26/08/2023

Use of neural networks to automate administrative dossier processing

Author: Ortega Núñez, Daniel

Supervisor: Chousa Arza, Brais

Collaborating Entity: PricewaterhouseCoopers Asesores de Negocios S.L.

ABSTRACT This paper presents the design and development of a first version of an automatic documental review system based on object detection techniques using neural networks and its combination with Optical Character Recognition (OCR). In this first version, the system focuses on the detection and verification of "*Números de Identificación Fiscal*" (NIF) or "*Número de Identidad de Extranjero*" (NIE) in legal representation documents. To achieve this detection, an architecture based on the YOLOv8 model has been implemented. The developed solution has been tested to evaluate the accuracy of the detection model, the execution times, and the validation threshold, from which results have been obtained showing that the system is able to detect and verify NIF or NIE accurately within the representation templates with which the model has been trained, and obtaining adequate execution times to handle a significant volume of dossiers.

INDEX TERMS YOLO, OCR, Artificial intelligence, Dossiers, Neural network, CNN, Document review, Legal representation.

I. Introduction

The documental review is a fundamental process in the administrative dossier processing. This process is mainly performed manually by document reviewers, which can be difficult and susceptible to errors, especially in environments with a high volume of dossiers. This manual aspect of the document review process can lead to an accumulation of dossiers due to the time required for this review, which can cause specific problems in this type of procedure, such as: delays in the resolution of dossiers, errors in the document review, or a possible increase in the cost of the staff in charge of the document review.

This problem leads to the need to find a solution to automate document review, reducing the time and errors associated with manual review. To this end, we explore the integration of object detection models in digitized documents and the application of OCR to verify the validation of the document and, therefore, the applicant's compliance with the requirements.

In summary, this article addresses the need to optimize the document review process in administrative procedures through automation, presenting a technological approach that integrates object detection techniques and OCR for document verification, specifically for legal representation documents.

II. Project definition

The main objective of the project is to design and implement a solution to automate the document review in administrative documents of an existing platform, specifically it will focus on the review of the document that allows the accreditation of legal representation. For this purpose, in the first instance, a version of the object detector is developed to identify and validate the NIF/NIE in the corresponding document, providing a technological alternative to improve the speed and accuracy in the accreditation of legal representation.

To carry out this development, the final system must have different essential requirements in order to confirm the validity of the solution provided:

i) Detection of regions of interest: The system must be able to detect the position and area of the information of interest in the documents.
ii) Verification of the information detected: Once the information in the document has been detected, it must pass through a verification process to confirm its

exactitude, which implies comparing the information extracted by the OCR with the actual values present in the dossier.

iii) Document format flexibility: The system must be able to handle different types of documents and formats, adapting to variations in legal representation documents.

iv) Integration with platform: The system must be designed in such a way that it can be easily integrated with the existing document review platform, without interrupting or affecting the current workflow.

v) Transparent results: The system must provide clear and detailed results, including information about the regions of interest detected, object detector and OCR confidence scores, and the verification decision, to be fully transparent about the decisions made.

vi) Scalability potential: The solution must be scalable to handle a considerable volume of documents and dossiers, with reasonable processing times.

Based on these requirements, a solution can be designed to reduce the manual workload, minimize human errors, and speed up the document review process in administrative scenarios.

III. System description

The designed solution uses as main technologies a neural network based on YOLO to locate the fields of interest of the document (trained to detect NIF/NIE), and OCR to digitize the detected information and compare it with the real information of the file.

To detail the operation, it is necessary to start by describing the tasks performed to prepare the model used in the solution:

A. DATA COLLECTION AND ANNOTATION

First of all, in order to carry out the training of the data detection model, it is necessary to create a training dataset. In this case, as it is going to be trained for a particular set of objects (NIF/NIE detection), a dataset had to be developed manually from templates of the legal representation document of the existing platform.

From this dataset, it is necessary to label the positions of the fields of interest within the image, in other words, the coordinates where the detector model should find that field. Therefore, the location of the NIF/NIE in the used templates of the dataset was manually labeled for the specific casuistic of the project. For this, the manual annotation tool "Computer Vision Annotation Tool (CVAT)" was used, which allowed drawing bounding boxes around the NIF/NIE on the image and stored the coordinates of these boxes. When exporting the labels, the CVAT allows to use a format suitable for training the YOLO model.

