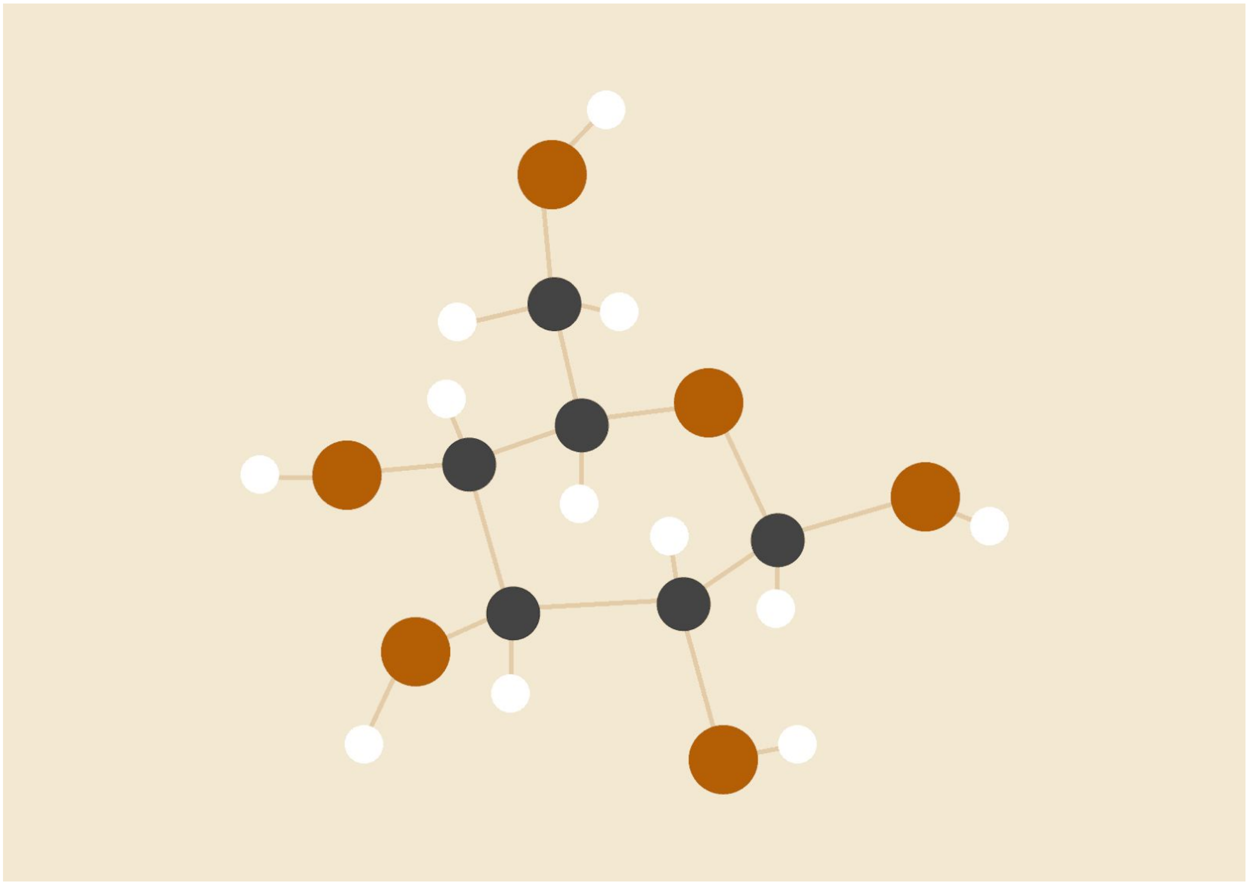
Violence Detection in Video Classification

Utilizing the RWF-2000 Video Database



Changhao Ying, Hongfei Niu, Spencer Staub

12/07/2020

Machine Learning II

Amir Jafari

1. 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 we 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.

2. 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 well 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.

2.1. 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.

3. Preprocessing

Preprocessing was performed separately for each model, however, there were 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 we can resize each video to (224,224), (64,64) and (50,50). Each resize is a separate directory and will be used by different models depending on the memory and computing power needed. Files are finally saved to a directory formatted like the unzipped RWF-2000 videos and can be accessed the same way.

