# Face Liveness Detection Based on Perceptual Image Quality Assessment Features with Multi-scale Analysis

Jatin Patel (1229894), Hitarth Panchal (1214537), Vaibhav Parikh (1217458), Anik Dutta (1191528), Dr. Garima Bajwa

*Abstract*—**The vulnerability in the biometric is still a major problem. The faces is being used globally as the key area for biometric purpose. The spoofing attack can also be carried out on the face detection. This paper tries to solve the problem through find the descriptors of the image using the symmetric gaussian and asymmetric gaussian. The research paper carries out the 21 features and from the image and later on random forest algorithm is used. This results are compared with the results of the previous research paper and what work they have done. This paper end with how the better results can be achieved by using other deep learning model.**

*Index Terms*— **HaarCascade, Mean subtracted and contract normalized (MSCN), Probability Density Function (PDF), Symmetric Generalized Gaussian Distribution (SGGD), Asymmetric Generalized Gaussian Distribution (AGGD), Skewed Voigt Distribution (SVD), Gradient Magnitude Similarity (GMS), Effective Pixel Similarity Deviation (EPSD), Convolutional Neural Network (CNN), Image Quality Assessment (IQA)**

## I. Introduction

FFacial Biometric system is one of the most prominent security issues since facial recognition is prone to spoofing attacks owing to the availability of many images on social media, making it easily reproducible. A general face biometric system has three parts: Face Detection, Face Liveness, and Face Recognition. The paper we are following focuses on the Face liveness aspect of the facial biometric system. It proposes a perceptual image quality assessment (IQA) using a multiscale analysis. The basis for this algorithm is the generation of certain artifacts during the recapturing and manufacturing process, like the Luminance artifact and the Distortion artifact. The paper introduces luminance quality-aware features for contrast and structural distortion. For the distortion artifacts, they used the gradient magnitude similarity feature, which is mainly sensitive to the artifacts generated as a result of the video and image compression algorithms employed along with the motion blur. The Mean Subtracted and Contract Normalized (MSCN) coefficients, which were used extensively in Image Quality Assessment, were modified to be used as the luminance-aware quality feature for the Face Liveness detection, and the Effective Pixel Similarity Deviation (EPSD) has been used to get the standard deviation map by selecting effective pixels and combining both of them to get a multiscale descriptor consisting of 21 features for classification. Later on the random forest algorithm is used to train the model and this model is compared with similar results. This ends with how the accuracy of the algorithm can be improved with the help of the deep learning model.

## II. Dataset and preprocessing

### A. Dataset

The dataset is the CASIA Face AntiSpoofing Dataset (2012), which consists of 600 videos. The Dataset has 200 real videos and 400 attack videos. The dataset has a total of 50 subjects. It has 12 videos for each subject under different resolutions of the video the second, fourth, sixth, and eighth are low-quality videos captured using a used USB camera since camera usage degraded the image quality. The first, third, fifth, and seventh are the normal quality captured using a fairly newer (not degraded) USB camera. The High-Resolution images were captured using a Sony Nex 5 camera, which had a maximum resolution of 1920x1080, but the images were cropped to 1280x720 size patches to capture. This was done to reduce the amount of calculations required while not losing out on the quality of the images. The motion type of blinking was considered an important factor in the dataset since it can act as a pivot for face liveness determination. But even that can be spoofed using eye cut-out photos as it has been done in the dataset generation, where they printed the high-resolution images mentioned above on the best quality copper paper. These printed images were used to make the warped image attack videos and the cut-out attack videos. The high-quality video replay attacks were the same 1280x720 ones captured with Sony Nex 5 but were downsized due to the iPad's resolution.

### B. Video Preprocessing

Every single frame of every single video was preprocessed to improve the speed of the descriptors detection. Haar cascade algorithm for face detection was used for the face detection from every video frame. After the detection of the face from the particular frame, the box is formed, which pinpoints the entire face. The pixel coordinate from the box starts, and the size of the box is stored in the Excel data. The Excel data has the following information: the name of the frame, video name, x coordinate, y coordinate, height, width, and information about whether the image is real or fake. The algorithm for this was ran on the Google Colab and data was stored at the google drive. This face identification was carried out every single frame of all 600 videos.

## III. Feature Extraction

The authors have presented a novel technique for detecting face liveness in the paper. This approach tries to extract features pertaining to image quality on multiple scales. Their proposal provided a manual feature engineering approach measuring 21 features. The following section delves deep into how each of them is calculated.

