Face Spoofing Detection

Harshit Allagadda [37677101]

Abstract

This project presents a deep learning approach for Face Anti-Spoofing, addressing the growing need for secure facial recognition systems that can differentiate between real and spoofed faces. Using a Convolutional Neural Network (CNN) model trained on the CelebA dataset. The model incorporates real-time integration with a webcam, allowing for live analysis and detection of spoofed faces in practical applications.

Further, the project explores PatchCNN, designed to focus on local facial regions to enhance spoofing detection accuracy, particularly in challenging scenarios where traditional models may struggle. Additionally, the NUAA dataset is incorporated, offering diverse spoofing examples that complement the CelebA dataset and help improve the model's robustness.

Keywords- Face Spoofing, ***Deep Learning***, ***Convolutional Neural Network***, ***Histogram of Gradients***, ***CelebA-Spoof Dataset***, ***PatchCNN***, ***NUAA Dataset***, ***Spoofing Detection***, ***Facial Recognition***, ***Security Systems***

1 Introduction

In the modern era,most of the access control based systems are using some sort of biometric based authentication because they are simple and user friendly in nature.They also reduce the need for manually identifying user by facilitating automatic processing.Among various biometric modalities, facial recognition stands out as a vital source of visual information, easily captured in uncontrolled environments without requiring user consent[1].

Face recognition technology is at the forefront in the advancement of security measures, revolutionizing the way identity verification is approached across multiple domains, including banking, law enforcement, and secure access control. Leveraging distinct facial features allows this technology to provide a non-intrusive and seamless identity confirmation method, which significantly enhances user experience [2].

Moreover, recent advancements in deep learning, especially with Convolutional Neural Networks, have improved the accuracy of facial recognition systems. These systems can now learn intricate patterns from data, thereby enhancing their performance in real-time applications [1].

2 Motivation

The widespread adoption and benefits of Face Recognition Technology, introduced a security risk called spoofing attack. A spoofing attack occurs when an unauthorized user attempts to deceive a facial recognition system by presenting a counterfeit representation, such as a photo, video, or even a 3D model of a legitimate user's face[3].The development of spoofing attacks is in parallel with advancement of facial recognition technology. As systems become more advanced, so do the methods employed by attackers[4].

These spoofing attacks not only affect the organizations employing face recognition technology but also on the individuals whose identity is being compromised. This gives a call to develop robust countermeasure strategies to combat the spoofing techniques.Integrating these strategies

into conventional biometric applications is vital for bolstering security and mitigating unauthorized access risks[2].

3 Objectives

In this project, I am focused on the following key objectives:

To develop a Convolutional Neural Network model that effectively distinguishes between real and fake faces in various spoofing scenarios.

To test the model on the NUAA impostor dataset , to check for the model generalizability.

To integrate the CNN model with a webcam for real-time analysis, enabling immediate detection of spoofing attempts.

To develop a Patch Based CNN to improve accuracy for spoof detection by focusing on local patterns.

4 Literature Review

4.1 Evolution of Spoofing Attacks

The evolution of spoofing attacks targeting facial recognition systems evolved with technology. Earlier, the attacks were relatively unsophisticated mostly using low-resolution images or video recordings available from social media. These methods were typically easy for facial recognition systems to detect because they are of poor quality.

With progress in technology, attackers began employing more sophisticated and advanced techniques. It started with print attacks, where high-resolution photographs were used to trick recognition systems. Although they are simple, they remained effective against systems lacking robust liveliness detection mechanisms. Later on came, replay attacks, where pre-recorded videos of legitimate users were played in front of cameras. This method posed a more significant challenge for systems that could not differentiate between real-time interaction and recorded footage.

The evolution continued with the arrival of 3D mask attacks, in which attackers made use of 3D printing technologies to create realistic replicas of user's face. These masks incorporated depth and texture, making detection significantly harder.

The most remarkable development in this field has been the rise of deepfake technology. They can generate hyper-realistic videos that can imitate a person's facial expressions and movements, making it extremely difficult for detection. This discovery not only raised concern about identity theft but also posed broader societal risks, such as misinformation campaigns [5].

The landscape of spoofing attacks is now characterized by increasingly sophisticated methods that challenge the security and reliability of facial recognition systems. As these attacks become more advanced, the need for effective detection and countermeasures grows paramount.

4.2 Relevant Work

Based on the different types of spoofing attack, the countermeasures proposed in the past decade can be divided into several groups.Below, I discuss these groups.