Since our dataset is extremely large we 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 our 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 our 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 we can create x and y numpy arrays for both the train and test sets.

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

4. Networks

A variety of networks were 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.

4.1. MLP

4.1.1. Setup

The MLP Network starts with 64train/16test examples. We are 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 (50,50,5). I randomly picked 64 train/ 16 test videos to fit the model.

4.1.2. Network

It is a 2 layer network with 100 and 200 neurons respectively with activation relu. A final dense layer with 2 neurons and a sigmoid activation to perform binary classification.

4.1.3. Compile and Fit

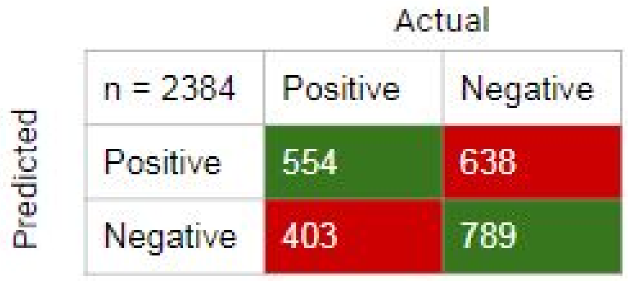
The network is compiled with binary Crossentropy, Adam Gradient Descent, and accuracy metrics. Batch size is 512 with 30 epochs.

4.1.4. Results

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

Accuracy:56.33% F1-Score:56% Cohen Kappa:12%

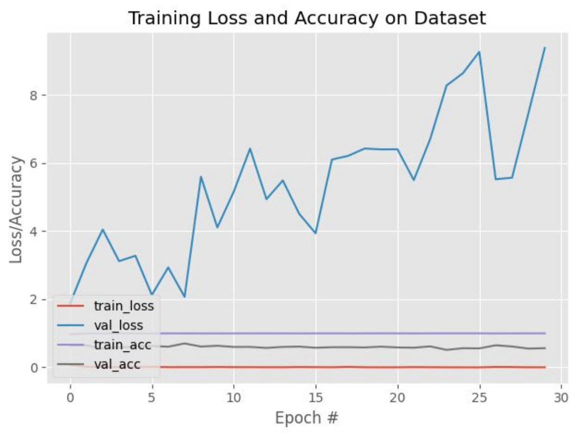
Confusion Matrix:



Actual

Predicted	n = 2384	Positive	Negative
	Positive	554	638
	Negative	403	789

Loss/Accuracy Graph:



4.2. Conv1D

4.2.1. Setup

The Con1d Network starts with 64train/16test examples. We are 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 (50,50,5). I randomly picked 64 train/ 16 test videos to fit the model.

4.2.2. Network

It is a 2 layer network of (5,5) kernel size, 3 filters with activation relu in the first layer. There is a MaxPooling1D with 3 strides. I flatten it and used 100 neurons, 0.1 dropout with activation relu in the second layer. A final dense layer with 2 neurons and a sigmoid activation to perform binary classification.

4.2.3. Compile and Fit

The network is compiled with Binary Crossentropy, Adam Gradient Descent, and accuracy metrics. Batch size is 200 with 30 epochs.

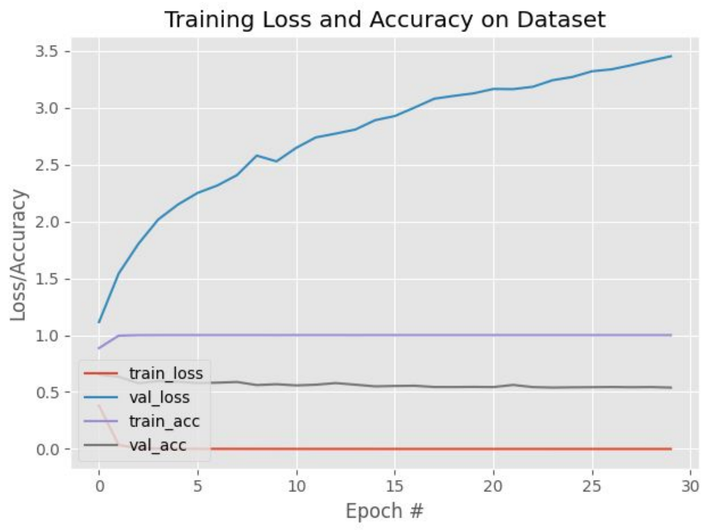
4.2.4. Results

Accuracy: 53.9 % F1-Score: 0.52 Cohen Kappa: 0.078

Confusion Matrix:

