 frames per second. The database is split evenly between violence and non-violence clips, as Ill as split 80/20 between test and train respectively. The video was downloaded in a ~11gb zip file and unzipped resulted in about 100gb worth of video files in .avi format.

3.1. Video Quality and Description

Since the videos are captured by surveillance cameras, many of them are poor quality due to dark environments, low resolution, low frames per second, quick moving objects, non-color video, lighting, blur and image obstruction.

Videos are also in a variety of different formats including large crowds, one-on-one interactions, full frame conflict, conflict that occurs partially out of frame or far away and close contact events where it could be hard to distinguish two humans from each other.

4. Preprocessing

Preprocessing was performed separately for each model, hoIver, there are a few steps taken to establish a directory of videos converted to numpy in tensor format with shape = [nb_frames, img_height, img_width, 5]. Preprocessing began with the cv2 package deriving 3 RGB flows and 2 Optical Flows. Optical flow is defined as the apparent motion of individual pixels on the image plane. In other words the motion of images and objects between frames of a video. This will assist us in deriving human action between frames. After creating the 5 total flows for each video I can resize each video to (224,224) and (64,64). Each resize is a separate directory and will be used by different models depending on the memory and computing poIr needed. Files are finally saved to a directory formatted like the unzipped RWF-2000 videos and can be accessed the same way.

Since my dataset is extremely large I will have to randomly select a smaller number of videos in order to avoid memory errors instead of 1600 train and 400 test videos split evenly between violence and non violence most of my models use a 64/8, 128/32 or 256/64 depending on preprocessing steps and complexity of the model.

Labels will also have to be established for each video, however, that is performed per each network using a custom binary label encoding. For each video in my directory a video will be classified as 1 or 0 depending on if it is in the violence or non-violence subdirectory. Once labels are created I can create x and y numpy arrays for both the train and test sets.

Finally, some of my models normalize the data using a simple (data - mean / standard deviation) function. HoIver, due to the size of my data when using normalization I must decrease the number of videos sampled further. In some cases normalization does not provide enough accuracy increase to justify limiting my sample size.

5. Networks

A variety of networks are attempted from a basic Multi Layer Perceptron or pretrained networks to more advanced Long Short-Term Memory networks. Below are each network summarized with results included.

5.1. Conv2D

5.1.1. Setup

The Conv2D Network starts with 256 training examples and 64 test examples. I am unable to utilize all 800 and 200 train/test files due to memory issues. The network uses all 5 channels (3 RGB and 2 Optical) with an input shape of (64,64,5).

5.1.2. Network

It is a 2 layer Conv2D network with 32 and 64 neurons respectively and a (3,3) kernel size with activation relu. After each Conv2D steps there is a MaxPooling2D. Following the Conv2D steps I flatten, add a dense layer with 256 neurons with relu activation and a final dense layer with 2 neurons and a sigmoid activation to perform binary classification.

5.1.3. Compile and Fit

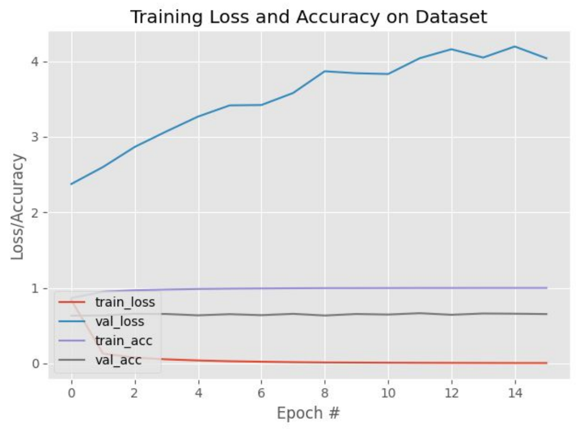
The network is compiled with Binary Crossentropy, Stochastic Gradient Descent, and accuracy metrics. Batch size is 16 with 32 epochs and callbacks with early stopping and a model checkpoint using val_loss.

5.1.4. Results

Since this is a balanced dataset accuracy will be my best metric to use, but I will also utilize F1-Score, Cohen Kappa and a confusion matrix to measure the performance of each network.

Accuracy: 65.27% F1-Score: 0.65 Cohen Kappa: 0.30

Loss/Accuracy Graph:



5.2. Resnet50

5.2.1. Setup

The Resnet Network starts with 256 training examples and 64 test examples. I am unable to utilize all 800 and 200 train/test files due to memory issues. The network uses only 3 channels (3 RGB) with an input shape of (149, 64*64,3).