4.2.1 Texture Based

Early detection methods mainly focused on analyzing skin textures to differentiate real and fake faces. Some of the techniques are Local Binary Patterns and Histogram of Oriented Gradients. LBP studies local pixel variations and generates unique texture maps for real faces, while HOG

captures essential contour information by analyzing gradients within localized regions of an image.Additionally, the Difference of Gaussian method has been used to highlight texture edges and variations, especially under different lighting conditions, making it effective for distinguishing between genuine and spoofed faces. For instance [6] Matti et al., proposed approach analyzes the texture of the images by using multi-scale local binary patterns and to classify they used a non-linear SVM classifier with radial basis function kernel.

4.2.2 Motion Based

In order to combat print and replay attacks, motion-based methods make use of involuntary facial movements like eye blinking and lip movements. Eye Blinking Detection operates on the principle that real faces exhibit spontaneous blinking patterns that static images and videos cannot reproduce, providing a reliable indication of liveliness. Lip Movement Analysis, in which natural lip motions are observed, further bolsters real-time spoof detection by identifying dynamic changes unachievable in a single frame replay.In [7], Anjos et al. proposed a method of anti-spoofing by analyzing the motion relationship between the background and the face region, which was tested on Replay-Attack database. The results shown that it can get a good performance on the photo attack.

4.2.3 Depth Based

Depth-based approaches utilize 3D information to detect the presence of a live, three-dimensional face as opposed to a flat image. These, employ depth cameras or Time-of-Flight sensors to capture depth maps which reveals the face's three-dimensional structure, making it difficult for attackers to succeed with printed images.The paper[8] describes a dual-cue approach for live face detection, utilizing both textual and depth features. The textual features are extracted from 2D facial image regions using a convolutional neural network . Depth representation, on the other hand, is obtained from images captured by a Kinect sensor.

4.2.4 Multi Spectral Based

To combat the complex spoofing attacks, multi spectral imaging techniques emerged. They capture skin reflections under varying light wavelengths. By making use of how a face responds to visible, infrared, and other light spectra, multi spectral methods reveal discrepancies which allow us to differentiate real from fake. In [9]Pavlidis et al, selected proper working spectrum such that the reflectance differences between genuine and fake faces increased.

5 Proposed Approach

5.1 Dataset Preparation

The dataset,I used in this project is CelebA-Spoof Dataset. This dataset is one of the most extensive and diverse datasets used in face spoofing domain.It has over 625,000 images belonging to more than 10,000 subjects. Some of the Key aspects of this dataset include diversity, covering a wide range of lighting conditions, ethnicity, age groups, and poses (e.g., forward-facing, backward, and angled views). Such diversity ensures that models trained on CelebA-Spoof are capable of handling real-world data, which is essential for practical applications of anti-spoofing systems.

Each image in the dataset has 43 attributes of which 40 describe about real (live) attributes of the image and the remaining 3 describe about the spoofing attributes. The live attributes include detailed annotations on features such as skin tone, eye shape, lips, and accessories like glasses and hats. The spoofed attributes include the type of spoof (e.g., photo, mask, or 3D model) and provide information on environmental factors such as lighting or the camera setup used for creating the spoof. Additionally, each image is accompanied by a bounding box file, which contains coordinates of the face within the image. This allows models to concentrate on the facial region, minimizing background interference, which acts as a noise.

Data preprocessing is a critical step in preparing the CelebA-Spoof dataset for training Convolutional Neural Network models.To enable effective facial feature extraction and reduce noise from background elements, the following functions were implemented:

The function `standard_width_height_scaling` initially scales each image to a uniform size of 224x224 pixels and adjusts the corresponding bounding box coordinates to match this scaled size. This scaling is essential because it standardizes the input dimensions across all images. By treating all faces uniformly in terms of size, it allows the model to focus exclusively on facial features, minimizing distractions from variations in image size.

The function `read_crop_img_with_bbox` is essential for extracting the region of interest from each image. It loads the image, applies the bounding box coordinates, and crops it to isolate the face.This focused cropping improves model accuracy by minimizing irrelevant background data which adds as noise and allowing it to concentrate on discriminative facial features necessary for spoof detection.

