THE BRIDGE

Convolutional 3 Layers

Secure Authentication System

Conv3L S.A.S.

v1.0

data-science-ft-sev-mar22
July 2022

Table of content

Introduction

In terms of completing our formation in Data Science, we wanted to implement a secure system based on facial recognition as an authentication method.
The team that has brought this project together implemented the knowledge learnt in the Bootcamp, in a final group project.

The Bootcamp given by The Bridge was awarded as Best Data Science Bootcamp by Course Report in 2021, proving the quality of the content they offer, in a technical and theoretical level. We also want to thank our Lead Instructor Alberto Becerra Tomé and our Teacher Assistant Julio Bravo-Ferrer Acosta, for their effort and dedication throughout the intense Bootcamp, in addition to their guidance in this system, so it could be brought together.

Talking about us, the authors of this project, we are six passionate data scientist who try their best to learn in this short period of intense formation:

- Ouissam Fechtali Othman.
- Daniel Martinez Soria.
- Jose Enrique Vera Rodríguez.
- Jose Luis Torres Andrades.
- Francisco Ruiz Guerra.
- Mario Chacón Ruiz.

Nowadays, covering the main theme of the project, the traditional login way has been through a user and password. This is implemented in basically every system we know. We have wanted to create something different, something more secure for the user, implementing a 3 layer security system as an authentication method.

Our system is based on 3 independent security layers to log in. This can be implemented in websites and web applications, following current programming standards, where secure access to private information is an essential requirement. The system we want to create is implemented as an API, with several Endpoints, that way it can communicate to log into a website through a dedicated user interface (Login page).

We will define this documentation separating the 3 authentication layers mentioned above:

- Facial Recognition.
- Email and Password.
- Secure Pin.

The project encompasses the technologies learnt throughout the Bootcamp: Python language, numpy objects, Machine Learning techniques, image processing, use of databases and APIs routes, apart from having to cover Full Stack tasks ourselves, as our team had no Full Stack developers.

Through this document, we will be developing and explaining the content of the project.

Methodology

We have implemented an organized and effective methodology to establish a productive environment.

The whole project has been based on Agile Methodology. Since the first part, we established sprints and tasks for those sprints. We had a Scrum Master to conduct the daily meetings and also sprints meetings. Also we simulated a product owner to perform our work within a real working environment, even though this is the final project of an practical-academic formation into the digital sector. We established short daily meetings to update each one of us and also used tools like Slack and Discord as our communication channels, inside and outside those meetings. For the main meeting, the first one of our project, we established the sprints and also the activities, individual and group ones, that were required for the correct development of our authentication system. From that meeting, we were establishing daily meetings (as Agile Methodology indicates), to communicate with each other and be an active part of the whole project, participating and being in constant communication among the members of the team.

We have used Trello to have a common platform that allowed having a board containing the to do tasks. Aswell as the individual tasks, group ones were posted in our Trello board so we could be all aware of the upcoming tasks. All of the members of the team were in constant contact with our Trello platform. That way, we have been involved and aware of the holistic project, even though each of us was focused on a specific part of the project.

In addition, our work has been implemented through GitHub, creating a repository where all of us are collaborators. Each one of us had permits to contribute and add content to our whole project, so we could all add and participate equally. Within the GitHub platform, we have followed professional methodology and have created branches, different from the main branch of our *Conv3l* system, for development, and the creation of the different softwares that will form the program. Once we had content created, we would merge the branches to the branch used for the development of the project using what is called *pull requests*, so we could all observe and analyze the changes. That way we would avoid conflicts and overwriting of coding done for the project.

To maintain the professionality of our project, we also created and worked in a virtual environment for a simpler way to display the project for future users, avoiding version conflicts.

Also, by implementing testing to our functions, we have implemented a solid way to advance in our software, maintaining a quality in our codes that could be reliable to our colleagues that would need that code for the next step of the project.

Resources

The Technologies used for this project have been the following:

- Python.
    - OpenCV library.

    - Numpy library.
    - Pytest library.
    - Os library.
    - Imutils library.
    - Flask library.
    - MySQL library.
- Trello.
- Slack.
- Discord.
- Visual Studio Code.
- Google drive.
- Github.
- iStock (image bank).
- Pexes (image bank).
- Canvas.