5.2.2. Network

ResNet-50 is a convolutional neural network that is 50 layers deep with over a million images trained from the ImageNet database. The network has an image input size of (224,224), but my input of (64,64) works Ill too. my Network is finished off with an average pooling with shape (2,2) a dense function with 512 neurons with relu classification, 0.5 dropout and finally a dense layer with 2 neurons and sigmoid activation for binary classification

ResNet50 Architecture

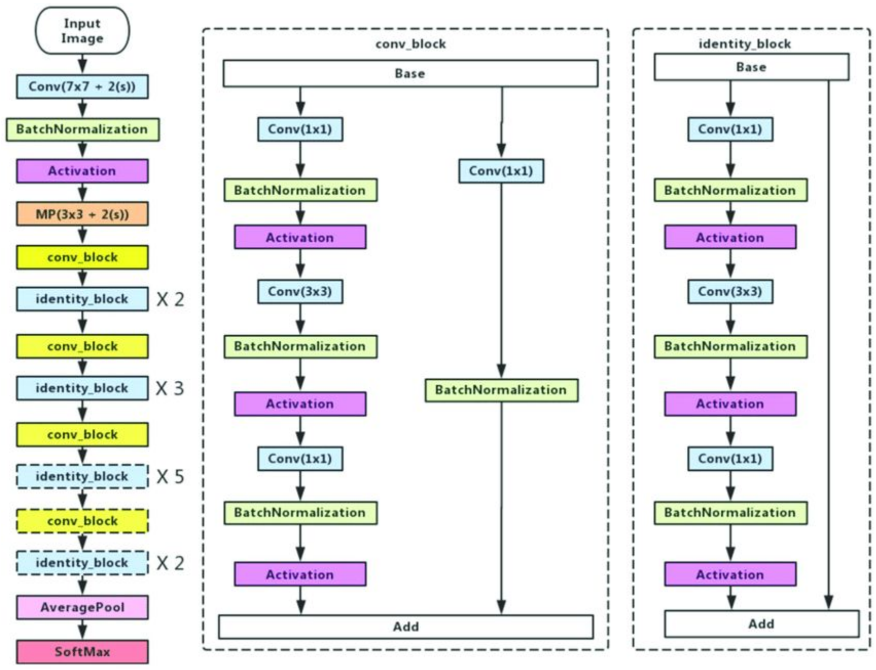


Figure 1

5.2.3. Compile and Fit

The network is compiled with Binary Crossentropy, Stochastic Gradient Descent, and accuracy metrics. Epoch size is 30, steps per epoch is 32, validation steps is 32 and callbacks with early stopping and a model checkpoint using val_loss.

5.2.4. Results

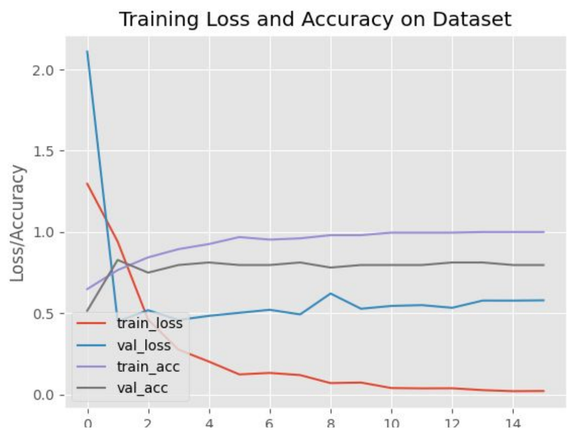
Since this is a balanced dataset accuracy will be my best metric to use, but I will also utilize F1-Score, Cohen Kappa and a confusion matrix to measure the performance of each network.

Accuracy: 84.375% F1-Score: 0.84 Cohen Kappa: 0.6875

Confusion Matrix:

n = 64	Positive	Negative
Positive	26	6
Negative	7	25

Loss/Accuracy Graph:



5.3. Time Distributed CONV2D + LSTM

5.3.1. Setup

The Time Distributed Conv2D + LSTM Network starts with 256 training examples and 64 test examples. I am unable to utilize all 800 and 200 train/test files due to memory issues. The network uses 5 channels (3 RGB and 2 Optical Flows) with an input shape of (149, 64, 64,5).

5.3.2. Network

This network begins with an input layer of (batch_size, 149, 64, 64,) and then proceeds to its first TimeDirstributed Conv2D layer. I utilize the

TimeDistributed to evaluate multiple frames of a video and make a classification from there. After the Conv2D layer I apply a max pooling with (2,2) pool size. Flatten, and finally add a Dense layer with 128 neurons and relu activation all within the TimeDistributed network. From there an LSTM layer is applied along with another Flatten and Dense layer with sigmoid activation.