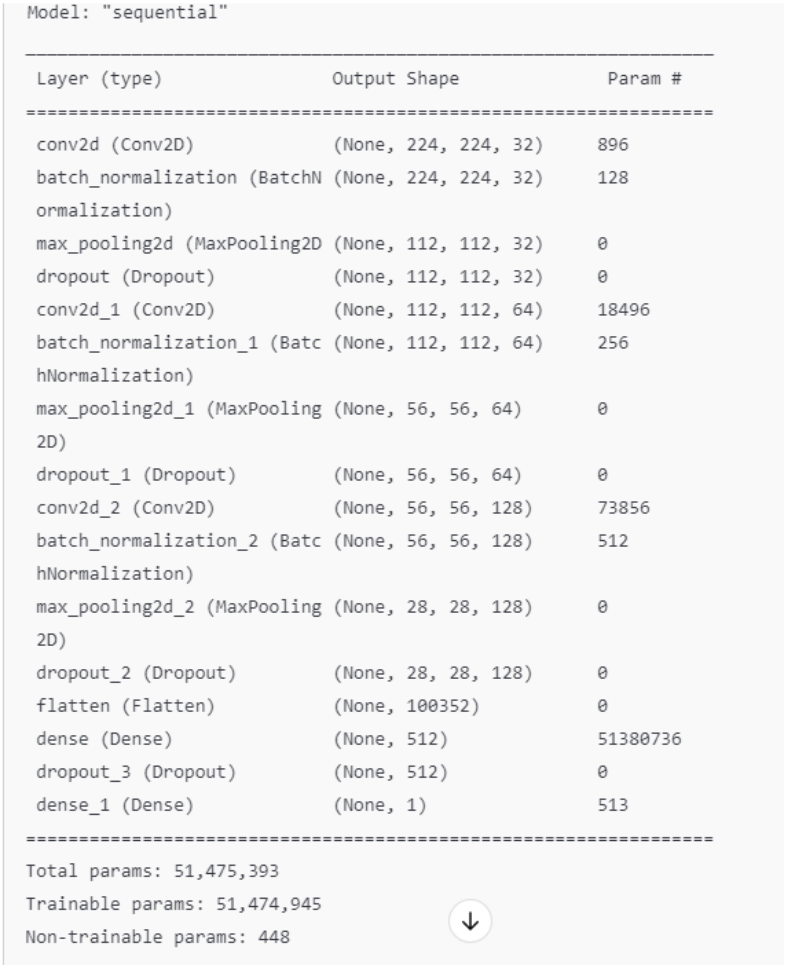
We split the dataset into training and testing sets, allocating 80% of the data for training and 20% for testing. The training data is further divided, with 80% used for training the model and 20% reserved for validation. This three-way split allows for effective training while providing a separate dataset for model validation and testing, ensuring the model's performance can be accurately assessed.

5.2 CNN Architecture and Training

I will be using CNN in this project because they are particularly effective for image-related tasks, especially facial recognition, due to their ability to automatically learn spatial hierarchies of features. They excel in capturing local patterns, allowing them to recognize intricate details in facial features that differentiate real faces from spoofed ones. Their architecture is designed to process pixel data directly, eliminating the need for manual feature extraction.

5.2.1 CNN Architecture

The proposed CNN model for this project consists of 16 layers in total, with specific types of layers designed to facilitate effective learning from the image data.The architecture is given in figure 1