Scripts content

First Layer - Facial Recognition

Our project is developed using Python.

For the first part of the authentication system, we needed to process different files so we could generate pictures of ourselves. That way the model that we would later develop could be trained and could “recognise” us for the final login idea, giving us access or not, depending if the software was able to recognise the face in front of it and that face was within the database of ‘allowed’ individuals to log in.

To be able to achieve that processing, us, the members of the project, had to record ourselves in a video so we could record our faces in front of the camera in different angles, with different illumination settings and even different haircuts (for those who need it) so the model could be trained and could recognise our faces within different environmental features, such as light, glasses, face angles, facial expressions, etc.

Such features were captured in the videos. Once we had nurturing content in those videos, we then had to process them.

That was the first step to create the secure access system we want to implement in our project. The way to process those videos was with the OpenCV library. Using Python as our coding language, we were able to develop a code that would process the video, being able to recognise the faces detected with the software created with OpenCV. What that library enabled us to do was basically recognise an object, and that specific object was a face. Specifically, the steps needed to capture our faces was to set up a camera. OpenCV has got the .VideoCapture() method to do that. We set the camera to be our webcams. Then, we had to create the route to store the faces that we would capture. After that, we had to use the .read() method to “turn on” the webcam and then be able to capture our preferred objects. After setting the sizes we would transform the images captured and setting the hyperparameters required to adjust the capture with the .detectMultiScale() method.

Once we were able to recognise faces in those videos, we were able to obtain different frames and (at the same time), different pictures of our same faces, so then we could have a numerous database for our model to be trained with each of the faces the model detected.
In particular, we obtained three hundred pictures of each one of us initially. Further along the project, we needed more pictures out of each of us so we could train and “teach” the model different angles, haircuts, lights than the first video. The reason to generate more pictures of ourselves is basically to feed our model enough information for it to be able to recognise our picture more easily and with less room for error. That way, the six members of the team that would get access to the main log in system that we aim for the present project.

The end of the first part of the project was converting each of those pictures into numpy arrays(the specific object training models require), so they could be processed in the following stages of the project. To be able to obtain a list of arrays (representing each array a single picture), we first converted the captured frames into a scale of grays, that way we could eliminate color factors such as clothes worn at the moment of the scan and log in

stage. Also, converting those files into a scale of grays is a mandatory step of the training process based on EigenFaces (we will dive into it on the following lines), a method we decided would be the base for our machine learning model.
Once we had all of the pictures into a tone of grays, we then processed the images into a list of numbers, which will be the element that the model will be able to accept as input and then process it.

Now that we have preprocessed images, then we are able to train our model.

The model was created using OpenCV library method .EigenFaceRecognizer_create(). That way we had a model established. Then we feed the model the numpy arrays we have got (transformed into structured size, our selected size was 150x150 pixels and in a grayscale). To feed the model, we have had to label each one of us's faces. That way, the model is able to recognise who is who. For that, we create loops that go through the content of each folder in our route and path. The method used in OpenCV library to train our data is .train(), establishing as arguments the numpy arrays (preprocessed images) and the labels created. Once the model is trained, having fed the data we need, we want to store the model so it's ready to act anytime we need. This is basically, everytime we want it to predict a face so it can identify a face so it can determine whether someone is allowed to log in using our authenticator system. The method used to store our model es .write(), giving the name to our model within the brackets.

Once we have got our model trained and stored, we are able to predict a result. To be able to make a valid prediction, we had to develop a code enabling us to do so. Firstly, we load the model we created previously and then we read that model. Then,we had to activate the webcam that is going to capture the image we are going to predict. You may realize this is similar to the steps developed earlier, exactly. This is a necessary step that we have to repeat in order to recognise the object we want to predict, whether allowed or not, in our system. We have to set the camera again as we are not uploading a set file with a set picture of ourselves, but we are going to use a live video of ourselves that could be implemented, for example, in a door box, allowing us access into a room. Once the camera is up and running, we had to specify the object we want our model to focus on. In our case, obviously, was our face. Then, again, we have got to set hyperparameters, repeating the process previously explained, so we could read a clean face. At this point, that we have got a face identified within a live video, we use the method .predict(). This OpenCV method, will determine whether the face is evaluating has been previously trained or not. If we have got a labeled trained face, the model will recognise it and, in our authentication system, will allow it to enter/log in. On the other hand, if the evaluated face is not within the labels and faces registered previously in our trained model, will not recognise it and won't get access to it. That way, we have got the first step of our 3 layers secure authentication system. Ultimately, to break and to turn off the prediction step, we set the ESC key with the method .waitKey() == 27. That way, if we press the mentioned key, we can exit the recognition process.