n = 2384	Positive	Negative
Positive	464	728
Negative	371	821

Loss/Accuracy Graph:



Above shows increasing validation loss which is not conducive to a well performing model.

4.3. Conv2D

4.3.1. Setup

The Conv2D Network starts with 256 training examples and 64 test examples. We are 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).

4.3.2. 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 we 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.

4.3.3. 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.

4.3.4. Results

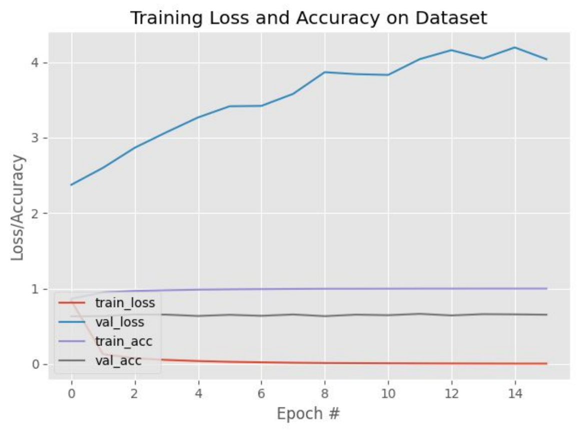
Since this is a balanced dataset accuracy will be our best metric to use, but we will also utilize F1-Score, Cohen Kappa and a confusion matrix to measure the performance of each network.

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

Confusion Matrix:

n = 64	Positive	Negative
Positive	26	6
Negative	7	25

Loss/Accuracy Graph:



4.4. Resnet50

4.4.1. Setup

The Resnet Network starts with 256 training examples and 64 test examples. We are 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).

4.4.2. 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 our input of (64,64) works well too. Our 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

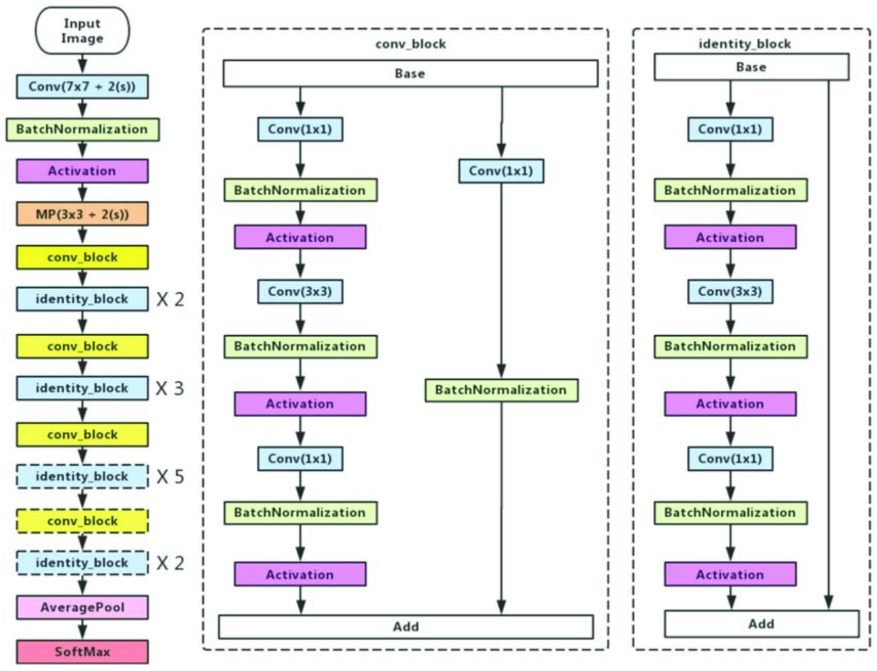


Figure 1

4.4.3. 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.

