Detect Violence detection system.

In this project we present two parts with different models which detect violence.

The both part can analyze videos and identify whether the video shows violence or not.

The first part:

In the first part we actually created two systems to compare between them

The first system is a simple system of logistic regression that also called precpecton .

The second systems is uses more layers and called multi layer percepton.

Of course the multi layer perceptron system gives significantly better results.

About the first system:

In the first system We built a model work like logistic regression called precpecton. It is simple model with one layer. Which is calculation of sum of input vectors with the value multiplied by corresponding vector weight. The displayed output value will be the input of an activation function.

After we build the model, we train him on dataset and check validation to see the result.

About the second system:

We wanted to improve the results, so we built a new model (multi layer percepton) which is more comlicated.

This model made from several layers, to solve complex problems. The diagram below shows an MLP with three layers. Each input in the first layer on the left (the input layer), sends outputs to all the perceptrons in the second layer (the hidden layer), and all perceptrons in the second layer send outputs to the final layer on the right (the output layer).

The diffrence between the two models:

one layer percpecton model (logistic regression) represents the most simple form of neural network, in which there is only one layer of input nodes that send weighted inputs to a subsequent layer of receiving nodes, or in some cases, one receiving node. This single-layer design was part of the foundation for systems which have now become much more complex. A multilayer perceptron (MLP) is a class of feedforward artificial neural network that contains one or more hidden layers (apart from one input and one output layer). While a single layer perceptron can only learn linear functions, a multi layer perceptron can also learn non – linear functions.

The first part has three files.

Now we will explain about the files and what its purpose to do:

- ***input data A-***

  Cv2 code from kaggle from dataset page

  The code split the videos with cv2 (Python Video Library) to five different videos to enlarge the dataset

  After that code transform the new videos to matrix (numpy array) and save them in csv file .

- ***Model-gourmet-percepton-logloss and Model_gpu_multilayer_perceptron***-

  The both models in this file have Auxiliary Functions and import dataset:

  Auxiliary Functions:

  Sigmoid function to improve the one layer percpecton

  Predictor function to improve the multy layer percpecton

  Function that returns all the metrics of classification :currency AUC ROC are return a table of true/false positive/negative

  We use function to save us the system for compare between models or further training

  Import dataset:

  We take the data and split it to tree groups train, test and validation.

  Train for train the model, **Test** to check the model, and Validation is a like new unused before videos.

Models:

We used sklearn Percepton as benchmark.

We have built Logistic Regression in Tensorflow which behavior is close.

Now we want to improve the model, for that we build the MLP model.

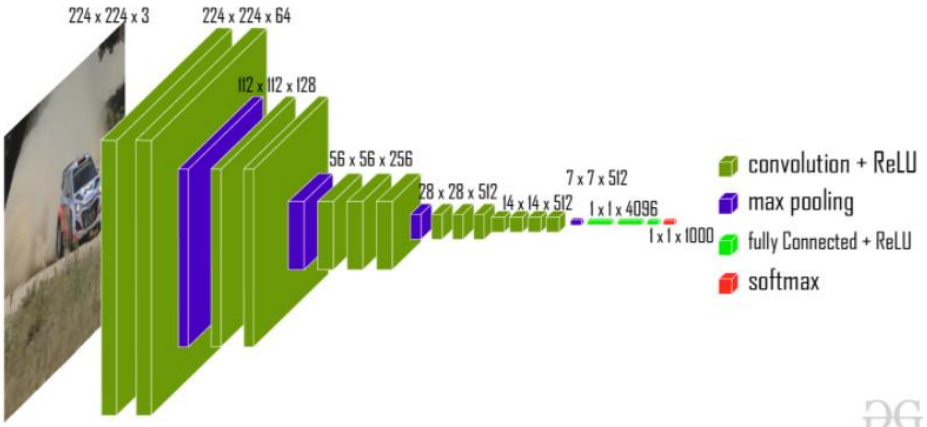
So we do all necessary stages.