Second Layer - Email and Password

For the second part of our 3 layer authentication system ….

Third Layer - Google Authenticator

As the third layer of our 3 layer authentication system…..

Results

After having elaborated the 3 layers mentioned throughout the entire project, we have found that we were able to generate a whole image bank, so we could train a model with specific individuals (the authors of this project), and that way, we could establish a method to securely authenticate each one of the program users. We have also created a database in which we store the details of the users who signed up to be able to access the secure method to log in. Within the database, as we mentioned above, we have generated a csv certificate for each user. Along the way, we also created a communication channel to generate a secure pin, using Google Authenticator app, validating a PIN number generated on the log in system homepage.

With those layers programmed and functioning, we created a path and a structure able to sign up an user, then feed image frames of that user's face so our facial recognition machine learning model can train on them so it can give access, to then complete the other 2 security layers to ensure a safe authentication throughout a created website graphic interface.

Ultimately, following cybersecurity philosophy, we have created a login system that satisfies the three essential elements, something that we are, something that we have and something that we know. Our face is what we are. The csv certificate is what we have . And our Pin number is what we know.

That way, we focus all of our efforts to avoid any false positives, reassuring the consumers our product would be reliable, giving access with a minimal amount of error.

Bibliography

For the Bibliography of this project, we have based the learning process and we have supported our code with the knowledge proceeding from:

References

Agarwal, V. (2020, May 13). *Complete Image Augmentation in OpenCV*. Towards Data Science. Retrieved July 12, 2022, from https://towardsdatascience.com/complete-image-augmentation-in-opencv-31a6b02694f5

Amat Rodrigo, J. (2021, May 1). *Reconocimiento facial con deep learning y python*. Ciencia de datos. Retrieved July 9, 2022, from https://www.cienciadedatos.net/documentos/py34-reconocimiento-facial-deeplearning-python.html

The Bridge. (2022, March 15). *SVL_DS_Marzo_TB*. GitHub. https://github.com/AlbertoBecerraTB/SVL_DS_Marzo_TB

Chowdhury, S. (2021, May 7). *How to Capture Images Using React Webcam*. JavaScript in Plain English. Retrieved July 7, 2022, from https://javascript.plainenglish.io/capture-images-via-webcam-using-react-9282bb87de5a

Chowdhury, S. (2022, March 5). *React webcam*. GitHub. Retrieved July 10, 2022, from https://github.com/Sristi27/React-webcam

Esparza Franco, C., Tarazona Ospina, C., Sanabria Cuevas, E., & Velazco Capacho, D. A. (2017, May 1). *RECONOCIMIENTO FACIAL BASADO EN EIGENFACES, LBHP Y FISHERFACES EN LA BEAGLEBOARD-xM*. ResearchGate. Retrieved July 9, 2022, from https://www.researchgate.net/publication/319958509_RECONOCIMIENTO_FACIAL_BASADO_EN_EIGENFACES_LBHP_Y_FISHERFACES_EN_LA_BEAGLEBOARD-xM

GitHub. (2020, April 13). *HaarCascades*. GitHub. Retrieved July 07, 2022, from https://github.com/opencv/opencv/tree/master/data/haarcascades

Infinite Loop Development Ltd. (2019, November 13). *AuthenticatorAPI.com*. GitHub. Retrieved July 10, 2022, from https://github.com/infiniteloopltd/AuthenticatorAPI.com

Lü, C. (2022, July 5). *Create React App*. React. Retrieved July 10, 2022, from https://create-react-app.dev/docs/getting-started/