It is worth mentioning that these labels include the coordinates (x, y) of the upper left corner and lower right corner of the bounding box around the NIF/NIE, and the class number of the object (necessary in case of detecting several objects).

B. YOLOv8 OBJECT DETECTION MODEL

The Ultralytics YOLOv8 model was selected as the basis for this project due to its ability to perform object detection with high accuracy and efficiency, as well as its large amount of documentation.

The model was configured to suit the specific requirements of the project. To do so, from the YALM (Yet Another Markup Language) configuration file, the architecture of the model is defined. In this case, the YOLOv8n version was selected to speed up training, since it is the network with the fewest parameters.

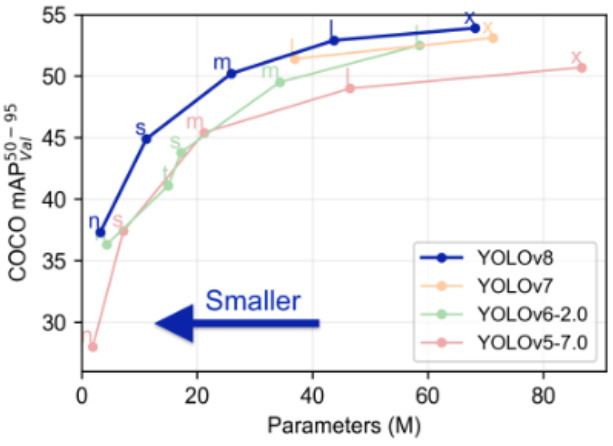

Figure 1: Number of parameters depending on the model.

During the training process, the model is dedicated to adjust its weights and internal parameters to learn how to detect the classes of objects indicated in the images. The training was performed iteratively over 100 epochs, and multiple times for different dataset sizes. Once the training is finished, the last model, and the best model, is saved. The model used was a model trained from Google Colab, with a set of 300 images and only detecting one object class (NIF).

In addition to the main technologies mentioned above, the developed system has several image and data treatment processes. A general scheme of the system is shown below and all the processes that are carried out during the execution of the solution are detailed in order:

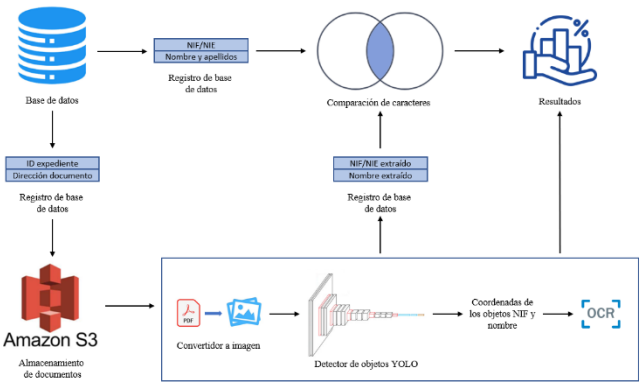

Figure 2: System design.

A. LOADING DOSSIER INFORMATION

The first stage of the process involves the collection and preparation of the necessary dossier data. In this test version, no direct communication with the database has been implemented, a data structure called "data_mapping" has been created containing the information of the test dossiers to be processed. This information is previously introduced and includes for each document name: the dossier identifier, the applicant's and representative's NIF/NIE, as well as the associated file addresses. In addition, the documents to be processed will have to be uploaded to the selected folder.

B. DOCUMENT TRANSFORMATION

To enable automated processing, a process of converting PDF documents to images is carried out using the pdf2image library.

This is done for all documents to be analyzed at the same time. The list of all documents with PDF extension within the input folder is obtained, and all found documents are transformed to images. These images are stored with the same name in the same location as the transformed documents.