4.4.4. Results

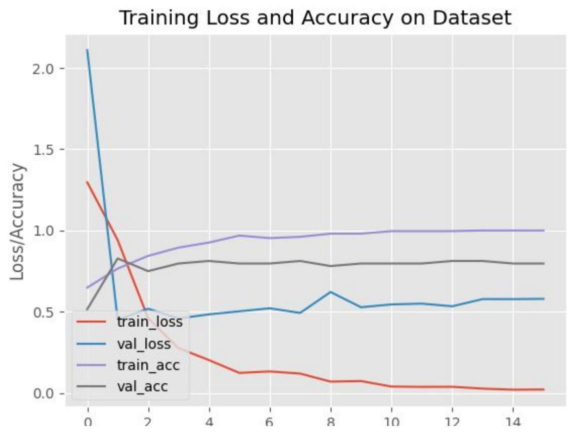
Since this is a balanced dataset accuracy will be our best metric to use, but we will also utilize F1-Score, Cohen Kappa and a confusion matrix to measure the performance of each network.

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:



4.5. EfficientNet-B0

4.5.1. Setup

The EfficientNet Network starts with 200 training examples and 60 test examples. We are 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). For better performance, the videos are cropped again with 28 frames each clip. And the final input is in shape of (814, 5, 28, 100*100).

4.5.2. Network

Convolutional neural networks are usually developed at a fixed computational resource, and then increase the dimensions when more resources are added in order to reach higher accuracy. For example, VGG can extend from VGG-16 to VGG-19 by increasing the number of layers. Although this method does improve accuracy, they usually require long manual tuning and still produce secondary performance.

Compared with traditional CNN, which arbitrarily increases the network dimensions, the EfficientNets use a series of fixed scaling factors to uniformly adjust the network dimensions, which reach 10 times efficiency and higher accuracy than SOTA.

In this paper, the EfficientNet is applied with the pretrained weight “efficientnet-b0” and is added with a con2d layer in order to change the input channels from 3 to 5. The network has an image input size of (224,224), but our input of (100,100) works well too. The Full-Connection part of our network model contains: an average pooling with shape (2,2), a dense linear layer of 512 neurons with batch normalization and relu process, a dense function of 256 neurons with batch normalization, relu process and a 0.2 dropout, finally a dense layer with 2 neurons and sigmoid activation for binary classification.