After we do that we check the metrics on new model and the result is more better with MLP model.

The second part

This system uses extracting a set of frames belonging to the video, sending them to a pretrained network called VGG16, obtaining the output of one of its final layers and from these outputs train another network called LSTM.

VGG is a specific convolutional network designed for classification and localization.Like many other popular networks like Google-Net, Alex Net etc.



VGG-16 architecture

LSTM- Long short-term memory is an artificial recurrent neural network (RNN) have memory and are able to analyze the temporal information of the video and can distinguish between a video that had violence and a video that did not.

If the dataset already exists thanskip the first step which is to download the data.

if you want to put your own dataset put it into the project folder, in folder named data/Violence the Violence data and in folder named data/NonViolence the non Violence data.

The following folders are also used in this project:

Prerequisites,Python3,opencv-python,numpy,matplotlib,keras,h5py

How to install:

Make sure python3 and pip is installed

then install:

install opencv-python

pip install cv2

install PyWavelets

pip install numpy

install matplotlib

python -m pip install -U matplotlib

install keras

pip install Keras

install h5py

pip install h5py

install twnsorflow 2.4

Now we will explain each part of the project:

1) ***Part One Downloading the data set***

We will use the function **download_data** to download the datasets.

This function will use the **maybe_download_and_extract** function to Download and extract the data if it doesn't already exist.

the url for the dataset was takken from
https://www.kaggle.com/mohamedmustafa/real-life-violence-situations-dataset

we uploaded the dataset to dropbox.

In addition we will use the auxiliary function **_print_download_progress**

to print the amount of download processed.

2) ***Part Two Loading the data set and initialization :***

* First of all, we define the directory to place the video dataset ,

* Then we will wish each image a size of 224 * 224

* We initialize the number of channels by three (RGB).

* Then we will then initial two classes for classification
   (of violence and non-violence).

* We will initialize a variable of the amount of files we want to send for training.

* Initialize a variable in the number of frames per video.

*Initialize a variable in the Number of frames per training set
(_num_images_train = _num_files_train * _images_per_file).

*Function **get_frames –** function used to get 20 frames from a video file and convert the frame to a suitable format for the neural net.

*Function **label_video_names**– function used to label the dataset to Violence and NonViolence and save in data_dir folder.

more explanation for **label_video_names**:
The function run (in for) over the files and classifies each file with violence in 1-0 and with no violence in 0-1 And puts into an array called **labels** and puts the file names into another array call **names**

After that we use **begram** (zip function) on the lists **labels** and **names** and save it in a new list afther that we Suffle the new list data (names and labels)
And return the names and the labels.

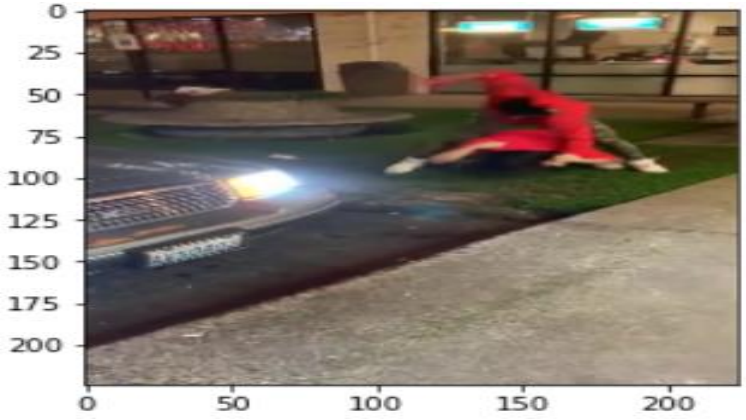
3) Plot a video frame to see if data is correct:

* we get the names and labels of the whole videos.

* we print the path and the name for the chosen video.

* we load 20 frames of one video for example**.**

* To plot the frames we convert the frames back into uint8 pixel format.



4) Pre-Trained Model-VGG16:

The following creates an instance of the pre-trained VGG16 model using the Keras API. This automatically downloads the required files, if you don't have them already.