5.3.3. Compile and Fit

The network is compiled with Binary Crossentropy, Stochastic Gradient Descent, and accuracy metrics. Epoch size is 30, batch size is 64, steps per epoch is 32, validation steps is 32 and callbacks with early stopping and a model checkpoint using val_loss.

5.3.4. Results

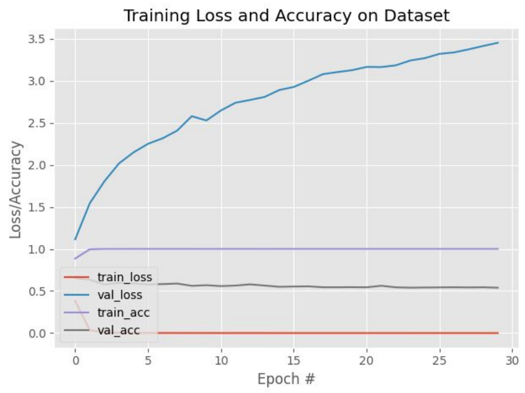
Since this is a balanced dataset accuracy will be my best metric to use, the Accuracy, BCE Loss and Confusion Matrix will be applied to test the result.

Accuracy: 81.5% F1-Score: 0.74 Cohen Kappa: 0.5

Confusion Matrix:

n = 16	Positive	Negative
Positive	7	1
Negative	3	5

Loss/Accuracy Graph:



Both accuracies in the above model show to increase over time. Significantly for the training set, but only slightly for the validation set. Both losses also decrease for each epoch with slight variations positive and negative.

6. Model Results and Evaluation

Since the RWF-2000 database was evenly distributed between violence and nonviolence I am able to use Validation Accuracy as my primary metric for evaluation. After that I utilized Cohen Kappa, F1 and BCE loss to evaluate my models further. A basic model.fit was applied to most of my networks to train.

Initially I began with 3-dimensional Conv2D networks. Each network had an input of (frames, height*width, channels), frames being the added dimension as the networks will be iterating over each video. Channels are 3 RGB and 2 Optical flows. I utilized one pre-trained network in this experiment; ResNet50. All of this network is based off of the Imagenet database and performed Ill at 84% validation accuracy. These results are also expected to be better than my MLP and CNN 2-dimension networks since rather than evaluating every frame the 3-dimensional networks will evaluate each video and within each video every frame as they relate to each other. In this case the networks are able to piece all of the information of a single video together for classification.

Finally, I attempted a Time Distributed Conv2D network with a Long Short-Term Memory layer. This model accepted all 4 inputs being (frames, height, width, channels) as it is also iterating over each video for classification. Time Distributed networks are great for classifying frames in chronological order like a video is.

Unlike CNN a Time Distributed CNN will apply its layers to each frame individually rather than one at a time before it merges them back to create a single classification based on each frame in the video. Time Distributed networks combined with LSTM provide to be especially useful. LSTM allows us to detect relationships between frames in chronological order. Given the benefits stated above about Time Distributed and LSTM networks it did not manage to have the highest accuracy (81.5%) it also wasn't a very stable network as results tended to fluctuate. I suspect this problem was due to low memory of my Virtual Machines and as a result I had to limit my data sample. With more data I are confident that Time Distributed networks with LSTM are the best methods for this problem.

7. Conclusion

Overall I am satisfied with the performance of all of my models with ResNet50 being the best overall at an 84.475% validation accuracy. Unfortunately, the Time Distributed Conv2D network with LSTM did not perform as Ill as expected, but I believe this was due to the major limitation of working with the enormous size of the data. In the future I would like to explore additional uses of Time Distributed networks, RNN networks, LSTM networks and finally adding additional GPU’s and memory to my Virtual Machines in order to handle all of the data provided.

8. Percentage of Code

(283 - 16) / (283 + 90) * 100 = 71.58%

9. References

Ming Cheng, Kunjing Cai, and Ming Li. "RWF-2000: An Open Large Scale Video Database for Violence Detection." arXiv preprint arXiv:1911.05913 (2019).

9.1. Links

https://github.com/mchengny/RWF2000-Video-Database-for-Violence-Detection

https://medium.com/smileinnovation/how-to-work-with-time-distributed-data-in-a-neural-network-b8b39aa4ce00

https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/

https://www.worldscientific.com/doi/pdf/10.1142/S2196888820500013

https://blog.coast.ai/five-video-classification-methods-implemented-in-keras-and-tensorflow-99cad29cc0b5