### *A. Extraction of Feature 1-2*

Mean subtracted and contract normalized (MSCN) coefficients [1] are very sensitive to image quality. Authors are using modified MSCN values to employ it for face liveness detection. The same is calculated as follows:

$$I_r[i,j] = (I[i,j] - \mu_I[i,j])/(\sigma_I[i,j] + 1)$$

Here, I is the grayscale face image obtained from section [cite]. Assuming it's shape is M x N. Also, i and j are local image coordinates, $0 \leq i < M$ and $0 \leq j < N$. $\mu_I$ and $\sigma_I$ are local mean and local standard deviation matrices with the same shape as I. Those are defined as follows:

$$\mu_I[i,j] = \sum_{u=-3}^{3} \sum_{v=-3}^{3} \mathrm{w}[u,v] * I[i+u, j+v]$$

Here, w a 2D Gaussian kernel that is zero-centered and has a standard deviation of 1.17 in horizontal and vertical directions. w has a shape of 7 x 7 as it's sampled up to three sigma to cover 99.7% of probability mass. Such is also normalized to have unit volume.

It can be seen below in Fig H1,

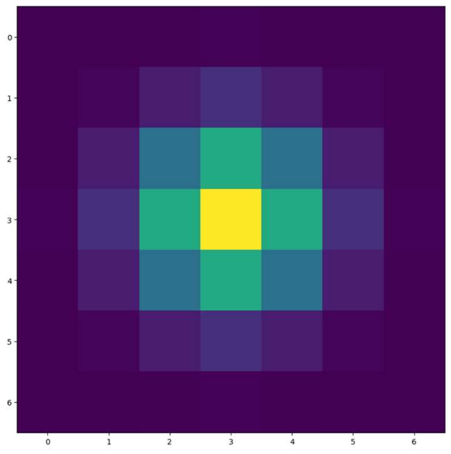


Figure 1:- Zero mean 2D Gaussian kernel of shape 7 x 7 with standard deviation of 1.17

$$\sigma_I[i,j] = \sqrt{\sum_{u=-3}^{3} \sum_{v=-3}^{3} w[u,v] * \{I[i+u, j+v] - u_I[i,j]\}^2}$$

$I_r$ is MSCN coefficient matrix. Since the Gaussian window is sized 7 x 7, the face image is edge padded first before calculating $I_r$ so that it retains the same shape as I, that is M x N. The values of MSCN matrix follow a bell-shaped distribution as you can see below in Fig H4,

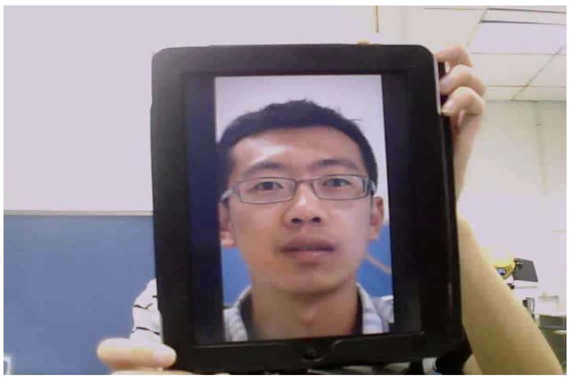


Figure 2: A frame from the fake video

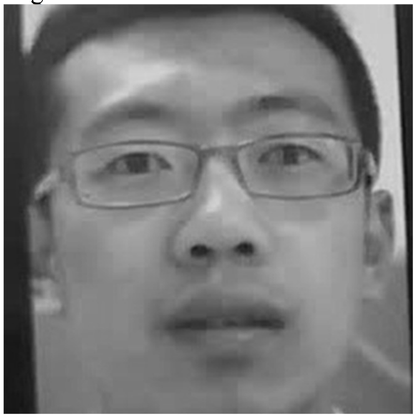


Figure 3: Detected face from the image shown in fig H2 and converted to grayscale

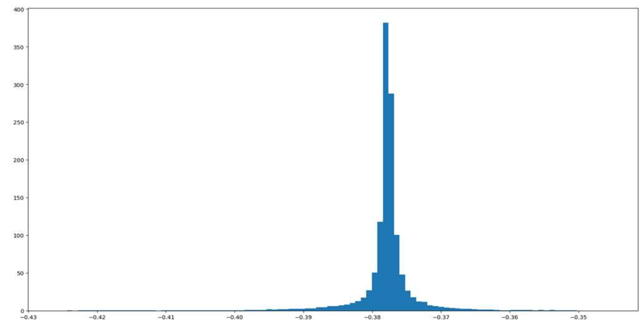


Figure 4: Normalized histogram of MSCN coefficients calculated from the image shown in Fig H3.