Mühler, V. (2020, April 22). *JavaScript API for face detection and face recognition in the browser implemented on top of the tensorflow.js core API*. Face API. Retrieved July 9, 2022, from https://justadudewhohacks.github.io/face-api.js/docs/index.html

Omes, G. (2020, May 24). *Reconocimiento Facial OpenCV - Python*. YouTube. Retrieved July 7, 2022, from https://www.youtube.com/watch?v=cZkpaL36fW4

OpenCV. (2022, May 20). *FAQ OpenCV*. FAQ. Retrieved July 06, 2022, from https://github.com/opencv/opencv/wiki/FAQ

OpenCV. (2022, June 5). *OpenCV*. OpenCV: Installation in Windows. Retrieved July 6, 2022, from https://docs.opencv.org/4.6.0/d3/d52/tutorial_windows_install.html

OpenCV. (2022, June 5). *OpenCV: Cascade Classifier*. OpenCV: Cascade Classifier. Retrieved July 6, 2022, from Cascade Classifier

OpenCV. (2022, June 5). *OpenCV: Image Processing in OpenCV*. OpenCV: Image Processing in OpenCV. Retrieved July 06, 2022, from https://docs.opencv.org/4.6.0/d2/d96/tutorial_py_table_of_contents_imgproc.html

Pearson IT Certification. (2011, June 6). *Understanding the Three Factors of Authentication*. Pearson. https://www.pearsonitcertification.com/articles/article.aspx?p=1718488

Pytest. (2021, March 7). *Pytest: helps you write better programs*. Pytest. Retrieved July 10, 2022, from https://docs.pytest.org/en/7.1.x/index.htm

Rawat, A. (2022, July 1). *Calling REST APIs From the IDE : VSCode*. ITNEXT. Retrieved July 10, 2022, from https://itnext.io/calling-rest-apis-from-the-ide-vscode-57e59e003133

React. (2021, July 14). *Documentación de React y recursos relacionados.* React. Retrieved July 9, 2022, from https://es.reactjs.org/docs/getting-started.html

Rosebrock, A. (2021, May 10). *OpenCV Eigenfaces for Face Recognition*. pyimagesearch. Retrieved July 09, 2022, from https://pyimagesearch.com/2021/05/10/opencv-eigenfaces-for-face-recognition/

Roseman, D. (2017, August 29). *How to store dictionary and list in python config file*. Stackoverflow. Retrieved July 10, 2022, from https://stackoverflow.com/questions/45948411/how-to-store-dictionary-and-list-in-python-config-file

Ruan, B. (2019, August 18). *Face Detection with Face-api.js*. Benson Technology. Retrieved July 10, 2022, from https://bensonruan.com/face-detection-with-face-api-js/

Sarkar, T. (2021, April 24). *PyTest for Machine Learning — a simple example-based tutorial*. Towards Data Science. Retrieved July 11, 2022, from https://towardsdatascience.com/pytest-for-machine-learning-a-simple-example-based-tutorial-a3df3c58cf8

Shaw, A. (2018, October 27). *Getting Started With Testing in Python*. Real Python. Retrieved July 10, 2022, from https://realpython.com/python-testing/

Shaw, A. (2021, November 10). *Advanced Visual Studio Code for Python Developers*. Real Python. Retrieved July 11, 2022, from https://realpython.com/advanced-visual-studio-code-python/#testing-your-python-code-in-visual-studio-code

Vlogs, R. (2012, September 16). *Viola Jones face detection and tracking*. YouTube. Retrieved July 6, 2022, from https://www.youtube.com/watch?v=WfdYYNamHZ8

Wilson, J. (2021, August 20). *How to install ReactJS on Ubuntu 20.04*. RoseHosting. Retrieved July 11, 2022, from https://www.rosehosting.com/blog/how-to-install-reactjs-on-ubuntu-20-04/

Suggestions

For the next versions, we can implement the following improvements to the current project:

- Mounting the whole project into a website application.
- Having access to stronger computational systems to be able to handle mass size labels and registrations.
- Adding different Endpoints:
    - User list: to show a list of the signed up users.
    - Add user: to sign an user up into the database.
    - Delete user: to erase an existing user from the database.