OSINT-Based Tool for Social Media User Impersonation Detection Through Machine Learning

1st Rajaa Alqudah
King Abdullah II School of Engineering
Princess Sumaya University for Technology
Amman, Jordan
r.alqudah@psut.edu.jo

2nd Mohammed Al-Qaisi
King Abdullah II School of Engineering
Princess Sumaya University for Technology
Amman, Jordan
mohammed.w.alqaisi@gmail.com

3rd Rakan Ammari
King Abdullah II School of Engineering
Princess Sumaya University for Technology
Amman, Jordan
rak20170685@std.psut.edu.jo

4th Yazan Abu Ta'a
King Abdullah II School of Engineering
Princess Sumaya University for Technology
Amman, Jordan
yaz20170254@std.psut.edu.jo

2023 International Conference on Information Technology (ICIT) | 979-8-3503-2006-0/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICIT58056.2023.10226010

***Abstract*—With the explosion of social media platform usage in recent years, privacy and online security have become major concerns. Malicious users create fake social media profiles, posing as regular people or public figures to gather personal information, damage reputations, or show off their social engineering skills. As a result, social media platforms, such as Facebook, Instagram, and others, provide the perfect place for identity theft. Malicious users can create fake social media profiles with the same name and profile image as another person.**

In this paper, Open Source Intelligence (OSINT) platform is used to collect and analyze publicly available information to identify potential impersonators. The proposed approach employs web scraping, machine learning, and web development modules in Python and can be hosted on the AWS cloud for optimal performance and scalability. It accurately scrapes social media platforms (Facebook and Instagram) and presents potential impostor profiles based on a user's uploaded photo only, or the photo with a name, or the photo with the name and a username. The proposed model achieved an accuracy of 88% with precision and recall matrices of 86% and 89% respectively.

***Index Terms*—Social Media, OSINT, Privacy, Machine learning, Face recognition**

I. Introduction

In today's era, social media has become a crucial part of people's lives. People rely on social media platforms for all sorts of communication for either business or personal purposes. Whether a social media profile belongs to the person presented in it can never be known. This problem is caused due to social media impersonators who mimic legitimate profiles and can have a severe impact on the reputation of the individual they are targeting. Unaware organizations can also be greatly affected by impersonators. Users can go ahead and create an identical social media page to harm the reputation of the targeted organization.

Open Source Intelligence (OSINT) [1] is a technique for collecting and analyzing information from publicly available sources. Many techniques are utilized in OSINT including search engines, public records, news sources, and data analysis. This framework lacks a tool that specifically targets social media impersonators, and social media platforms lack detection algorithms against such frauds.

Artificial Neural Networks (ANNs) are machine learning schemes that are inspired by biological nervous systems. These networks are built from many interconnected nodes known as neurons. A basic ANN consists of three basic layers: the input, hidden, and output layers. The way the layers work together is by first loading the input as a multidimensional vector from the input layer and then distributing it to the hidden layers. These hidden layers are responsible for assessing how a stochastic change inside itself affects or improves the final output, based on the decisions made by the prior layer. Then the ANN continues to train by repeating this process for every hidden layer until a well-trained model is achieved. An ANN training process where many hidden layers are involved is known as a deep learning process [2]. Convolutional Neural Networks (CNNs) are similar to ANNs, as they have the same basic structure, but were built for a different purpose. They were introduced from the study of the brain's visual cortex to be used in image recognition applications.

This paper proposed a framework to overcome the concerns that people and organizations may have about the misuse of their images or their brand identity images. The framework incorporates OSINT, machine learning, and web scraping automation and uses the Python programming language to achieve the goals. The purpose of the framework is to allow people to check if there are social media accounts that are impersonating them, or in other words, using their image and/or name. This is achieved by taking an image and name as inputs from the user and then using web scraping to collect all social media accounts that have matching names. Then, using a machine learning algorithm, check if any of those found profiles have a profile picture that contains a matching face to the user's face.