The researchers suggest estimating this distribution using Symmetric Generalized Gaussian Distribution (SGGD), a probability density function (PDF). It's defined as follows:

$$f(x; \mu, \sigma^2) = \left(\beta / \left(2 * \alpha * gamma(1/\beta)\right)\right) * e^{-(|x - \mu| / \alpha)^\beta}$$

Where $\text{gamma}(a) = \int_0^\infty t^{a-1} e^{-t}\, \text{dt}$ and $a > 0$

Variance of this distribution $\sigma^2$ is given by,

$$\sigma^2 = \alpha^2 * gamma(3/\beta)/gamma(1/\beta)$$

The parameters $\mu$ and $\sigma^2$ are estimated by minimizing the sum of negative log-likelihood of this function on observed data points, which in this case is our MSCN coefficients. Such is done using the SciPy Python library. $\mu$ and $\sigma^2$ form two features out of 21. The estimated SGGD distribution for MSCN coefficient distribution is shown in Fig H4 below.

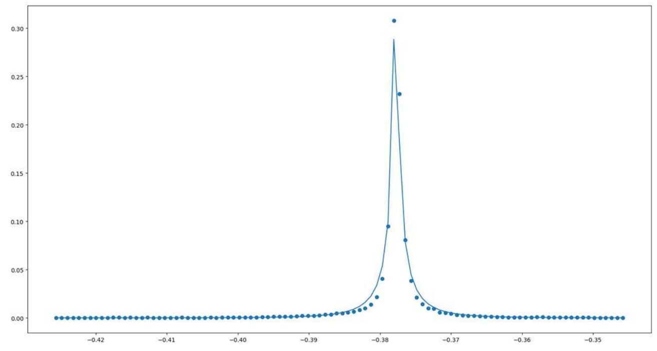
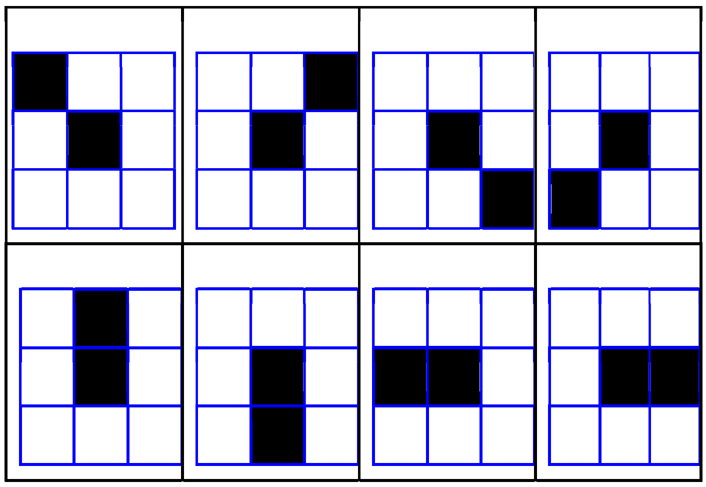


Figure 5: Estimated SGGD (line plot) imposed on the normalized histogram of MSCN coefficients (dotted/scatter plot)

The dotted/scatter plot indicates the normalized histogram of the MSCN coefficients, and the line plot refers to the estimated SGGD distribution.

## *B. Features 3-10*

The paper describes calculating these eight features by following [2]. It instructs to find the product of two adjacent MSCN coefficients. There are a total of eight such pairs, which are shown below.



The center MSCN coefficient is multiplied with eight neighboring coefficients to form eight adjacent MSCN matrices, each represented by ${I_d}^i$ where $0 \leq i < 8$. Before calculating ${I_d}^i$ , $I_r$ is padded such that ${I_d}^i$ has shame shape as $I_r$ which is M x N.

