**Real-Time Face Detection System**

**Submitted for**

**Statistical Machine Learning CSET211**

Submitted by:

**(E23CSEU0433) Harshit**

**(E23CSEU0447) Khushi Garg**

**(E23CSEU0442) Aryan Chauhan**

Submitted to

**DR. NITIN ARVIND SHELKE (DR. SUSMITA DAS)**

**July-Dec 2024**

**SCHOOL OF COMPUTER SCIENCE AND ENGINEERING**

A close-up of a logo

Description automatically generated

**INDEX**

|  |  |  |
| --- | --- | --- |
| **Sr.No** | **Content** | **Page No** |
| **1.** | **Abstract** | **3** |
| **2.** | **Introduction** | **4** |
| **3.** | **Related Work** | **5** |
| **4.** | **Methodology** | **6** |
| **5.** | **Hardware/Software Required** | **7** |
| **6.** | **Experimental Results** | **8** |
| **7.** | **Conclusions** | **9** |
| **8.** | **Future Scope** | **10** |
| **9.** | **GitHub Link** | **11** |

1. **Abstract: This project implements a real-time face detection system using a Siamese Neural Network model for identity verification. The system collects images through a webcam, processes them, and uses them to train a Siamese Network that distinguishes between known and unknown faces. The network is trained with anchor, positive, and negative images to create embeddings, which are then compared using L1 distance. The model is evaluated based on metrics like precision and recall, enabling real-time verification of identity.**

**2. Introduction:** **Face detection is a fundamental technology in computer vision with significant applications across industries. This project aims to address the need for fast and reliable face detection in dynamic environments. By leveraging real-time image processing and machine learning techniques, the system enhances security measures and provides solutions for automated access control**.

**3. Related Work : Siamese Networks are a popular approach for face verification, as they excel in learning discriminative features between paired images. Compared to Convolutional Neural Networks (CNNs) alone, Siamese Networks compute similarity via shared parameters, optimizing memory and computational efficiency. In recent years, similar projects have utilized One-Shot Learning for identification with limited samples, making Siamese Networks an ideal choice for projects requiring robust, real-time verification. This project builds on the success of previous approaches by incorporating L1 distance as a similarity metric, which has shown promising results in facial recognition and verification tasks.**

**4.Methodology:**

1. **Data Collection:**

* **Images are collected from a webcam in real-time.**
* **The script captures anchor and positive images upon specific key presses.**
* **Anchor and positive directories store images from the user, while negative images are populated from a dataset of non-user faces.**

**2. Data Pre-processing:**

* **Images are resized to 100x100 pixels and normalized for consistency.**
* **Pre-processing functions ensure efficient storage and facilitate learning.**

**3. Model Architecture:**

* **The core architecture is a Siamese Network with a shared embedding layer.**
* **The embedding function consists of several convolutional and pooling layers, followed by a dense layer that outputs a 4096-dimensional embedding.**
* **The model calculates the L1 distance between embeddings of anchor and validation images, determining similarity based on this distance.**

**4. Training:**

* **The model is trained on paired images, where each pair contains either an anchor-positive or anchor-negative combination.**
* **A binary cross-entropy loss function is used, with the Adam optimizer to adjust weights.**
* **Training data is shuffled and split into training and testing sets with a 70-30 ratio.**

**5 Evaluation:**

* **The model's accuracy is evaluated using metrics like recall and precision on the testing dataset.**

**5.Hardware/Software Required:** **This project uses Python, OpenCV, TensorFlow, and a GPU-compatible device for real-time processing. A camera for video input is essential, and a high-performance computer ensures smooth and quick operations.**

**- Hardware:   
 - Computer with at least 4GB RAM  
 - Webcam for video capture**

**- Software:  
 - Python (version 3.7 or above)  
 - OpenCV library  
 - IDE (e.g., PyCharm, Jupyter Notebook)  
 - Git for version control**

**- Python libraries: cv2, num-py, matplotlib, and tensorflow.keras.**

**6. Experimental Results: The model was trained for 50 epochs, demonstrating significant precision and recall, confirming its effectiveness for face verification. During testing, the model consistently identified anchor-positive pairs and rejected anchor-negative pairs, achieving high accuracy. The system was also able to verify identities in real-time when integrated with a webcam. Precision and recall scores showed that the Siamese Network effectively minimizes false positives, enhancing the reliability of the system.**

**7. Conclusions: This real-time face detection system successfully implemented a Siamese Network for facial verification with strong accuracy. By leveraging image pairs, it was able to learn distinctive features, effectively differentiating between individuals. The model's performance demonstrated the Siamese Network's capacity to handle real-time face verification, making it a promising solution for secure authentication in applications like access control and identity verification.**

**8. Future Scope:** **Future enhancements can involve:**

* **Improving Model Robustness: Incorporating more diverse images, including different lighting conditions, angles, and occlusions.**
* **Deployment: Implementing the model on a mobile or edge device to make it more accessible and reduce the dependency on high-end hardware.**
* **Automated Data Collection: Introducing automatic data augmentation to increase the training dataset's diversity.**
* **Enhancing Accuracy: Experimenting with other similarity metrics or deeper network architectures to improve precision and recall.**
* **User Interface Development: Creating a user-friendly interface for easier interaction and deployment in real-world settings.**

**9.GitHub Link of Your Complete Project:**

**Link :-**