The framework includes two main parts: web scrapers

to gather the information needed to get an accurate result and machine learning modules which are used for image search and to enhance the speed of the system. The tool was developed that was hosted on the cloud using a Windows server and has the specifics of 4 processors, 4 GB RAM, and 30 GB file system storage.

The tool will be triggered when a user selects a name and picture as input to the web page, this mechanism is called reverse image search and the only limitation is if the model has not trained on the input picture, it will display to the user to wait for the web scraper. After the web scraper finishes gathering the information, the user needs to verify their account and by that, it is decided that the other accounts are the impersonators of the user. Fig. 1 below is a high view of the system.

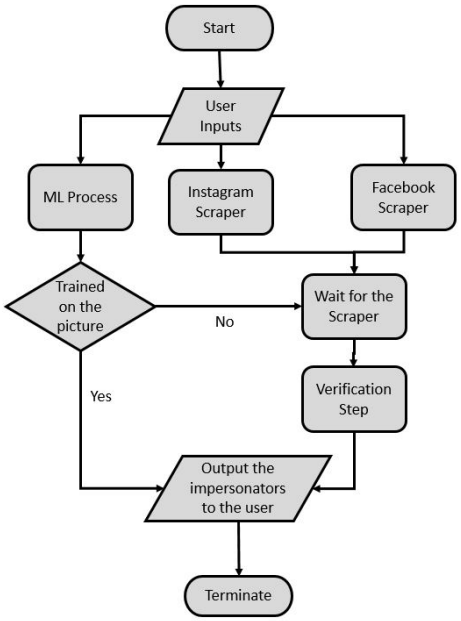

Fig. 1: System Overview

The rest of the paper is organized as follows: Section II presents the related literature and background. The proposed design is discussed in section III while the results are presented and analyzed in section IV. Finally, section V concludes the paper with a summary of the findings and future work.

II. Literature Review

OSINT concept is not new, it has been introduced years ago to collect information from publicly available sources [1], [3]. With the emergence of social media, big data, and machine learning in recent years, many studies discussed the OSINT framework in the new era [4]–[9]. The authors of [8] described the state of OSINT with a focus on the cyber security angle. On the other hand, the authors of [9] list OSINT's issues and the impact of OSINT trends throughout the years. Examples of these issues are great quantities of accessible information on the web, information contradiction, and the manipulative character of publicly available information to mention some.

EagleEye [10] is an open-source Python-based tool that needs at least one image of a person and their name. The tool then uses the name to find potential Facebook and Instagram profiles. Then, using face recognition, the tool tries to find the actual Facebook and Instagram profiles of the person you are looking for.

PimEyes [11] is a proprietary web-based application that takes an image only as input and uses reverse image search and web scraping techniques. PimEyes does not search social media platforms and requires a subscription to show the results.

Social Mapper [12] on the other hand takes images and usernames as input and searches social media websites only. The output will be in CSV file format that lists all the matched social media profiles. Social Mapper is considered slow compared with the other tools.

To be able to distinguish this work from others', table I compares the proposed approach with other tools like Eagle-Eye, PimEyes, and Social Mapper.

Biometric data, unlike passwords, PINs, cards, and keys, cannot be forgotten, stolen, or lost. Many studies focus on face recognition techniques and more specifically in answering one question, given an image of a face, how can its identity be determined? [13], [14]. Face recognition is an evolving field, in which there are still challenges to be overcome. In recent years, the field has received a great deal of attention and development. Face recognition is done using the FAREC Methodology which consists of four main stages: face detection, face alignment, face cropping, and feature extraction [15].

Web scrapers are tools that simulate human behavior in accessing web pages either using low-level HTTP or by launching and simulating the usage of a web browser, such as Google Chrome [16], [17]. In terms of legality, the authors of [18] give different cases that have occurred in the past like eBay's lawsuit against Bidder's Edge for their use of bots.