The VGG16 model contains a CNN part and a fully-connected (or dense) part, which is used for classification. If include_top=True then the whole VGG16 model is downloaded, which is about 528 MB. If include_top=False then only the convolutional part of the VGG16 model is downloaded, which is 57 MB.

We can observe the shape of the tensors expected as input by the pre-trained VGG16 model. In this case it is images of shape 224 x 224 x 3. Note that we have defined the frame size as 224x224x3. The video frame will be the input of the VGG16 net.

****5) VGG16 model flowchart:***

The following chart shows how the data flows when using the VGG16 model for Transfer Learning. First we input and process 20 video frames in batch with the VGG16 model. Just prior to the final classification layer of the VGG16 model, we save the so-called Transfer Values to a cache-file.

The reason for using a cache-file is that it takes a long time to process an image with the VGG16 model. If each image is processed more than once then we can save a lot of time by caching the transfer-values.

When all the videos have been processed through the VGG16 model and the resulting transfer-values are saved in to a cache file, then we can use those transfer-values as the input to LSTM neural network. We will then train the second neural network by using the classes from the violence dataset (Violence, No-Violence), so that the network can learn how to classify images based on the transfer-values from the VGG16 model.

more explanation for VGG16 model flowchart:

**** We will use the output of the layer prior to the final***
**** classification-layer which is named fc2. This is a fully-connected (or dense) layer.***

**** We have Function to process 20 video frames through VGG16 and get transfer values.***

**** We have Function proces_transfer and it work like generator that process one video through VGG16 each function call.***

**** The functions make_files- it save transfer values from VGG16 used for training:***

- *Read the first chunk to get the column dtypes*
- *Initialize a resizable dataset to hold the output*
- *Write the first chunk of rows*
- *Resize the dataset to accommodate the next chunk of rows*
- *Write the next chunk.*
- *Increment the row count*

**** The functions make_files_test - save transfer values from VGG16 used for testing:***

- *Read the first chunk to get the column dtypes.*
- *Initialize a resizable dataset to hold the output.*
- *Write the first chunk of rows.*
- *Resize the dataset to accommodate the next chunk ofrows.*
- *Write the next chunk.*
- *Increment the row count.*

*6) Split the dataset into training set and test set:

In this point We are going to split the dataset into training set and testing. The training set is used to train the model and the test is used to check the model accuracy.

*make the training set :

In this point we will process all the video frames for training through VGG16 and save the transfer-values.

*make the testing set:

Then we will process 20% the video frames for testing through VGG16 and save the transfer-values.

*7)Load the cached transfer values into memory:

*functions to load the saved transfer-values into RAM memory:

We saved all the video frames transfer-values into a disk, but we have to load those transfer-values into memory in order to train the LSTM net. if you didn't save the transfer-values into disk you would have to process the whole videos each training. It's a time consuming way to processe the videos through VGG16 net.

8)Recurrent Neural Network:

The basic building block in a Recurrent Neural Network (RNN) is a Recurrent Unit (RU). There are many different variants of recurrent units such as the rather clunky LSTM (Long-Short-Term-Memory) and the somewhat simpler GRU (Gated Recurrent Unit) which we will use in this tutorial. Experiments in the literature suggest that the LSTM and GRU have roughly similar performance. Even simpler variants also exist and the literature suggests that they may perform even better than both LSTM and GRU, but they are not implemented in Keras which we will use in this tutorial.

A recurrent neuron has an internal state that is being updated every time the unit receives a new input. This internal state serves as a kind of memory. However, it is not a traditional kind of computer memory which stores bits that are either on or off. Instead the recurrent unit stores floating-point values in its memory-state, which are read and written using matrix-operations so the operations are all differentiable. This means the memory-state can store arbitrary floating-point values (although typically limited between -1.0 and 1.0) and the network can be trained like a normal neural network using Gradient Descent.