In the paper, the normalized histogram of each ${I_d}^i$ is estimated using Asymmetric Generalized Gaussian Distribution (AGGD), a probability density function (PDF). It's used in modeling skewed data distributions. It's defined as follows:

$$f(x; \mu, \beta, \alpha_1, \alpha_2) = k * e^{-[(\mu - x)/\alpha_1]^\beta} if x < \mu$$
$$f(x; \mu, \beta, \alpha_1, \alpha_2) = k * e^{-[(x - \mu)/\alpha_2]^\beta} if x \geq \mu$$

Where $k = \beta / \left((\alpha_1 + \alpha_2) * gamma(1/\beta)\right)$.

Here the parameters $\beta$, $\alpha_1$ and $\alpha_2$ determine the skewness in the data. Such can be represented by

$$\eta = (\alpha_2 - \alpha_1) * \left(gamma(2/\beta)/gamma(1/\beta)\right)$$

To estimate these parameters, the sum of negative log-likelihood of this function must be minimized. Minimization is usually done by calculating the partial derivatives with respect to the PDF's parameters. Differentiating a piecewise function such as the above is tricky, even for libraries like SciPy. Doing the same in SciPy gave inaccurate results. Using these results for further calculations might compromise the integrity of this work. One of the examples is shown below in Fig H6.

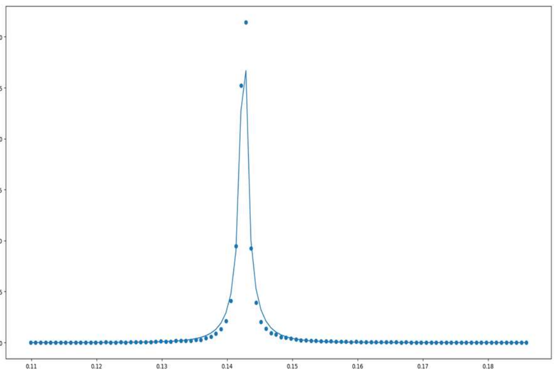


Figure 6: (Top) Input face grayscale image, (Bottom) Dotted plot refers to the normalized histogram of $I_d^1$ of the left image, and line plot refers to estimated AGGD distribution. The matching around the center is inaccurate. That area accounts for most of the probability mass.

We're estimating the Skewed Voigt Distribution (SVD) to circumvent this issue. Doing so gave us better results, as seen in Fig H7. The skewed Voigt function is described below.

$$\mathrm{f}(x; A, \mu, \sigma, \gamma, skew) = \mathrm{Voigt}(x; A, \mu, \sigma, \gamma)\left\{1 + \mathrm{erf}\left(\left(skew(x - \mu)\right)/\left(\sqrt{2\sigma}\right)\right)\right\}$$

Where

$$\mathrm{Voigt}(x; A, \mu, \sigma, \gamma) = \mathrm{A} * Real\{\mathrm{w}(z)\}/\sigma\sqrt{2\Pi} \text{ and}$$
$$\mathrm{z} = (x - \mu + i\gamma)/\sigma\sqrt{2}$$
$$\mathrm{w}(x) = e^{-x^2}\mathrm{erfc}(-ix)$$

$\mathrm{erfc}(x)$ is the complementary error function and $\mathrm{erf}(x)$ is the error function.

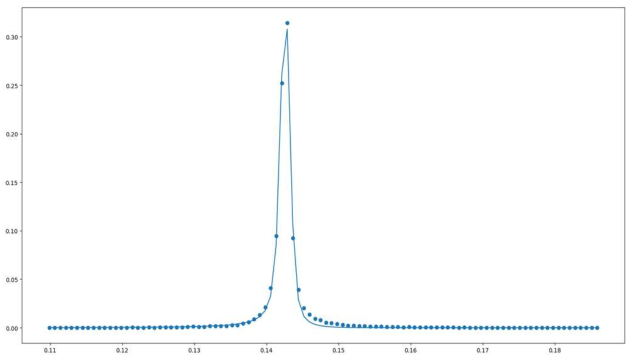


Figure 7: Estimated Skewed Voigt Distribution for the normalized histogram of $I_d^1$ of the left image shown in Fig H6.

For each $I_d^i$, the skew parameter is estimated. Since there are eight such matrices, there are eight skew values which form our feature 3 to feature 10.

## C. *Features 11-20*

Furthermore, the authors propose to calculate a downsampled image B. The grayscale face image I is downsampled by two and blurred using the 2D Gaussian kernel shown in Fig H1 to obtain B. Repeating the steps shown in Section 3A and 3B on image B, form feature 11 to feature 20.