C. DETECTION OF REGIONS OF INTEREST IN THE DOCUMENT

Once the documents are converted to images, these images are processed one by one using the trained YOLO model. This model, trained to detect NIF/NIE, identifies the regions that it considers that store this object:

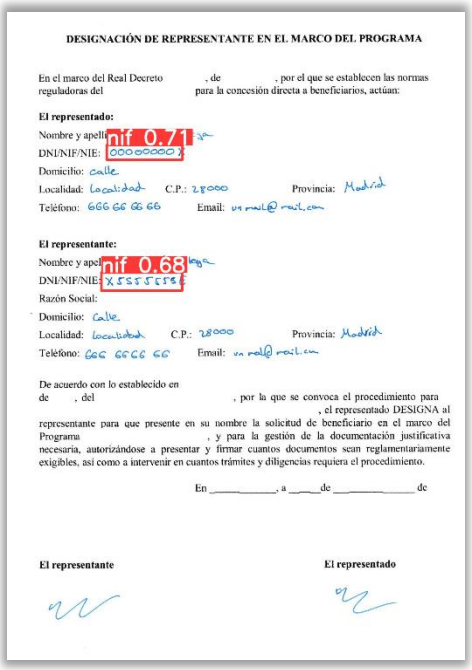

Figure 3: Document processed.

If the NIF object is found in the processing, the information of the coordinates of the contour box of the object is used, and the image is cropped twice, for the detection of the NIF of the interested and for the NIF of the representative, saving them in the folder indicated as "output". In case no object is detected, a message detailing the event is displayed, and a negative verification is exported as the result of this processed document.

D. IMAGE PROCESSING AND OCR

Once the region of interest containing the NIF of an image has been identified, the text extraction process is carried out in several steps to guarantee an accurate and reliable reading.

First, as discussed above, the images are cropped from the coordinates of the objects detected by the neural network. These cropped images must be processed using the OCR, but before this is done, they are processed to facilitate the reading of the OCR and improve the accuracy with which the information is extracted from the images.

This treatment consists of transforming the image to grayscale, and then applying a binary threshold to segment the image in black and white, where the pixels of the image that exceed the indicated threshold value of the scale become white, and those below become black. In this way, the detection of image edges and details is

improved, highlighting the characters, and avoiding some possible OCR reading failures due to the image quality.

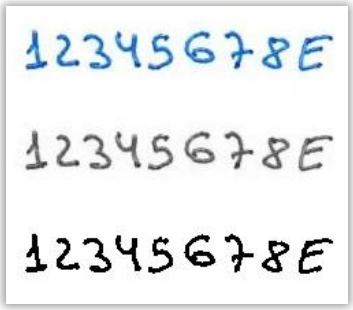

Figure 4: Image treatment before OCR.

Once the image has been treated, it is processed using OCR. Specifically, the EasyOCR library has been used for this character recognition task, extracting with it the contents of these images, in other words, the fields of interest (NIF of the interested party and representative) in string format.

It is worth mentioning that this process also provides a confidence score for each text extraction, which could be considered for the document validation decision.

E. TEXT PROCESSING

The next step of the system is to process the text extracted from the OCR. The aim is to ensure that the text maintains the usual NIF/NIE format, avoiding reading errors when confusing similar characters, or even the introduction by the user of special characters that do not maintain the format stored in the database.

To do this, first of all, all letters in the text are converted to capital letters, and symbols such as "-" or "." are eliminated. Secondly, all texts are processed by a function developed, which controls the specific formats of the NIF and NIE.

The formats of both NIF and NIE are shown below to understand the function:

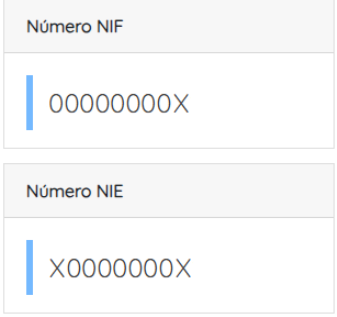

Figure 5: NIF and NIE format.

Taking these formats into account, the purpose is to control the correspondence of the characters in relation to numbers or letters to a certain point by means of the commented function. For this purpose:

- The first character can be a number or a letter depending on whether it is a NIF or NIE.
- The intermediate characters must always be numbers.
- The last character must always be a letter.