In today's era, cloud computing has been essential for the IT industries all over the world. It consists of hardware, software, and internet infrastructure which makes the most powerful architecture of computation. In [19], the authors refer to the term cloud in cloud computing as the collection of networks and services where the user can benefit from cloud computing power whenever demanded. Then it specifies the three services provided by the cloud computing providers as follows: Software as a Service (SaaS), Platform as a Service, and Infrastructure as a Service.

III. Proposed Design

The proposed approach is to open a social media platform's website and perform a name search. After successfully obtaining a list of profiles, a Python package called Urllib [20] was used to scrape this list. Urllib contains multiple modules used for working with URLs, such as opening, reading, parsing URLs, and exception handling. Urllib is used to download images into a folder that will be used later for the machine learning part and face matching.

Tools	Highlights	Limits
EagleEye [10]	Takes images and usernames as input. Uses the face-recognition Python module. Outputs the links of user's social media accounts.	Searches only Facebook and Instagram. Performs public Facebook search, which yields extremely inaccurate results. Has no form of caching or model training for performed searches.
PimEyes [11]	Takes an image only as input. Proprietary web-based application. Uses reverse image search and web scraping techniques. Accurate and ends up listing most websites where the face in the input image is recognized.	Does not search social media platforms. Requires a subscription to show results.
Social Mapper [12]	Takes images and usernames as input. Searches social media websites only. Outputs a CSV file listing all the matched social media profiles.	Built-in web scraper techniques are detected by social media platforms. Slow.
Proposed Tool	Takes an image, and optionally a name and a username as input Use face-recognition Python module and a trained ML model. Outputs links of impersonating accounts of the individual recognized in the input image.	May produce false positives due to the lack of public images. Limited to only social media platforms.

TABLE I: Comparing the proposed approach with other available tools

For machine learning, Keras model was used to process, train and identify the images extracted from the social network. The model has been trained on all the images inside the directory dataset and exported the features needed to compare it with the user image. The same feature extraction process is applied to the user image, and the model will predict which one of the images is like the input. The model produces a confidence score of the predicted result.

Flask was used as the web framework due to its fast and straightforward implementation. Flask handles HTTP requests and responses as Python code. Flask is used to handle all the web server's routes and to redirect the users as the code executes.

Machine learning algorithm and web scraping code were imported to the Flask project and then deployed on AWS using the Elastic Beanstalk service as an EC2 instance. Elastic Beanstalk is a service that is used to deploy web applications and automate the deployment process by setting up the EC2 instance automatically and installing the virtual environment. CI/CD was integrated to make the workflow more efficient and faster and to connect the project's repository to the AWS CodePipeline service so that each commit applied to the repository is automatically updated on the web server.

The flow starts by retrieving the user input image to test it in the machine learning model, then as optional, prompts the user to enter their full name and/or username. Depending on the machine learning model's output, if there is a match, the output will be the name of the user on which the model has previously trained on; this output name is then sent to the web scraper to find all the profiles that match that name and image. On the other hand, if no match was found, the tool will ask the user to input their name and/or username to generate better results. After the web scraper is done retrieving all the profiles, a one-to-one image comparison is performed on all the profile pictures to eliminate the least relevant profiles, thus improving the speed and accuracy of the final result overall. The final verification step is just to help the tool make it clear to the user that the output profiles are indeed impersonators. Fig. 2 summarizes the phases of the flow.

A. Phase One: User inputs phase

This phase describes the inputs required for the tool to start the image search. An Image and an individual name have to be given, and the username is optional. The tool starts by checking the input image with the trained model that has already been trained. If the image was found by the model the tool will retrieve the name and insert it as an input to the web scraper, if the model couldn't recognize the person, then the user is asked to enter a name for the second phase.