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 224, 224, 32)      896
 batch_normalization (BatchN (None, 224, 224, 32)      128
 ormalization)
 max_pooling2d (MaxPooling2D (None, 112, 112, 32)      0
 dropout (Dropout)           (None, 112, 112, 32)      0
 conv2d_1 (Conv2D)           (None, 112, 112, 64)      18496
 batch_normalization_1 (Batc (None, 112, 112, 64)      256
 hNormalization)
 max_pooling2d_1 (MaxPooling (None, 56, 56, 64)        0
 2D)
 dropout_1 (Dropout)         (None, 56, 56, 64)        0
 conv2d_2 (Conv2D)           (None, 56, 56, 128)       73856
 batch_normalization_2 (Batc (None, 56, 56, 128)       512
 hNormalization)
 max_pooling2d_2 (MaxPooling (None, 28, 28, 128)       0
 2D)
 dropout_2 (Dropout)         (None, 28, 28, 128)       0
 flatten (Flatten)           (None, 100352)            0
 dense (Dense)               (None, 512)               51380736
 dropout_3 (Dropout)         (None, 512)               0
 dense_1 (Dense)             (None, 1)                 513
=================================================================
Total params: 51,475,393
Trainable params: 51,474,945
Non-trainable params: 448
```

Figure 1: CNN Architecture.

Input Shape

The dimensions of input images is of size 224x224x3 pixels. This size is manageable for computational efficiency while still retaining enough detail for effective facial recognition.

The model begins with convolutional layer and has three in total.These layers utilize filters to extract important features such as edges, textures and patterns from the input images which are crucial for differentiating between real and fake faces.

The window size for these filters is set to 3×3 which is optimal for capturing local features while maintaining computational efficiency. The number of filters doubles with each subsequent layer, increasing from 32 in the first layer to 128 in the last. This design choice allows the model to capture increasingly complex and abstract features as it processes the data deeper in the network. The activation function used for this layer is ReLU as it helps to introduce non-linearity into the model and allowing the model to learn complex patterns while avoiding issues like vanishing gradients.

Following each convolutional layer, batch normalization is applied. This layer normalizes the outputs, helping to stabilize and accelerate the training process by reducing internal covariate shifts. It allows the model to learn faster and can lead to improved overall performance.

Then comes max-pooling layers,they are incorporated to down-sample the feature maps generated by the convolutional layers. This reduces the dimensionality of the data, preserving essential features while minimizing the computational load. These layers use a window size of 2x2. They

also contribute to translational invariance, allowing the model to recognize faces in various positions.

To combat overfitting, four dropout layers are included.We drop 25 percent of neurons at each phase. These layers randomly deactivate a portion of the neurons during training, ensuring that the model does not become overly reliant on any single feature. This enhances the model's generalization capabilities and improves its performance on unseen data.

The model concludes with two dense layers, where the flattened outputs from the previous layers are processed to yield the final classification. The first dense layer employs the ReLU activation function, while the output layer uses the sigmoid activation function to produce probabilities for the two classes: real and fake. The sigmoid function is particularly suitable for binary-class classification tasks.

The output layer has one unit corresponding to the binary classification problem (real and fake faces).The sigmoid activation function ensures that the outputs represent the binary class probabilities, enabling the model to assign inputs to true class if the output is above 0.5 and spoof class if it is below 0.5.

5.2.2 Training the Model

The model is compiled using the Adam optimizer, which is known for its efficiency in training deep learning models.It combines the advantages of two popular optimization techniques: AdaGrad and RMSProp. Adam adjusts the learning rate for each parameter based on the first and second moments of the gradients, allowing for adaptive learning rates that can speed up convergence.

The loss function used is Binary Crossentropy, which is appropriate for the binary classification task. It measures the performance of the model by quantifying the difference between the predicted probability distribution and the true distribution.

During training, the model is fitted on the training dataset for 5 epochs allowing it to learn from the data while periodically validating its performance on a separate validation set. This validation helps monitor the model's accuracy and adjust the training process as needed to prevent overfitting. The final trained model is saved for future use in real-time analysis.

5.3 Patch CNN

Building on the CNN approach discussed in Section 5.2, we explore the application of a Patch-based CNN to improve face spoofing detection performance. The CNN model, while effective, faces limitations in capturing fine-grained facial details that are crucial for distinguishing between genuine and spoofed faces.

To address these challenges, we propose a patch-based approach, inspired by the methodology discussed in [10], which focuses on analyzing local regions of the image rather than the entire face. This allows the model to detect subtle variations that may otherwise be overlooked by a global CNN approach. This method involves dividing input images into smaller, manageable patches that are processed individually. A visual example of this patch-based approach is shown in Figure 2 below.

Below are the advantages of this approach:

Analyzing patches allows the model to concentrate on local features that are essential for distinguishing live from spoofed faces, capturing subtleties in facial regions that might indicate authenticity.

Each patch encompasses contextual information from different regions of the face, enhancing the model's ability to recognize patterns associated with genuine versus fake images.

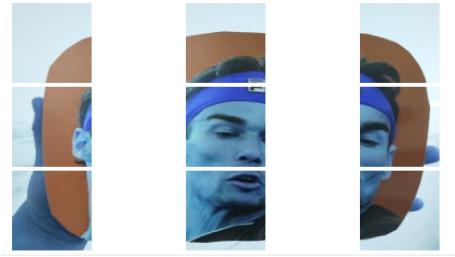


Figure 2: Example of an image divided into patches

Processing smaller patches decreases computational demands, making the approach efficient even with high-resolution images, which is crucial for real-time applications.

5.3.1 Patch CNN Architecture

The architecture for the Patch CNN model starts with an input shape of (9, 74, 74, 3), where the image is divided into 9 patches of size 74x74 pixels with 3 color channels (RGB). Each patch is then processed separately by the CNN model presented in section 5.2, which is applied using the TimeDistributed wrapper. This wrapper allows the CNN to operate independently on each patch, treating the patches as sequential data, while sharing the same parameters for all patches.The architecture for patch cnn is shown in Figure 3

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 9, 74, 74, 3)]    0

 time_distributed (TimeDistr (None, 9, 128)            254208
 ibuted)

 global_average_pooling1d (G (None, 128)               0
 lobalAveragePooling1D)

 dense_1 (Dense)             (None, 512)               66048

 dropout_1 (Dropout)         (None, 512)               0

 dense_2 (Dense)             (None, 1)                 513

=================================================================
Total params: 320,769
Trainable params: 320,449
Non-trainable params: 320
```