EfficientNet-B0 Architecture

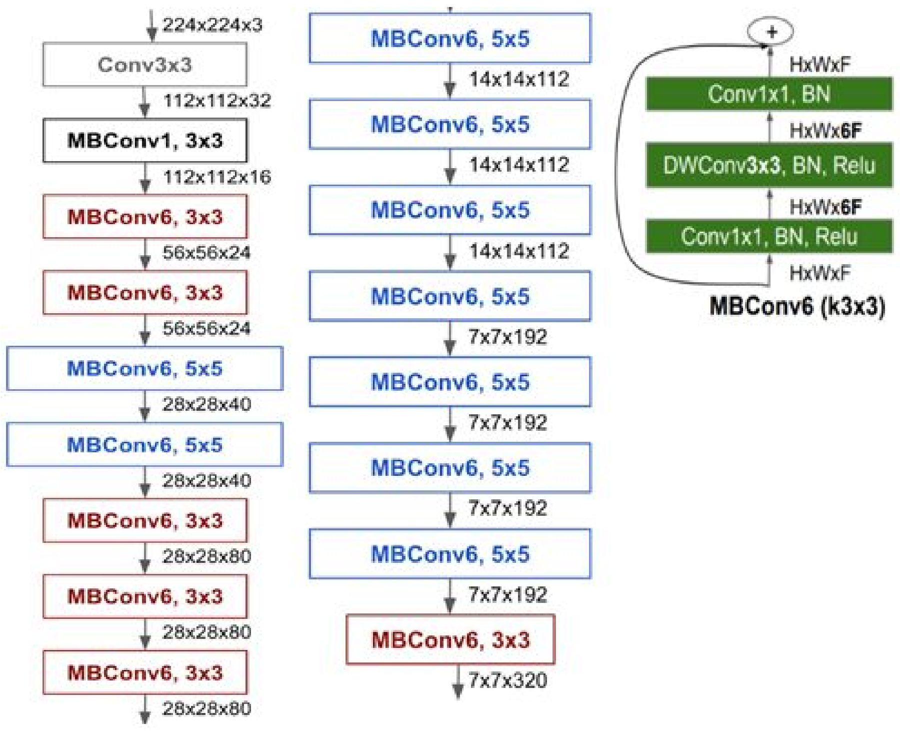


Figure 2

4.5.3. Compile and Fit

The network is compiled with Binary Crossentropy, Stochastic Gradient Descent, and accuracy metrics. Batch size is 100 and the model does not set a fixed epoch number. The best model is saved when the minimum loss value refreshes.

For the learning rate, the initial value is setas 1e10^-3, and the weight decay is 0.5*LR. Three decreasing schedulers are also set in order to adjust the result, avoiding falling into local optimum. The model chooses SGD optimizer, pursuing both direction accuracy and speed.

4.5.4. Results & Evaluation

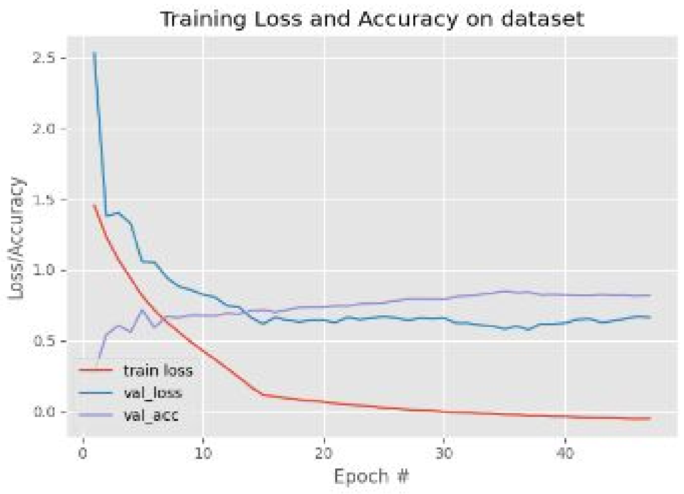
Since this is a balanced dataset accuracy will be our best metric to use, the Accuracy, BCE Loss and Confusion Matrix will be applied to test the result.

Accuracy: 0.791659 ; BCE Loss: 0.7271

Confusion Matrix:

n = 229	Positive	Negative
Positive	105	18
Negative	32	74

Loss/Accuracy Graph:



Based on the result, the model classifies the Violence video relatively better than the Nonviolence videos, which may be caused by the moving noises. The L2 regularization and LR scheduler make effort because the loss value decreases stable and well. Overall, the model has a relatively good ability of classification, and we will drop the optical flow data in the next model and make a contrast.

4.6. VGG-16

4.6.1. Setup

The VGG-16 Network starts with 200 training examples and 60 test examples. We are unable to utilize all 800 and 200 train/test files due to memory issues. The network uses 3 flows (3 RGB). For better performance, the videos are cropped again with 28 frames each clip. And the final input is in shape of (814, 3, 28, 100*100).

4.6.2. Network

Based on Traditional AlexNet, for the structure of VGG, the three 3x3 convolution kernels are applied instead of 7x7 convolution kernels, and two 3x3 convolution kernels are applied instead of 5*5 convolution kernels. Under the same condition of the perceptual field, the VGG model increases the depth of the network structure, and the effect of the neural network could be improved to a certain extent..

In this paper, the VGG-16 is applied with the pretrained weight, in order to increase the training speed. The network has an image input size of (224,224), but our input of (100,100) works well too. The Full-Connection part of our network model contains: an average pooling with shape (2,2), a dense linear layer of 512 neurons with batch normalization and relu process, a dense function of 256 neurons with batch normalization, relu process and a 0.2 dropout, finally a dense layer with 2 neurons and sigmoid activation for binary classification.

