IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

AN AUTOMATED FRAUD DETECTION IN EXAMINATION FROM HALL TICKET

Chithra S Ravi, Suma S G, Mathu Uthaman, Namitha T N

Assistant Professor, Assistant Professor, Assistant Professor, Assistant Professor
Computer Science and Engineering,
Mahaguru Institute of Technology, Alappuzha, India

Abstract: Bougs hall ticket prevision consists of a web application where the invigilator uploads the image of the hall ticket. The hall ticket is send to the backend of the web application where the text from hall ticket is infusion using OCR(Pytesseract) and face recognitionof the student is done by using technquies like Face Recognition Algorithms like (deepface) facenet etc. A database is there to store the information of the students. Details as well as photos are compared and result is sent to the frontend and the invigilator could separate the student as real or fake.

Keywords **- OCR, MTCNN,Face recognition algorithms**

I. INTRODUCTION

The rise of online events and examinations has created a new set of challenges for organizers, participants, and institutions. One of the most pressing challenges is the need to verify the authenticity of hall tickets. In order to prevent fraud and ensure the security of participants, it is crucial to have a reliable, efficient, and secure method of verifying hall tickets. Traditional methods of verifying hall tickets, such as manual inspection or barcode scanning, can be time-consuming, error-prone, and susceptible to fraud. To address these challenges, we propose building a web application that leverages advanced image recognition and OCR technologies to verify hall tickets quickly and accurately.Our proposed web application uses state-of-the-art machine learning models such as MTCNN and Inception ResNet to extract and verify the photo on a given hall ticket. By comparing the extracted face to photos in a database, the application can accurately identify if the person presenting the hall ticket is the same as the one on the photo.

1. 1 MITCNN

MTCNN (Multi-Task Cascaded Convolutional Networks) is a deep learning model for facedetection and recognition developed by researchers at the University of Science and Technology of China. It is a multi-task learning framework that simultaneously detects faces, extracts facial landmarks, and estimates the bounding boxes of detected faces. MTCNN consists of three cascaded stages, each of which uses a deep convolutional neural network (CNN) to perform a specific task.. The first stage uses a proposal network to generate candidate regions containing faces. The second stage refines the bounding boxes of the candidate regions and eliminates false positives. Finally, the third stage extracts facial landmarks and aligns the detected faces. MTCNN uses a three-stage hierarchical approach to achieve high accuracy in face detection. The first stage proposes candidate windows using a CNN that is trained to differentiate between faces and non-faces. The second stage refines the candidate windows to accurately localize the faces in the image. Finally, the third stage uses a CNN to extract facial landmarks and align the faces. MTCNN has several advantages over other face detection models. First, it can detect faces of varying sizes and orientations, making it suitable for use in a wide range of applications. Second, it is highly accurate, witha low false positive rate, making it ideal for use in applications that require high precision.

1.2 Inception Resnet

InceptionResNet consists of a series of inception and residual blocks, each of which is designed to extract and learn features from the input image. Inception blocks use a set of parallel convolutional layers of varying kernel sizes to capture features at different scales. . Finally, it is relatively easy to train and fine-tune, making it suitable for a wide range of applications. In our proposed web application for verifying hall tickets, InceptionResNet is used to compare the extracted face from the hall ticket to photos in a database. The database contains photos of individuals who are authorized to use the hall ticket, and the InceptionResNet model is used to compare the features of the extracted face to those of the authorized users. If the features of the extracted face match those of an authorized user, the application can verify the authenticity of the hall ticket.

3.Pytesseract

Pytesseract is a Python wrapper for Tesseract-OCR, an optical character recognition (OCR) engine developed by Google. Tesseract-OCR is designed to recognize text in images and convert it to machine- readable text. Pytesseract makes it easy to use Tesseract-OCR in Python applications, allowing developers to extract text from images and use it in a wide range of applications.

Pytesseract is easy to install and use. It can be installed using pip, the Python package installer, and is available for Windows, macOS, and Linux. Once installed, Pytesseract can be used to extract text from images in a few lines of Python code.

Preprocessing of images: Before using Pytesseract to extract text from an image, it is important to preprocess the image to improve the accuracy of the OCR engine. This may involve resizing the image, enhancing the contrast, or removing noise.

Language support: Tesseract-OCR supports a wide range of languages, and Pytesseract makes it easy to specify the language to use for OCR. However, it is important to ensure that the correct language is used for the image being processed, as this can significantly affect the accuracy of the OCR.