Following this guideline, dictionaries with similar numbers and letters that OCR may confuse have been declared, to be modified with this formatting criteria as appropriate. The controlled characters are shown below in Table I:

TABLE I
MONITORED CHARACTERS

LETTER/ CHARACTER	NUMBER
O	0
I	1
J	3
A	4
G	6
/	1
Z	2
S	5

F. CHARACTER STRING COMPARISON

Once the text processing is finished, it is possible to compare with greater certainty the information extracted from the OCR with the real one. So, using the difflib library to compare the similarity of characters, a similarity coefficient is obtained. This comparison is carried out character by character, in other words, the first character of a string is compared with the first character of the other string, the second with the second, and so on.

Through this process, the comparison information can be obtained from the similarity coefficient, the higher the similarity coefficient, the more similar are the two strings of characters.

G. VERIFICATION PROCESS

A key point is the formulation of the validation threshold, which determines the required similarity between the text extracted by the OCR and the actual NIF/NIE values to consider a document revision as valid. This threshold is based on the similarity coefficient calculated by comparing character strings but could be further defined

from the OCR confidence score, or if available, from the detection of other objects.

In this project, a validation threshold of 89% has been set as a balance between strictness of validation and acceptance of minor differences, which corresponds to one distinct character between strings.

In situations where the similarity coefficient does not meet the validation threshold, the system records the verification as invalid. This allows the document reviewers to manually verify the case without affecting the original flow of the file.

H. RESULTS EXTRACTION

Once the similarity coefficient has been obtained and the document validation decision has been calculated, all the data obtained during the processing of each document is collected. These data include: the dossier information (identifier and real NIF/NIE), the extracted NIF/NIE, the OCR scores, the YOLO contour box scores, the similarity percentages, and the final verification decision.

These data are stored in a data structure called "results", with which finally a "dataframe" of pandas will be created to later extract them with Excel format. In Table II below it is shown how the results table is structured, and the different data types of each variable.

TABLE II

TABLE DESIGN FOR RESULTS

CAMPO	DATA TYPE	DESCRIPTION
id_expediente	UUID	Unique dossier identifier
applicant_nif	VARCHAR	NIF/NIE of the interested coming from the database
representative_nif	VARCHAR	NIF/NIE of the representative coming from the database
applicant_extracted_nif	VARCHAR	NIF/NIE of the interested from the extraction of the document
representative_extracted_nif	VARCHAR	NIF/NIE of the representative from the extraction of the document
applicant_box_confidence	FLOAT	Score of detection by the YOLO model for the NIF/NIE of the interested
representative_box_confidence	FLOAT	Score of detection by the YOLO model for the NIF/NIE of the representative
applicant_ocr_score	FLOAT	Reliability of the reading by the OCR of the interested NIF/NIE
representative_ocr_score	FLOAT	Reliability of the reading by the OCR of the representative's NIF/NIE
applicant_similarity	FLOAT	Percentage of similarity between the extracted NIF/NIE of the interested and the one in the DB
representative_similarity	FLOAT	Percentage of similarity between the extracted NIF/NIE of the representative and the one in the DB
verification_decision	BOOLEAN	Decision for the verification of the document based on the results obtained

It is worth mentioning that this data may be loaded into the platform's database when deemed appropriate, showing users the corresponding interface in each case.

IV. RESULTS

This section presents the results obtained after the implementation of the automatic document review system, analyzing the most relevant aspects of the project and evaluating the efficiency and performance of the developed solution.

A. YOLO EXECUTION TIMES

One of the aspects evaluated in depth is the execution time of the YOLO model trained and used for the detection of objects in the documents.

During this object detection phase, the YOLO model processes each image in order to identify the NIF/NIE positions in the document.

The execution times oscillate between 135 and 155 milliseconds per image, which are considered appropriate times to handle large amounts of documents without generating significant waits in the execution of the solution.

Therefore, these values suggest that the YOLO model is efficient and scalable to a significant volume of files, affecting in the scale of hours for hundreds of thousands of additional documents. It is important to note that, as execution time is not critical for this particular project, there is scope to employ more complex models in order to achieve greater accuracy.