Figure 3: Patch CNN Architecture

After the patches are processed, their features are combined using GlobalAveragePooling1D, which pools the feature maps along the sequence dimension (across the patches). This pooling layer helps reduce the dimensionality of the feature maps while preserving important spatial information. The output is then passed through a dense layer with 512 units, using the ReLU activation function, followed by a dropout layer with a 50% dropout rate to prevent overfitting.

Finally, the output layer consists of a single neuron with a sigmoid activation function, making the final prediction of whether the face is real or spoofed. The sigmoid function outputs a prob-

ability value between 0 and 1, where values greater than 0.5 correspond to real faces and values below 0.5 correspond to spoofed faces.

5.3.2 Patch CNN Training

The Patch CNN model is compiled using the Adam optimizer, known for its efficiency in training deep learning models. Adam adapts the learning rate for each parameter based on the first and second moments of the gradients, allowing faster convergence.

For the loss function, we use binary crossentropy, which is suitable for binary classification tasks. It calculates the difference between the predicted probabilities and the true labels, providing a measure of the model's performance.

The model is trained for 5 epochs using the fit() function. This setup ensures that the model learns effectively from the training data while being evaluated on the validation set after each epoch to monitor its performance and adjust if necessary.

5.4 Webcam Integration

This section outlines the implementation of real-time face detection through webcam integration. The flowchart illustrating the implementation process is provided in the figure 4.

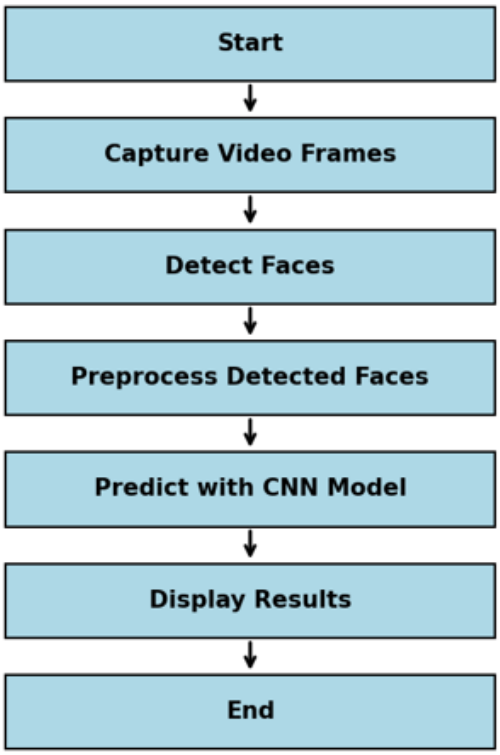


Figure 4: Flowchart for Webcam Intgration with Model

To start with, the system interfaces with the webcam, capturing continuous video feed. This involves extracting individual frames from the video stream for further processing.

Once the video frames are captured, the Haar Cascade classifiers are employed for face detection. These classifiers utilize pre trained cascade files to identify facial features.This step is essential because it provides the bounding box coordinates which are essential for preprocessing.

After detecting faces, preprocessing steps are taken to prepare the image for analysis. This involves all the preprocessing steps which are done for the training data.

The preprocessed data is then fed into the model for prediction.

Finally, the results of the detection process are visualized in real time. The system overlays the classification outcomes on the video feed, alongside the decision the probability metrics are also displayed.