Define LSTM architecture:

When defining the LSTM architecture we have to take into account the dimensions of the transfer values. From each frame the VGG16 network obtains as output a vector of 4096 transfer values. From each video we are processing 20 frames so we will have 20 x 4096 values per video. The classification must be done taking into account the 20 frames of the video. If any of them detects violence, the video will be classified as violent.

The first input dimension of LSTM neurons is the temporal dimension, in our case it is 20. The second is the size of the features vector (transfer values).

Model training-

****Function train_the_model-***

Allows training of a system that has never trained with the same data set

**** Function retrain_the_model-***

Allows training of a system that has been trained before with the same data set

```
train_the_model(20)
```

```
Epoch 1/20
2/2 - 5s - loss: 0.3207 - accuracy: 0.4933 - val_loss: 0.3039 - val_accuracy: 0.4976
Epoch 2/20
2/2 - 5s - loss: 0.2929 - accuracy: 0.4800 - val_loss: 0.2628 - val_accuracy: 0.5024
Epoch 3/20
2/2 - 4s - loss: 0.2598 - accuracy: 0.5040 - val_loss: 0.2490 - val_accuracy: 0.5035
Epoch 4/20
2/2 - 4s - loss: 0.2520 - accuracy: 0.4840 - val_loss: 0.2516 - val_accuracy: 0.4976
Epoch 5/20
2/2 - 4s - loss: 0.2506 - accuracy: 0.4960 - val_loss: 0.2495 - val_accuracy: 0.5024
Epoch 6/20
2/2 - 4s - loss: 0.2508 - accuracy: 0.5040 - val_loss: 0.2512 - val_accuracy: 0.5024
Epoch 7/20
2/2 - 5s - loss: 0.2506 - accuracy: 0.5040 - val_loss: 0.2482 - val_accuracy: 0.6918
Epoch 8/20
2/2 - 5s - loss: 0.2482 - accuracy: 0.6200 - val_loss: 0.2485 - val_accuracy: 0.4976
Epoch 9/20
2/2 - 5s - loss: 0.2490 - accuracy: 0.4960 - val_loss: 0.2458 - val_accuracy: 0.6494
Epoch 10/20
2/2 - 4s - loss: 0.2459 - accuracy: 0.6107 - val_loss: 0.2464 - val_accuracy: 0.5024
Epoch 11/20
2/2 - 5s - loss: 0.2462 - accuracy: 0.5040 - val_loss: 0.2411 - val_accuracy: 0.6953
Epoch 12/20
2/2 - 5s - loss: 0.2420 - accuracy: 0.6587 - val_loss: 0.2394 - val_accuracy: 0.6000
Epoch 13/20
2/2 - 5s - loss: 0.2399 - accuracy: 0.6253 - val_loss: 0.2333 - val_accuracy: 0.6847
Epoch 14/20
2/2 - 5s - loss: 0.2362 - accuracy: 0.6347 - val_loss: 0.2258 - val_accuracy: 0.7353
Epoch 15/20
2/2 - 5s - loss: 0.2285 - accuracy: 0.6653 - val_loss: 0.2191 - val_accuracy: 0.7071
Epoch 16/20
2/2 - 5s - loss: 0.2221 - accuracy: 0.6880 - val_loss: 0.2109 - val_accuracy: 0.7141
Epoch 17/20
2/2 - 5s - loss: 0.2129 - accuracy: 0.7040 - val_loss: 0.2044 - val_accuracy: 0.6871
Epoch 18/20
2/2 - 5s - loss: 0.2057 - accuracy: 0.6973 - val_loss: 0.1950 - val_accuracy: 0.7141
Epoch 19/20
2/2 - 5s - loss: 0.1992 - accuracy: 0.6947 - val_loss: 0.1819 - val_accuracy: 0.7282
Epoch 20/20
```

9)Test the model:

We test the model with 20% of the total videos, that were splited. this videos weren't used to train the model.

Print the model accuracy and loss:

Load the model & retrain the model:

Allows you to stop the training and allows you continued training from the stopping point.