B. Phase Two: Images Retrieving phase

With the help of web scraping techniques, a list of possible social media accounts images will be generated and stored in two folders; a Profile Picture folder which stores all the users' profile pictures, and an images folder to store all the images that will be used to train the model.

C. Phase Three: Image comparison phase

An image comparison on the profile pictures folder against the input image is held using a face recognition library in Python. The comparison is conducted by converting the images into a set of values and if the subtracted value is lower than a fixed threshold, then these images are considered a matching picture of the input's image. The last step in this phase is to match all pictures that have passed the threshold test with their accounts and store this information in a list. This phase was developed to lower the false-positive results that are coming from the web scrapers. A picture with no human face on it will be removed.

D. Phase Four: User's verification phase

Until the third phase, the tool would not know which accounts on the list are impersonating the user. Hence, a user's verification stage came in handy to ask the user which one of the presented accounts is yours. The user will be able to verify their accounts among the list and all other accounts are listed as impersonators for that user. Lastly, the tool will store this state, the profiles pictures folder, and the images folder in a file system and show it as a result.

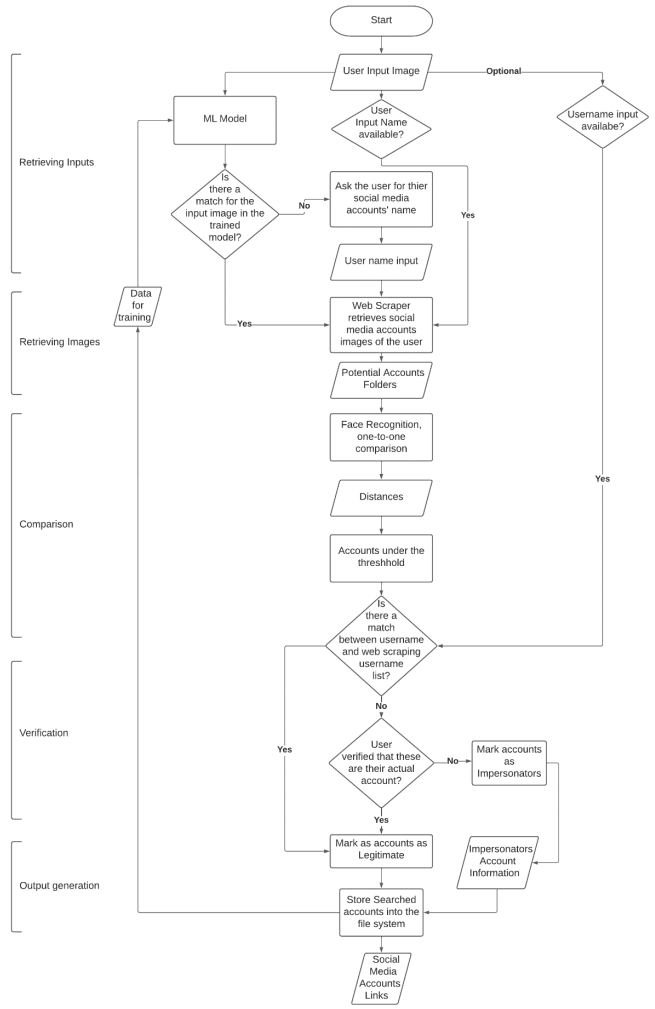

Fig. 2: Flow diagram of the system with phases

E. Phase Five: Output generation phase

Before outputting the result to the user, the images folder generated through the web scraping phase must be trained using the model. So, if any similar image is used as an input the result should be presented immediately. In the end, the tool should be able to present the verified social media account and its account impersonators.

F. Phase Six: Machine Learning Training

Every time the users use the tool and after presenting the impersonators to them, all the public photos from the web scraping search will be downloaded into a folder and a scheduled training process will be conducted at a predefined cadence to retrain on the new pictures.