6 Results

6.1 CNN Model

In this section, we present the outcomes obtained from training and evaluating the model on a dataset of 100,000 images, which included a balanced representation of 50,000 live and 50,000 spoof images. The model was assessed on its ability to distinguish between live and spoof faces, providing valuable insights into its effectiveness.

The CNN model achieved a remarkable accuracy of 91.76% on the test dataset, demonstrating its high proficiency in classifying live and spoof images. The evaluation metrics yielded a, Half Total Error Rate (HTER) of 0.08.

The confusion matrix further elucidates the model's performance, showcasing the distribution of predictions across the true classes as shown in the figure 5.

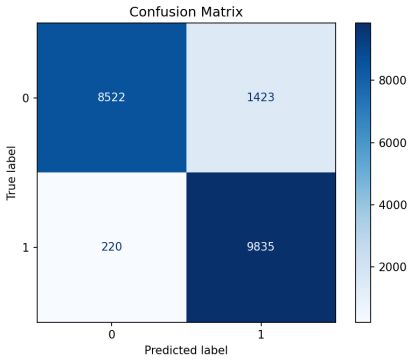


Figure 5: Confusion Matrix

The training loss per epoch and accuracy per epoch is depicted in the figure 6. This graph illustrates the model's convergence behaviour over the epochs, showcasing a decrease in loss training set.

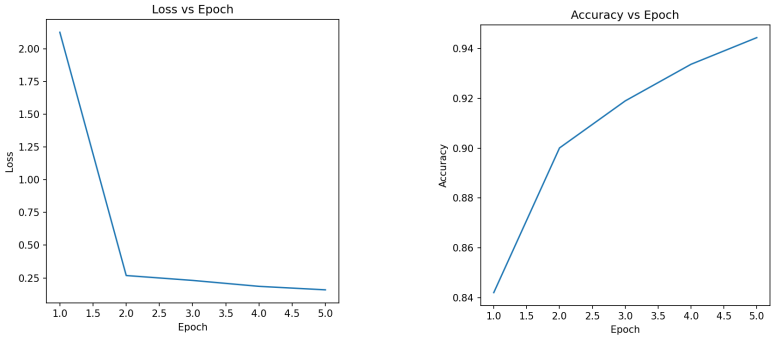


Figure 6: Loss vs epoch (left) and Accuracy vs epoch (right)

6.2 Patch CNN

ON the other hand the Patch CNN model achieved a remarkable accuracy of 93.45% on the test dataset, demonstrating its high proficiency in classifying live and spoof images. The evaluation metrics yielded a, Half Total Error Rate (HTER) of 0.05.

The confusion matrix further elucidates the model's performance, showcasing the distribution of predictions across the true classes as shown in the figure 7.

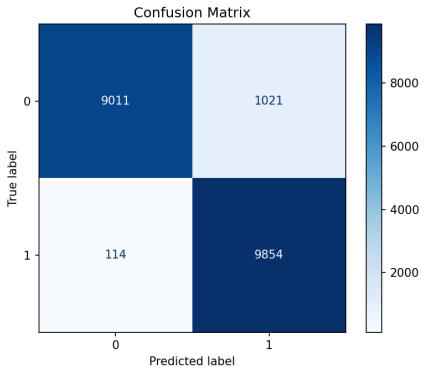


Figure 7: Confusion Matrix

The training loss per epoch and accuracy per epoch is depicted in the figure 8. This graph illustrates the model's convergence behaviour over the epochs, showcasing a decrease in loss training set.