## D. *Feature 21*

Here, the researchers propose to measure a gradient-based feature to strengthen the feature set further for the face liveness detection task. Firstly, the proposed technique instructs calculating distorted image D by applying a 2D Gaussian Kernel with standard deviation $\sigma_D$= 0.6 on I.

Afterwards, the gradient magnitude of I and D are calculated using horizontal and vertical Prewitt kernels shown below. The gradient magnitude of I and D are called $M_I$ and $M_D$ respectively.

$$h_x = [1/3, 0, -1/3; 1/3, 0, -1/3; 1/3, 0, -1/3]$$
$$h_y = h_x^T$$

$$M_I = \sqrt{(I \otimes h_x)^2 + \left(I \otimes h_y\right)^2},$$
$$\otimes \; denotes\ convolution\ operation$$
$$M_D = \sqrt{(D \otimes h_x)^2 + \left(D \otimes h_y\right)^2},$$
$$\otimes \; denotes\ convolution\ operation$$

Afterwise the Gradient Similarity Map (GMS) denoted as S is calculated as follows:

$$\mathrm{S[i, j]} = (2 * M_I[i,j] * M_D[i,j] + c) / (M_I[i,j]^2 + M_D[i,j]^2 + c)$$

Here, c is a small number used for stabilization purposes.

S is further altered using the following criteria.

$$s_t = \min\{\text{first } 5\ \% \text{ of decreasingly sorted S}\}$$

Change $\mathrm{S}[i,j]$ to 0 if it's less than the threshold $s_t$. Obtained S is now called Effective Pixel Similarity (EPS) map. The EPS map corresponding to Fig H6 left is shown below in Fig H8. Here, white pixels only passed the aforementioned criteria.

Figure 8: The EPS map corresponding to (Fig 6 top)

Furthermore, the average value of the EPS map is calculated, denoting it as $\mu_S$. Finally, the last feature Effective Pixel Similarity Deviation (EPSD) is measured using the following equation. Such concludes the feature extraction section.

$$EPSD = \sqrt{\left(\frac{1}{MN}\right) * \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} (EPS[i,j] - \mu_S)^2}$$

### E. *Execution of all features*

The code for the feature extraction was parallelized and run on the 4 laptops, where each laptop processed a particular set of frames. The frames were selected at intervals of milliseconds. The code was divided on the basis of the number of cores available on a particular PC. A total of 26 cores were available, 14 of which were in a single laptop with an i9 12th generation processor, while the other three laptops had only 4 cores on each one of them, two of them with an i5 9th generation, another with an i5 8th generation. Also, there was only one dedicated GPU available. The reason for parallelization was to reduce the computation time since the feature extraction was a very resource-intensive task, and we needed all the resources available at hand. It did make the computation a bit faster, but not significantly due to the nature of the processors. The final dataset with 21 features had data of approximately 112,000 frames. The feature extraction was processed on different laptops, so the final data was split up, which was then combined into a singular CSV file consisting of all features extracted from the videos and ready for our models to be trained on.

### F. *Random Forest*

The Random Forest algorithm was used for the classification of the videos into real or fake. There were many reasons for selecting this algorithm and why it performed well. The Random Forest algorithm uses the bagging and feature randomness to make a highly uncorrelated set of features to make decisions therefore, even after being an ensemble of decision trees unlike them, this would only consider the relevant features, which would be an important aspect to consider since we have a set of 21 features. Another reason was the size of our dataset, which consisted of 112000 frames; the data will also have non-linear relationships, high dimensional data, outliers, and imbalance in the dataset since there were 9 attack or fake videos and 3 real videos giving a ratio of 3:1. Our decision was fruitful which is evident from our results. We have used n_estimator as 100, which is used to get the best trade-off between model performance and time for training.

## IV. MobileNetV2: A Convolution Neural Network Based Approach

The Convolutional Neural Network mobile version 2 was used for the classification of the image into real and fake. One of the main reasons why this algorithm was chosen because it is light weight and can improve the computation speed. The other features of it were depth wise separable convolution, linear bottleneck and shortcut connections. The task at hand requires running the face liveness detection algorithm on videos imposing real-time runtime requirements. In recent years, Neural Networks have achieved high accuracy levels on various computer vision tasks, including image classification. Such revolution comes at a cost. Convolutional Neural Networks (CNN) achieving high precision requires more computational resources. However, [3] presents an efficient architecture that can run on mobile devices and embedded systems at a high speed. It's called MobileNetV2. We have finetuned this network for the binary image classification task initially pre-trained on the ImageNet dataset.