VGG-16 Architecture

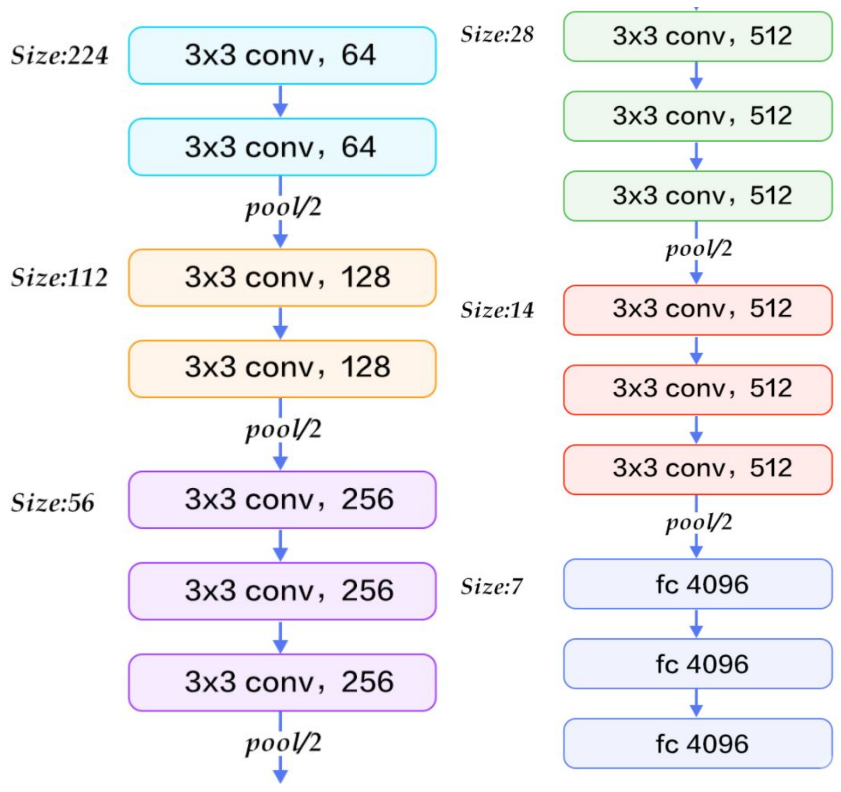


Figure 3

4.6.3. Compile and Fit

The network is compiled with Binary Crossentropy, Stochastic Gradient

Descent, and accuracy metrics. Batch size is 100 and the model does not set a fixed epoch number. The best model is saved when the minimum loss value refreshes.

For the learning rate, the initial value is setas 1e10^-3, and the weight decay is 0.5*LR. Three decreasing schedulers are also set in order to adjust the result, avoiding falling into local optimum. The model chooses SGD optimizer, pursuing both direction accuracy and speed.

4.6.4. Results & Evaluation

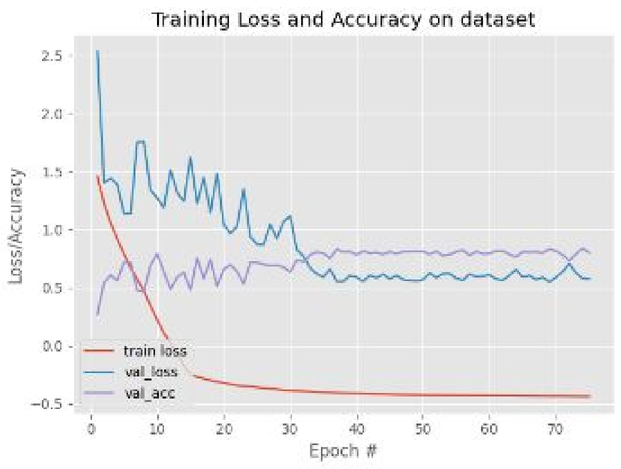
Since this is a balanced dataset accuracy will be our best metric to use, the Accuracy, BCE Loss and Confusion Matrix will be applied to test the result.