Multi-task Cascaded Convolutional Networks (MTCNN) is a neural network that detects faces and facial landmarks in images. It is one of the most popular and most accurate face detection tools today [21]. It consists of 3 neural networks connected in a cascade and performs three tasks: Face classification, Bounding Box Regression, and Facial Landmark localization. Face classification is a binary classification problem to decide whether there is a face in the picture or not. The Bounding Box Regression helps calculates the offset between the picture and the nearest ground truth. The Facial Landmark localization is formulated as a regression problem and it searches for the five facial Landmarks: left eye, right eye, nose, left and right mouth corners. Facenet_keras is a one-shot learning model. It fetches 128 vector embedding as a feature extractor. It consists of good accuracy even for such situations.

SVM (Support Vector Machine) creates an optimal hyperplane to classify the classes of training datasets based on the different features of the face. The dimensionality of the hyperplane is one less than the number of features. Different kernels can be applied to see what features are used by the classifier to remove the features if required. This can help to improve speed. On the other hand, the Face recognition algorithm consists of three stages: Face detection, Feature extraction, and Feature classification. Face detection is used to find the faces present in the image, extract the faces, and display them. Feature extraction extracts the biological components of your face. These biological components are the features of your face that differ from person to person. Feature classification is a geometry-based or template-based algorithm used to classify the features of the test data among different classes of facial features in the training data. The algorithm detects the face inside the image using MTCNN, then it uses Facenet_keras to extract the features from the image, and finally, it uses the SVM classification model to train the data among the different classes.

IV. Results and Analysis

This section will discuss the results generated after running different test cases.

A. Prototype setup

Two web scrapers were used, Facebook Scraper and Instagram Scraper. These two scarpers were chosen based on their popularity and image-sharing capabilities [22].

Facebook Scraper relies on two functions, getFBProfilePictures(), which searches for a name on Facebook and downloads the profile pictures of all the results, and getFBPictures(), which goes through all public profiles resulting by the first function and downloads their publicly uploaded images, which are then stored in the folder that will be sent for the ML model for training.

For Face Recognition, a Python library based on Dlib was used. Starting with the function matchFBProfile(), it takes all the profile pictures downloaded, and the user input image and creates an encoding for each one, after that, it calculates a metric called face_distance, so if this metric is below a predefined threshold, the system considers these two pictures as a match but if the metric is over the threshold there will be no match. The other function is splitImageIntoFaces() which is used to overcome the limitation of the ML model where it

can only train on one person per picture, so each downloaded picture goes through this function and if there's more than one face in the picture, it detects them, and each face is placed in its own image, and stored in a folder which is ready to be sent to the ML training server.

B. Experiment result and discussion

1) Case one: Input Image only: If the person has been trained on previously, the ML model will output the person's name. This name can then be used as input to the web scraper to scrape the social media platforms.

After the scraping is complete, one-to-one matching between the input image and the profile pictures of the scraped accounts will take place to finally output all profiles with a matching image. This output is shown to the user and the user is asked to verify which of them is actually the user. The other profiles can be considered impersonators.

In case the model is not trained on the person, the tool will not be able to retrieve the name of the person and therefore not be able to scrape the platforms for profiles. In this case, the user is asked to enter a name if available. In addition, with more users using the tool, the dataset will grow large enough to be able to retrieve a name for the input image most of the time.

As an example, suppose the user wants to search for the image shown in Fig. 3. First, the user needs to upload the image to the tool and pick the platform to search on i.e. Facebook or Instagram. After that, the tool gives the user the output as shown below in Fig. 4, which includes the profiles that were found during the ML and scraping process. The user is asked to choose which of the profiles is theirs and the rest are considered impersonators. The outputs of the ML model along with the output of the web scraping are shown in Fig. 5 to the user so that the user can verify which of the profiles is theirs. In case the ML model was not trained on that person in the input image, the scraping cannot take place and so no further actions can take place. The user is asked to enter a name at least as shown in Fig. 6.