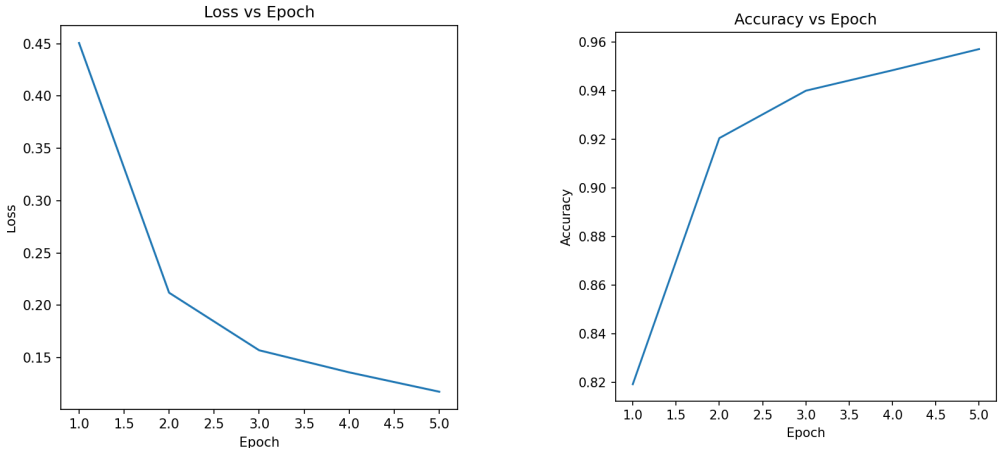


Figure 8: Loss vs epoch (left) and Accuracy vs epoch (right)

6.3 Comparing the models

In this subsection, we present a comparison of the performance of the proposed Patch CNN model with a traditional CNN model, evaluated on both the CelebA-Spoof and NUAA Impostor datasets.

The table 1 provides the results of both models evaluated on the CelebA-Spoof dataset, including metrics such as accuracy and HTER (Half Total Error Rate). The accuracy of each model indicates its ability to correctly classify faces as real or spoofed, while HTER gives a combined measure of both false acceptance and false rejection rates, which is crucial for evaluating spoof detection systems.

In the table 2, we present the accuracy of both models when tested on the NUAA Impostor dataset. The NUAA dataset contains a diverse set of face spoofing scenarios, including different types of spoofing attacks, making it a more challenging benchmark for evaluating spoof detection models.

Model	Accuracy (%)	HTER (%)
CNN	91.76	0.08
Patch CNN	93.75	0.05

Table 1: Performance of CNN vs Patch CNN on CelebA-Spoof Dataset

Model	Accuracy (%)
CNN	67.56
Patch CNN	75.81

Table 2: Performance of CNN vs Patch CNN on NUAA Impostor Dataset

6.4 Real time Integration

Additionally, I conducted real-time testing to assess the model's performance in practical scenarios. Below are the images from the test:

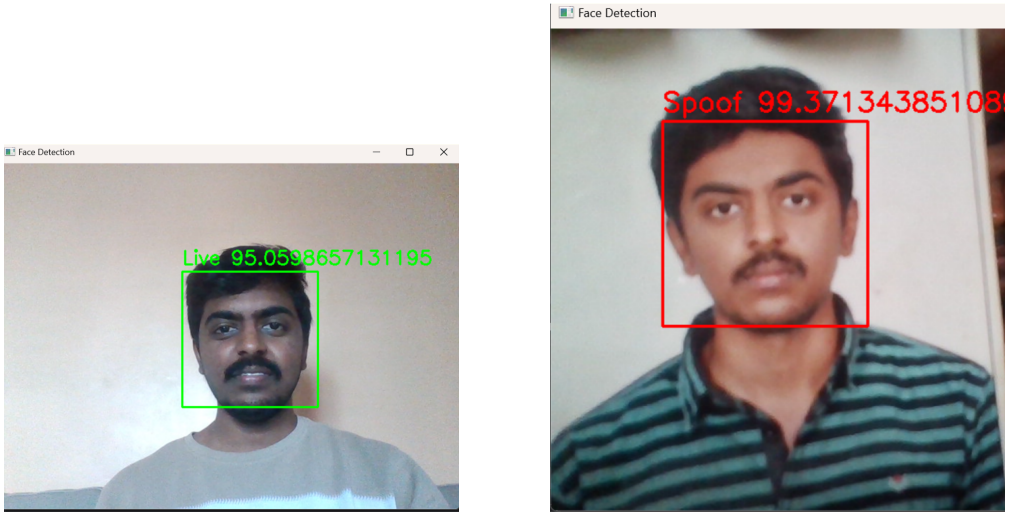


Figure 9: Live Image (left) and Spoof Image (right)

7 Conclusion and Future work

7.1 Conclusion

In this project, we thoroughly explored the critical need for face spoofing detection, highlighting the increasing prevalence of spoofing techniques and the vulnerabilities they expose in authentication systems. We discussed various detection techniques, such as motion based, multi spectral based and texture based techniques. However, we identified key limitations in these methods, which made them struggle to effectively differentiate between genuine and spoofed faces.