Accuracy: The accuracy of OCR engines can vary depending on the quality of the image being processed, the complexity of the text, and other factors. It is important to test Pytesseract on a range of images to determine its accuracy and adjust the preprocessing and other parameters as needed. Integration with other libraries: Pytesseract can be integrated with other Python libraries for image processing and analysis, such as OpenCV, PIL, and scikit-image, to enhance the accuracy and efficiency

In our proposed web application for verifying hall tickets, Pytesseract is used to extract text from the hall ticket, such as the name and registration number of the candidate. The extracted text is then compared to the data stored in the database to verify the authenticity of the hall ticket. By using Pytesseract in combination with MTCNN and InceptionResNet, we can create a fast and accurate method for verifying the authenticity of hall tickets, making online events and examinations more secure and reliable.

II. SYSTEM ANALYSIS

FACE-RECOGNITION MODULE

Data Collection: The first step in training a face recognition model is to collect a dataset of face images. This dataset should be diverse and representative of the target population, with enough variation in terms of facial expressions, lighting conditions, and poses.

Data Preprocessing: Before training the model, the face images must be preprocessed to remove any noise or artifacts that might affect the model's performance. This could include tasks like face alignment, cropping, and normalization to ensure that all face images have the

same size and orientation.

Feature Extraction: The next step is to extract meaningful features from the face images. This is typically done using a deep neural network that has been pre-trained on a large dataset of images, such as the Inception ResNet model. The pre-trained model can be fine-tuned on the face recognition task by replacing the last layer(s) of the network with new layers that are specific to the task at hand.

Training: The model is trained on the preprocessed face images using a supervised learning approach. During training, the model is presented with pairs of face images and one must learn to distinguish between them. The

loss function used during training typically takes into account the similarity between the feature vectors of the pairs of faces, and the model adjusts its weights and biases to minimize the loss.

Evaluation: Once the model is trained, it is evaluated on a held-out set of face images to assess its performance.It involvescalculating its precision and recall, or comparing its performance to other state-of-the-art models.

Deployment: After the model is trained and evaluated, it can is saved as .pt file and the image is passed for the recognition.

of text extraction from images.

4. Compatibility with different image formats: Pytesseract can extract text from a wide range of image formats, including JPEG, PNG, BMP, and GIF, making it a versatile tool for OCR.

TEXT-RECOGNITION MODULE

The process started with the image being loaded and converted into grayscale using OpenCV. This was essential as the OCR accuracy significantly increases with grayscale images, since they reduce complexity by eliminating color data.

OCR was then applied to this image via PyTesseract, a Python wrapper for Google's Tesseract- OCR Engine, to extract textual data. The text obtained from this process contained all the written information from the image, including the details such as student's name, registration number, branch, and examination center.

To separate and identify these details from the overall text, Python's Regular Expressions (re) library was used. Regex patterns were created for each piece of information to be extracted. These patterns depended on the specific format in which these details were printed on the hall tickets. The search function was then used to locate and extract the matching patterns within the text.

The extracted details were then ready for comparison with the database records. For this, we utilized Python's SQLite3 module to connect with the SQLite database. SQLite was selected for its lightweight and serverless nature, allowing for simple integration with the Python environment.

The key to the success of this method lies in the accuracy of the OCR and the precision of the regular expressions. However, the variability in image quality and text complexity of hall tickets might affect the OCR performance. Therefore, it is suggested to further explore image preprocessing techniques such as noise reduction, skew correction, or binarization to improve the OCR results in future work.

DATABASE MODULE

Creating the database and table:

The first part of the process is creating a SQLite database and a table named "students" if it doesnot already exist. We start by importing the sqlite3 module which allows Python to interact with SQLite databases.

We then connect to the SQLite database named "student.sqlite" by calling sqlite3.connect() function. This function will create the database file if it does not already exist. function. This function will create the database file if it does not already exist.

FACULTY MODULE

The faculty has the option to login to his account using the username and password. After logging in , faculty could upload the hall ticket photos and get the predictions. If the hall ticket is real, details of student is shown. If the hall ticket is fake, Status fake is displayed.

User Login Authentication: This feature allows users to enter their credentials (username and password) to gain access to the system. It verifies if the entered credentials match with the stored credentials in the application. If the credentials are valid, it allows the user to access the system. If they are invalid, it displays an authentication failure message.

Face Recognition: This feature enables facial recognition by comparing a provided image to a database of known faces. When an image is uploaded, it's processed and compared to known faces in a database. It uses a pretrained model, which generates an embedding (a numerical representation) of the face in the image and compares it to the embeddings of known faces in the database.