Fig. 3: User input image

2) Case two: Input Image & Name : In the second case, the user inputs the image as well as the name. In this case, the name retrieval step from ML can be skipped. The profile that the ML model has trained on (assuming the model has trained on the person previously) along with the web scraper

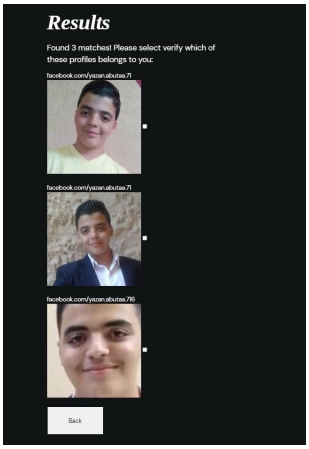

Fig. 4: User verification phase

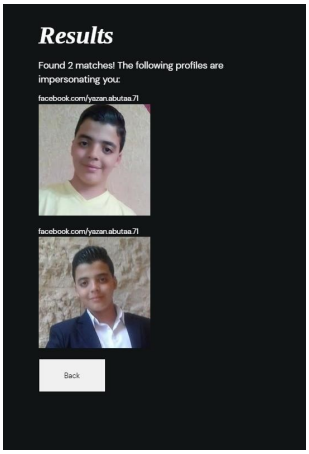

Fig. 5: The output after the user has selected their profile

output will be shown to the user and the user is then asked to verify which of the profiles is theirs. Now in case the profile that was output from the ML model has a different name from the user input name, the user verification step can be skipped and that account with a different name can be considered the impersonator. In case it was the same name, the user verification step will be required.

3) Case three: Input Image & Name & Username : In the third scenario, the user has to input all three parameters: image, name, and username. Here there is the advantage of comparing the usernames as well and skipping the user verification in case there is a match between the web scraping output username

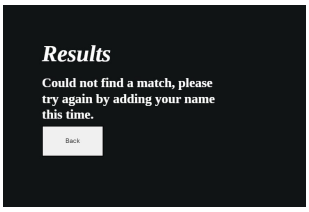

Fig. 6: Model is not trained on the person

and the input username by the user. In case there is no match, the user verification step is required. All the profiles with non-matching usernames are marked as impersonators.

C. Model Accuracy

Three classification metrics were used to evaluate the performance of the algorithm used in this paper. These metrics are accuracy, precision, and recall. Accuracy was calculated by dividing the number of correctly predicted values by the total number of values. The correctly predicted values equal the sum of True Positives (TP) and True Negatives (TN) while the total number of values equals the sum of True Positives, True Negatives, False Positives (FP), and False Negatives (FN) as shown in Equation 1.

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

Precision is the proportion of correct positive predictions of all cases classified as positive and it can be calculated as shown in Equation 2.

$$precision = \frac{TP}{TP + FP} \tag{2}$$

On the other hand, the recall measures the proportion of actual positive labels correctly identified by the model. The recall can be calculated as shown in Equation 3

$$recall = \frac{TP}{TP + FN} \tag{3}$$

The proposed model achieved an accuracy of 88% with precision and recall matrices of 86% and 89% respectively.

V. Conclusion and Future work

This paper presents the development of a tool aimed to detect the entity engaging in user impersonation, using machine learning and web scraping techniques. The tool is designed to be deployed on a cloud-based infrastructure to enhance its scalability and accessibility. The proposed approach was divided into three phases: machine learning, web scraping, and user verification phase. Each of these phases was complimentary to the other part to generate an accurate result for the user. The proposed model achieved an accuracy of 88% with precision and recall matrices of 86% and 89% respectively.

Multiple potential areas can be taken into consideration for future work. Firstly, a mobile application can be implemented for iOS and Android devices incorporating the same functionality presented in this work. Furthermore, the integration of the Meta API can enhance the scraping speed and the accuracy of the search. Additionally, more platforms to scrape can be added such as LinkedIn and Twitter. Another potential area is to generate a bigger dataset by increasing the number of users which means better results.