Accuracy: 0.8340611 BCE Loss: 0.6896

Confusion Matrix:

n = 229	Positive	Negative
Positive	112	11
Negative	22	79

Loss/Accuracy Graph:



Based on the result, the model classifies the Violence video still slightly better than the Nonviolence videos, which may be also caused by the moving noises. And for the training process, it seems that the schedulers do not make good effort in the beginning, because the Loss line fluctuates much. Maybe, the initial value of the LR could be smaller and we will get a better decreasing trend. Overall, the model has relatively better results than the EfficientNet-B0 model. It means that the optical flow data have some noises, such as the moving of the background or camera, which will negatively influence the training. However, if we could classify the background and target people to calculate useful optical flow data, then make a classification, I guess the result will improve a lot.

4.7. Time Distributed CONV2D + LSTM

4.7.1. Setup

The Time Distributed Conv2D + LSTM Network starts with 256 training examples and 64 test examples. We are 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).

4.7.2. Network

This network begins with an input layer of (batch_size, 149, 64, 64,) and then proceeds to its first TimeDirstributed Conv2D layer. We utilize the TimeDistributed to evaluate multiple frames of a video and make a classification from there. After the Conv2D layer we 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.

4.7.3. 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.

4.7.4. Results

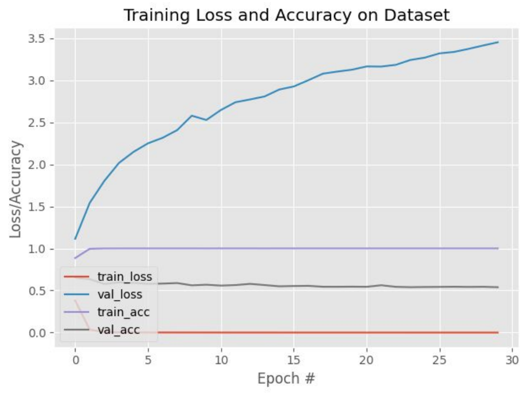
Since this is a balanced dataset accuracy will be our best metric to use, the Accuracy, BCE Loss and Confusion Matrix will be applied to test the result.

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.

5. Model Results and Evaluation

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

When initially building the networks we started with simple MultiLayer Perceptron networks as they were easy to implement. These networks accepted a 2-dimensional input shape with shape (frames * hight * width, channels) channels being 3 RGB and 2 Optical Flow. These produced mediocre results as we were classifying each frame individually without thought of previous frames. The same goes for our first Convolutional Neural Network of Conv1D. This also accepted a 2-dimensional input shape and performed similarly to the MLP Model. These results were expected given that this problem requires human action recognition in video requiring the network to classify actions across frames and not a single photo.

After the 2-dimensional networks were trained we moved to 3-dimensional networks with Conv2D. Each network had an input of (frames, height*width, channels), frames being the added dimension as the networks will be iterating over each video this time. We utilized three pretrained networks in this experiment; ResNet50, EfficientNet-B0 and VGG-16. All of these pretrained networks are based off of the Imagenet database and all performed well. The lowest being VGG-16 at 79% accuracy and the highest being EfficientNet-B0 at 83% and ResNet50 at 84%. These results were also expected to be better than our 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, we 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. We suspect this problem was due to low memory of our Virtual Machines and as a result we had to limit our data sample. With more data we are confident that Time Distributed networks with LSTM are the best methods for this problem.

6. Conclusion

Overall we are satisfied with the performance of all of our models with ResNet50 being the best overall at an 84.475% validation accuracy. EfficientNet-B0 came in a close second, which makes sense since both of these pretrained networks worked off of the same data. Unfortunately, the Time Distributed Conv2D network with LSTM did not perform as well as expected, but we believe this was due to the major limitation of working with the enormous size of the data. In the future we would like to explore additional uses of Time Distributed networks, RNN networks, LSTM networks and finally adding additional GPU’s and memory to our Virtual Machines in order to handle all of the data provided.

7. 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).

7.1. 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