The training recipe to update the weight parameters is described as follows. Cross entropy loss function is minimized. The Stochastic Gradient Descent optimizer is utilized to update the CNN weights with an initial learning rate set to 0.1, momentum set to 0.9, and weight decay to 0.0001. The model is trained for three epochs using these settings on the laptop. The laptop was equipped with a Nvidia GeForce RTX 4060 GPU, i7-12700H, and 16GB RAM. Nvidia RTX 4060 has 8 GB of Video RAM. Running the same on this laptop took around 1.5 hours with a batch size of 400. During training, the dataset was augmented by randomly flipping the images in the horizontal direction and applying random affine transformations such as translation, rotation, scaling, and shear.

The following is architecture of the MobileNetV2 Architecture:

<Input Shape> (<Filter Type> <Filter Size>, <Number of Filters>, <Stride>) -> <Output Shape>

Block 1
224x224x3 (Conv2D 3x3, 32, s=2) -> 112x112x32

Block 2
112x112x32 (Conv2D 1x1, 192) -> 112x112x192
112x112x192 (DConv2D 3x3, 192, s=1) -> 112x112x192
112x112x192 (Conv2D 1x1, 16) -> 112x112x16

Block 3
112x112x16 (Conv2D 1x1, 96) -> 112x112x96
112x112x96 (DConv2D 3x3, 96, s=2) -> 56x56x96
56x56x96 (Conv2D 1x1, 24) -> 56x56x24

56x56x24 (Conv2D 1x1, 144) -> 56x56x144
56x56x144 (DConv2D 3x3, 144, s=1) -> 56x56x144
56x56x144 (Conv2D 1x1, 24) -> 56x56x24

Block 4
56x56x24 (Conv2D 1x1, 144) -> 56x56x144
56x56x144 (DConv2D 3x3, 144, s=2) -> 28x28x144
28x28x144 (Conv2D 1x1, 32) -> 28x28x32

28x28x32 (Conv2D 1x1, 192) -> 28x28x192
28x28x192 (DConv2D 3x3, 192, s=1) -> 28x28x192
28x28x192 (Conv2D 1x1, 32) -> 28x28x32

28x28x32 (Conv2D 1x1, 192) -> 28x28x192
28x28x192 (DConv2D 3x3, 192, s=1) -> 28x28x192
28x28x192 (Conv2D 1x1, 32) -> 28x28x32

Block 5
28x28x32 (Conv2D 1x1, 192) -> 28x28x192
28x28x192 (DConv2D 3x3, 192, s=2) -> 14x14x192
14x14x192 (Conv2D 1x1, 64) -> 14x14x64

14x14x64 (Conv2D 1x1, 384) -> 14x14x384
14x14x384 (DConv2D 3x3, 384, s=1) -> 14x14x384
14x14x384 (Conv2D 1x1, 64) -> 14x14x64

14x14x64 (Conv2D 1x1, 384) -> 14x14x384
14x14x384 (DConv2D 3x3, 384, s=1) -> 14x14x384
14x14x384 (Conv2D 1x1, 64) -> 14x14x64

14x14x64 (Conv2D 1x1, 384) -> 14x14x384
14x14x384 (DConv2D 3x3, 384, s=1) -> 14x14x384
14x14x384 (Conv2D 1x1, 64) -> 14x14x64

Block 6
14x14x64 (Conv2D 1x1, 384) -> 14x14x384
14x14x384 (DConv2D 3x3, 192, s=1) -> 14x14x384
14x14x384 (Conv2D 1x1, 96) -> 14x14x96

14x14x96 (Conv2D 1x1, 576) -> 14x14x576
14x14x576 (DConv2D 3x3, 576, s=1) -> 14x14x576
14x14x576 (Conv2D 1x1, 96) -> 14x14x96

14x14x96 (Conv2D 1x1, 576) -> 14x14x576
14x14x576 (DConv2D 3x3, 576, s=1) -> 14x14x576
14x14x576 (Conv2D 1x1, 96) -> 14x14x96