References

[1] M. Glassman, M. J. Kang, Intelligence in the internet age: The emergence and evolution of open source intelligence (osint), Computers in Human Behavior 28 (2) (2012) 673–682.

[2] D. Learning, Deep learning, High-dimensional fuzzy clustering (2020).

[3] B.-J. Koops, J.-H. Hoepman, R. Leenes, Open-source intelligence and privacy by design, Computer Law & Security Review 29 (6) (2013) 676–688.

[4] J. R. G. Evangelista, R. J. Sassi, M. Romero, D. Napolitano, Systematic literature review to investigate the application of open source intelligence (osint) with artificial intelligence, Journal of Applied Security Research 16 (3) (2021) 345–369. doi:10.1080/19361610.2020.1761737.

[5] Y.-W. Hwang, I.-Y. Lee, H. Kim, H. Lee, D. Kim, Current status and security trend of osint, Journal of Wireless Communications and Mobile Computing 2022 (2022).

[6] A. Kanta, I. Coisel, M. Scanlon, A survey exploring open source intelligence for smarter password cracking, Forensic Science International: Digital Investigation 35 (2020) 301075.

[7] D. V. Lande, E. V. Shnurko-Tabakova, Osint as a part of cyber defense system (2019).

[8] F. Tabatabaei, D. Wells, Osint in the context of cyber-security, Open Source Intelligence Investigation: From Strategy to Implementation (2016) 213–231.

[9] J. Pastor-Galindo, P. Nespoli, F. Gómez Mármol, G. Martínez Pérez, The not yet exploited goldmine of osint: Opportunities, open challenges and future trends, IEEE Access 8 (2020) 10282–10304. doi:10.1109/ACCESS.2020.2965257.

[10] Eagle eye, https://github.com/ThoughtfulDev/EagleEye, accessed: 2023-5-5.

[11] Pimeyes, https://pimeyes.com, accessed: 2023-5-5.

[12] Social mapper, https://github.com/Greenwolf/social_mapper, accessed: 2023-5-5.

[13] R. Sharma, V. K. Sharma, A. Singh, A review paper on facial recognition techniques, in: 2021 Fifth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC), IEEE, 2021, pp. 617–621.

[14] R. Jafri, H. R. Arabnia, A survey of face recognition techniques, journal of information processing systems 5 (2) (2009) 41–68.

[15] W. Ali, W. Tian, S. U. Din, D. Iradukunda, A. A. Khan, Classical and modern face recognition approaches: a complete review, Multimedia tools and applications 80 (2021) 4825–4880.

[16] M. A. Khder, Web scraping or web crawling: State of art, techniques, approaches and application., International Journal of Advances in Soft Computing & Its Applications 13 (3) (2021).

[17] R. Gunawan, A. Rahmatulloh, I. Darmawan, F. Firdaus, Comparison of web scraping techniques: regular expression, html dom and xpath, in: 2018 International Conference on Industrial Enterprise and System Engineering (ICoIESE 2018), Atlantis Press, 2019, pp. 283–287.

[18] D. K. Mahto, L. Singh, A dive into web scraper world, in: 2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom), 2016, pp. 689–693.

[19] P. Srivastava, R. Khan, A review paper on cloud computing, International Journal of Advanced Research in Computer Science and Software Engineering 8 (6) (2018) 17–20.

[20] Url handling library, https://docs.python.org/3/library/urllib.html, accessed: 2023-5-5.

[21] N. Zhang, J. Luo, W. Gao, Research on face detection technology based on mtcnn, in: 2020 international conference on computer network, electronic and automation (ICCNEA), IEEE, 2020, pp. 154–158.

[22] Social media users, https://www.statista.com/statistics/264810/number-of-monthly-active-facebook-usersworldwide, accessed: 2023-5-5.