Student Details Extraction: Once a face has been recognized, this feature retrieves the details of the recognized student from hall ticket. It is compared with details in the student sqlite database. It uses the recognized name as a key to fetch details such as the registration number, branch, and center from the database.

Face Verification: The system verifies if the recognized face from the image matches with the stored details. If the facial recognition score is below a predefined threshold, it determines that the face in the image is a match. If the score is above the threshold, it determines that the face is not a match.

Result Display: Based on the verification result, the system renders a result page. If the face is verified, it displays the student's details, including name, registration number, branch, and center. If the face is not verified, it displays a message indicating that the verification was unsuccessful.

TECHNOLOGY USED

Python: It's a high-level, interpreted programming language with dynamic semantics. Python's simple, easy-to-learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse.

Base64: Base64 is a binary-to-text encoding scheme. It's used when there is a need to encode binary data, especially when that data needs to be stored and transferred over media that is designed to deal with text. In Python, the base64 library is used to encode and decode data.

OpenCV (CV2): OpenCV (Open Source Computer Vision Library) is an open-source computer vision and machine learning software library. It has more than 2500 optimized algorithms, which can be used for a wide range of tasks, such as detecting and recognizing faces, identifying objects, classifying human actions in videos, extracting 3D models of objects, and much more.

PyTesseract: PyTesseract is a Python wrapper for Google's Tesseract-OCR Engine. Optical Character Recognition, or OCR, is a technology used to convert different types of documents, such as scanned paper documents, PDF files or images captured by a digital camera into editable and searchable data.

SQLite3: SQLite is a C library that provides a lightweight disk-based database that doesn't require a separate server process and allows accessing the database using a nonstandard variant of the SQL query language. sqlite3 is a Python library for SQLite databases. It provides an SQL interface compliant with the DB-API 2.0 specification.

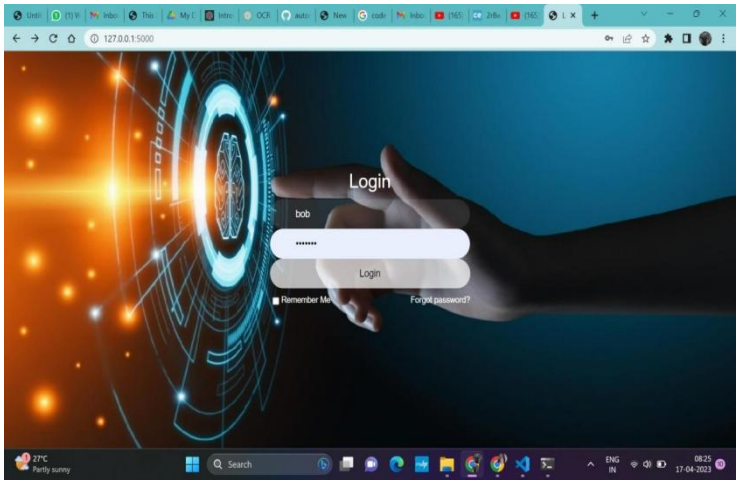
IV TESTING

Unit Testing: Each individual function should be tested to ensure it behaves as expected under a variety of conditions. Python's built-in unit test framework can be used to write these tests. For example, functions like face_match and get_student_details can be tested to ensure they return the expected results.

Integration Testing: After individual units are tested, they should be tested together to ensurethey interact correctly. This is particularly important for functions that call other functions or interact with external systems like databases.

Functional Testing: This involves testing the application against the functional requirements and specifications. It is conducted to ensure that the system accomplishes its intended functions and features.

V. RESULTS



VI CONCLUSION

In conclusion, the web application to verify hall tickets using MCTNN and Inception ResNet for face recognition and PyTesseract for text recognition offers an effective solution to enhance the security and authenticity of online events and examinations. By employing deeplearning techniques, the proposed system achieves high accuracy in face and text recognition, thereby reducing the likelihood of fraudulent activities. The system also enables real-time verification, which is critical for ensuring the timely and efficient execution of online events and examinations.

REFERENCES

[1]. "Real-time automatic identification document recognition using deep learning" by M. Nishida et al.

[2]. "OCR-based passport recognition system using CNN" by S. B. Cho et al. 3. 3."Document analysisfor passport recognition using deep learning techniques" by P. Gupta et al.

[3]. "Document image analysis using deep learning: a review" by S. Bhowmik.

[4]. "Deep face recognition using deep learning framework" by H. Wang et al.