To address this challenge, we utilized the CelebA Spoof dataset, which provided a robust framework for developing our convolutional neural network (CNN) model. I successfully built and trained the CNN model with 100,000 images compromising 50,000 real and 50,000 spoof images, which demonstrated impressive performance. When evaluated, the model achieved an accuracy of 91.76%, indicating its strong ability to accurately classify live and spoof faces.

To further improve accuracy, we developed a Patch CNN model, which focuses on analyzing local regions of the input image. This approach enhanced the model's ability to detect fine-grained differences between real and spoofed faces, achieving an impressive accuracy of 93.75% on the CelebA Spoof dataset.

Moreover, I outlined a flowchart for the integration of our model into a real-time system, illustrating its practical application in real-world scenarios.

Additionally, I tested both CNN and Patch CNN models on the NUAA Impostor dataset to assess their generalizability. The CNN achieved an accuracy of 67.56%, while the Patch CNN demonstrated superior performance with an accuracy of 75.81%.

In conclusion, Patch CNN outperformed CNN in both real-time detection and dataset evaluation, demonstrating superior generalization and accuracy. Its ability to focus on local regions of the input image proved crucial in effectively distinguishing between genuine and spoofed faces. These results highlight the potential of Patch CNN as a reliable and robust solution for face spoofing detection.

7.2 Future Work

One major difficulty lay in designing the CNN architecture to effectively capture fine-grained facial details, crucial for differentiating between live and spoofed faces, while avoiding overfitting. To further optimize the model's performance, conducting experiments to fine-tune hyperparameters such as learning rates, batch sizes, and model architecture will be a key step. This process will help improve the model's accuracy and detection rates.

Next, a key step involves running the model on the entire CelebA-Spoof dataset, which contains approximately 625,000 images. Training on this complete dataset will likely improve model accuracy and robustness, but handling such a large dataset is a considerable challenge in terms of computational resources and time.

For future work, extending the system to process and analyze video streams is a critical step. This would enable the detection of temporal inconsistencies and dynamic facial cues, improving spoof detection accuracy. Furthermore, incorporating multimodal approaches, such as combining face texture analysis with other biometric features like voice, gaze tracking, or motion patterns, could enhance the robustness of the system and provide a more comprehensive solution for face spoofing detection.

References

[1] Jianwei Yang, Zhen Lei, and Stan Z. Li. Learn convolutional neural network for face anti-spoofing. *CoRR*, abs/1408.5601, 2014.

[2] Lei Li, Xiaoyi Feng, Zinelabidine Boulkenafet, Zhaoqiang Xia, Mingming Li, and Abdenour Hadid. An original face anti-spoofing approach using partial convolutional neural network. In *2016 Sixth International Conference on Image Processing Theory, Tools and Applications (IPTA)*, pages 1–6, 2016.

[3] Zitong Yu, Yunxiao Qin, Xiaobai Li, Chenxu Zhao, Zhen Lei, and Guoying Zhao. Deep learning for face anti-spoofing: A survey. *CoRR*, abs/2106.14948, 2021.

[4] Gustavo Botelho de Souza, João Paulo Papa, and Aparecido Nilceu Marana. On the learning of deep local features for robust face spoofing detection. In *2018 31st SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, pages 258–265, 2018.

[5] Pavel Korshunov and Sébastien Marcel. Deepfakes: a new threat to face recognition? assessment and detection. *CoRR*, abs/1812.08685, 2018.

[6] Jukka Komulainen, Abdenour Hadid, and Matti Pietikäinen. Face spoofing detection from single images using micro-texture analysis. 10 2011.

[7] André Anjos and Sébastien Marcel. Counter-measures to photo attacks in face recognition: A public database and a baseline. In *2011 International Joint Conference on Biometrics (IJCB)*, pages 1–7, 2011.

[8] Yan Wang, Fudong Nian, Teng Li, Zhijun Meng, and Kongqiao Wang. Robust face anti-spoofing with depth information. *Journal of Visual Communication and Image Representation*, 49:332–337, 2017.

[9] I. Pavlidis and P. Symosek. The imaging issue in an automatic face/disguise detection system. In *Proceedings IEEE Workshop on Computer Vision Beyond the Visible Spectrum: Methods and Applications (Cat. No.PR00640)*, pages 15–24, 2000.

[10] Yousef Atoum, Yaojie Liu, Amin Jourabloo, and Xiaoming Liu. Face anti-spoofing using patch and depth-based cnns. In *2017 IEEE International Joint Conference on Biometrics (IJCB)*, pages 319–328, 2017.