1. Performance Evaluation

1.1. Dataset

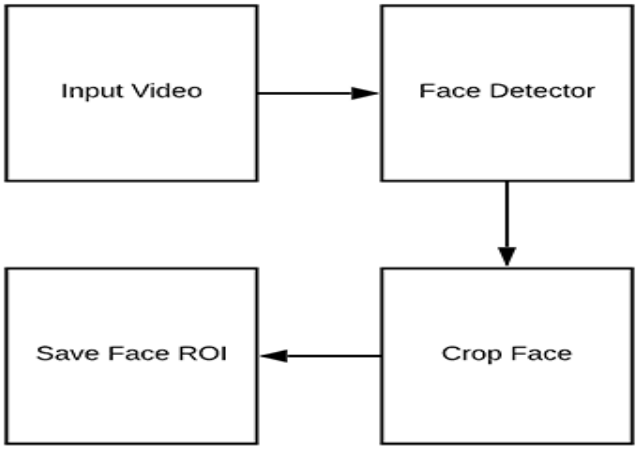


Figure 3: Detecting face ROIs in video for the purposes of building a liveness detection dataset.

The liveness model was trained on **ROSE-Youtu Face Liveness Detection Dataset.** The Database contains 4225 videos with 25 subjects in total (3350 videos with 20 subjects publicly available with 5.45GB in size). First, all publicly available videos are being downloaded and merged based on their liveness using windows video editor and then categorize into two classes i.e. real (live) and fake (not live). The OpenCV's deep learning face detector is then applied to both sets of videos to extract individual face ROIs for both classes. Since, the 'fake' video file is longer than 'real' video file, the longer skip frames was applied to help balance the number of ROIs for each class. This resulted into total of 40454 images with each class having 20227 images.

1.2. Experimental Results

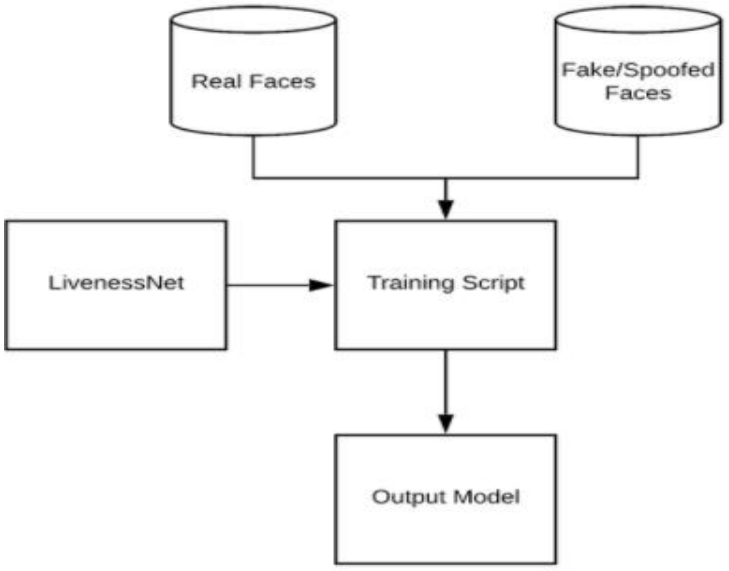


Figure 6: The process of training LivenessNet. Using both "real" and "spoofed/fake" images as our dataset, we can train a liveness detection model with OpenCV, Keras, and deep learning.

Several experimental evaluations were performed using shallow version of VGGNet CNN architectures. The reason for choosing such architecture is to ensure the liveness detector is fast, capable of running in real time. All the experiments were implemented on a 8GB RAM PC. The experiments are conducted several times on both GPU and CPU by changing the hyperparameters during the learning phase.

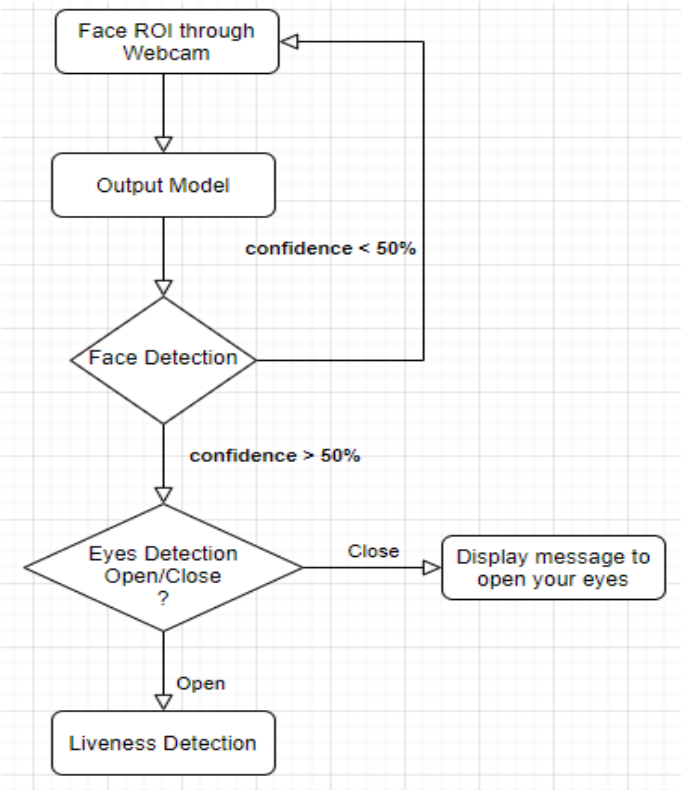
LR = Learning rate
LRD = Learning rate + Decay
D = Decay = LR / No. of Epochs

Batch Size = 32 for all

Epochs	Optimizer	Learning Rate	Train (%)		Validation (%)		Plots
			Loss	Accuracy	Loss	Accuracy	
20	SGD	0.001(LRD)	22.27	91.09	21.86	91.44	plot1
		0.001(LR)	18.86	92.73	18.73	92.47	plot2
		0.01(LR)	**10.81**	**96.09**	**8.29**	**97.11**	**plot3**
		0.01(LRD)	**13.13**	**95.13**	**7.95**	**97.25**	**plot4**
	Adam	0.001(LRD)	6.70	97.79	5.65	98.08	plot5
		0.001(LR)	7.17	97.61	4.44	98.66	plot6
		0.01(LRD)	**6.17**	**98.05**	**3.04**	**99.16**	**plot7**

All these experiments above performed on GPU. While doing so, GPU was approximately 20 minutes faster than CPU. From the above experiments, I found that the learning rate with a value of 0.01 either LR or LRD for each optimizer gives higher accuracy with limited overfitting while others tend to overfit more.

1.3. Working Flow



When you execute a module i.e. **liveness_demo.py**, the process shown in the figure above happens. The eye detection is done by OpenCV's Haarcascades eye detector whereas the face detection is done through OpenCV's deep learning face detector.

1.4. Limitations

- Since, ROSE-Youtu Face Liveness Detection Database only consider three spoofing attacks such as printed paper attack, video replay attack and masking attack, the built model can't detect 3D mask attack.
- Even though the model has a higher accuracy of 99.16%, the performance is still mediocre because of the limited dataset which lacks variation in ethnicity, skin tones and lighting conditions.