Block 7
14x14x96 (Conv2D 1x1, 576) -> 14x14x576
14x14x576 (DConv2D 3x3, 576, s=2) -> 7x7x576
7x7x576 (Conv2D 1x1, 160) -> 7x7x160

7x7x160 (Conv2D 1x1, 960) -> 7x7x960
7x7x960 (DConv2D 3x3, 960, s=1) -> 7x7x960
7x7x960 (Conv2D 1x1, 160) -> 7x7x160

7x7x160 (Conv2D 1x1, 960) -> 7x7x960
7x7x960 (DConv2D 3x3, 960, s=1) -> 7x7x960
7x7x960 (Conv2D 1x1, 160) -> 7x7x160

Block 8
7x7x160 (Conv2D 1x1, 960) -> 7x7x960
7x7x960 (DConv2D 3x3, 960, s=1) -> 7x7x960
7x7x960 (Conv2D 1x1, 320) -> 7x7x320

Block 9
7x7x320 (Conv2D 1x1, 1280) -> 7x7x1280

Block 10
7x7x1280 (AvgPool2D 7x7) -> 1x1x1280

Block 11
1x1x1280 (Conv2D 1x1, 2) -> 1x1x2

## V. Results

False Acceptance Rate (FAR):- The system falsely accepts the unauthorized user.
False Rejection Rate (FRR):- The system rejects the authorized user.

Half Total Error Rate (HTER):- This metric is used for biometric purposes to check the ratio of how many false users are accepted and rejected [1].

HTER= (FAR+FRR)/2

| | HTER | EER |
|---|---|---|
| Our Replication | 24.98 % | 19.98 % (th @ 0.26) |

| | | |
|---|---|---|
| Author | - | 12.7 % |
| **MobileNet V2** | **0.60 %** | **0.42 % (th @ 0.18)** |

Equal Error Rate (EER):- It is typically used on the verification task in biometrics. It is a point in the Receiver Operating Characteristic Curve (ROC) curve where the False Acceptance Rate (FAR) and False Rejection Rate (FRR) is equal.

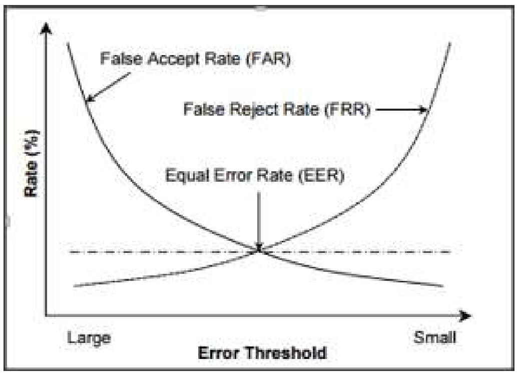


Figure 9:- Diagram of EER [4]

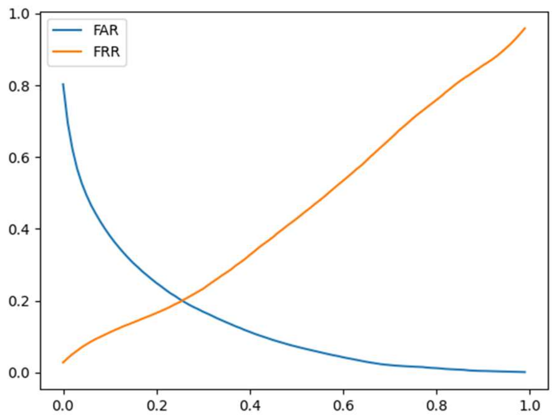


Figure 10: EER Plot of our Replication of Proposed Technique

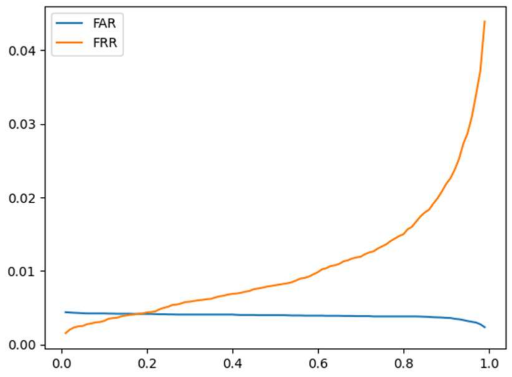


Figure 11: EER Plot of MobileNetV2 approach

## VI. Conclusion

Hitarth:- I worked on the feature extraction aspect of this paper. I had never worked on a density estimation problem before. My pursuit to replicate the ideas of the base paper taught me a lot about how the parameters of various probability density functions are estimated given a list of data points. Minimizing the negative log-likelihood function helps arrive at the best solution, while such works by calculating partial derivatives. Having learned all that, I'm confident that I can extend this work by fitting a different PDF on the data while also calculating various properties of the image pertaining to image quality. The computation time for manual feature extraction is very high as it goes from face detection to feature vector calculation. Feature vector calculation is done by minimizing complicated PDFs. The whole process takes a lot of time. Due to that, we weren't able to apply the same on other large face liveness detection datasets. Apart from that, I also programmed the CNN approach as they are superior at extracting relevant features automatically from the data, eliminating the need for manual feature engineering. Such can be seen in the results as it gives the lowest error rates. I believe I deserve a 70%-80% grade since I wasn't able to satisfy the first criterion. Due to the complexity of the AGGD function, I failed to model its parameters, and furthermore, the work wasn't performed on a novel dataset due to time constraints.

Vaibhav:- I worked on the preprocessing part, wrote code for the Symmetric Generalized Gaussian and tried to understand the Asymmetric Generalized Gaussian but the fitting of the graph was never as accurate as author..One of the main reason for it could be I didn't got the working of Beta 1 and Beta 2 I read research paper and understood that asymmetric generalized gaussian provides heavy tail which is difficult to get in the void skewed asymmetric model at that level which is required. I wrote code for the merging of the data from different PCs and lastly performed Random Forest on the processed data and calculated HTER. Deep learning model will perform better than the current model because hierarchical feature representation and it can also complexity in the face liveness. It is still very impressive that the just with the help of the descriptor extraction and applying machine learning model. We deserve marks because we tried to understand the Asymmetric Generalized Gaussian but wasn't successful in it. So it is not a perfect replication but quite close. Also CNN MobileNet V2 has performed the better the Authors (Equal Error Rate) EER and Half Total Error Rate (HTER).

Anik:- I worked on the paralyzing the process by the using the CPUs and GPUs for the feature extraction. I also works on the Asymmetric Generalized Gaussian distribution but faced similar problem as the other teammates faced. All of them were looking on the potential replacement of the asymmetric gaussian equation. I further worked on the PCA to potential reduce the and just value the important features.

We believe that even though it is not a perfect replication but still it had significant output. The CNN performed better than the authors EER and HTER.

Jatin:- To summarize, the main goal of replicating fell short as we were not able to reproduce the results from the original paper due to the tricky nature of the calculation of the Asymmetric Generalized Gaussian, but my fellow teammates found a way around the problem by using the Skew Voigt Gaussian instead for feature extraction, and the results were beyond the team's imagination. Also, the distribution of the processing (parallelization of processing on multiple laptops) helped us extract the features in time using the available resources, which would have been hard otherwise. The Random forest algorithm performed very well with the extracted set of features and was more effective than the original paper. Personally working on the Experimental analysis of the results I believe the decision to train a Mobile Net CNN model got us the best possible results with an HTER of less than 1% and an accuracy of 97%, which is quite a feat for us. Due to our approach based on multiple papers but not being able to replicate the original results, I believe a grade between 80-90 should be the best for our project.

## References

[1] D. L. Ruderman and W. Bialek, "Statistics of natural images: Scaling in the woods," in Advances in neural information processing systems, 1994, pp. 551-558.

[2] A. Mittal, A. K. Moorthy, and A. C. Bovik, "No-reference image quality assessment in the spatial domain," IEEE Transactions on Image Processing, vol. 21, no. 12, pp. 4695-4708, 2012.

[3] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov and L. -C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 2018, pp. 4510-4520, doi: 10.1109/CVPR.2018.00474.

[4] "Figure 1: Relationship Between FAR, FRR and Equal Error Rate (EER) [1]." ResearchGate, www.researchgate.net/figure/Relationship-between-FAR-FRR-and-Equal-Error-Rate-EER-1_fig1_342275067.

[5] "835.7 MB File on Mega." MEGA, mega.nz/file/9BFDiKqT#VLexsuFDZjoA97c1J_h9hInm8AG75h6kG-TUfm3hYwg. Accessed 14 Dec. 2023. (**CASIA Dataset Link**)