B. MODEL PERFORMANCE

Another aspect evaluated is the performance of the model in terms of accuracy and its limitations.

In relation to this, the developed model can successfully detect the NIF/NIE fields in documents based on the legal representation template used in the training, but its accuracy may be affected when facing different types of documents with different formats and structures.

In the context of object detection, this means that the model presents overfitting, and may have difficulties detecting regions of interest in documents that do not exactly follow the template used in training.

The overfitting of the model is a major concern, as it affects the flexibility and adaptability of the system, leading to the fact that, in the case of documents with different formats, the system fails to detect regions of interest, decreasing its accuracy drastically and reducing the number of dossiers that can be automatically review.

As a result, the solution has a limitation in that it is not generalizable to multiple document types without a model retraining. However, in the specific context of the document review platform on which the project is focused, it is working with the defined template, so the result obtained is valid.

C. VALIDATION THRESHOLD ANALYSIS

The last aspect evaluated in detail is the validation threshold used to decide where the detection of a region of interest in the digitized documents is correct or erroneous. This is a crucial component in the document review process, because depending on how strict the threshold is, the number of documents validated will increase or decrease.

When a strict validation threshold is used, that is, a high verification threshold, the tendency is to validate only those documents in which the extracted text coincides more exactly with the expected values. This leads to higher validation accuracy, as false positives are minimized. However, this approach can also lead to a reduced number of validated documents, as even small discrepancies in the extracted text due to reading errors can result in a negative validation. On the other hand, using a less stringent validation threshold, meaning a low threshold, increases the likelihood that more documents will be validated. However, this may lead to a higher number of false positives.

The choice of validation threshold should be a compromise between the accuracy desired, and how strict the threshold is. This should be based on an analysis of system requirements and tolerance for validation errors.

After an iterative adjustment, a validation threshold of 89% is defined for this project. This threshold allows for some variation in the extracted characters without compromising the validity of the document. It is important to note that the validation threshold is adjustable according to the needs of the process and can be refined according to the review requirements. Moreover, parameters such as OCR accuracies, or even additional parameters such as the requirement to detect other objects through YOLO, for example signatures (if the document does not have a signature, it will not be considered valid regardless of the percentage of similarity of the NIFs), could be added to this validation threshold.

V. CONCLUSIONS

This chapter presents the conclusions obtained after the execution of the automatic document review project. Through the definition of requirements, a functional solution has been achieved that meets the challenges proposed. The main conclusions are summarized below:

With the project, it has been possible to design and develop a document review system that speeds up the administrative dossier verification process. This system uses an object detection based on YOLOv8, which has proven to be effective in identifying regions of interest containing NIF and NIE, which could be expanded to new objects. In addition, OCR technology has been incorporated to extract information from the images, and a validation threshold has been established that determines the required similarity between the extracted values and the real values.

One of the main challenges of the project was the overfitting of the detection model, which limits its ability to adapt to different document formats. Although this problem does not have a major impact on this project, it is essential to consider it if a more versatile solution is needed. In addition, it should be considered that the detection model only identifies NIF and NIE, so a complete document review would require the detection of additional fields such as names or signatures.

In addition, automating verification reduces manual workload and minimizes human error, resulting in a faster and more reliable process. This improves the experience of both the reviewers and the applicants of the dossiers.

Despite the identified limitations, the solution meets the defined requirements, and it is possible to integrate the results easily into the existing document review platform when required.

In summary, the project has demonstrated the viability of implementing an automated document review system for administrative dossiers. Although there are areas for improvement, the current solution can significantly affect the document review process and have a positive impact on the management of administrative records.

VI. FUTURE TASKS

Opportunities for expansion and improvement have been identified for the developed system. Below are some areas where future work could be carried out to further optimize the system:

i) Expand the training data set to include different types of documents and templates, which will allow the model to generalize better in different contexts, solving the problem of overfitting.
ii) Use a model with a more powerful architecture, since by using a larger volume of parameters a higher accuracy could be obtained. This would affect the model execution speed but has a reduced impact since it can be performed independently from the manual review process without interrupting it.
iii) The current solution focuses on the detection of NIF and NIE, which meets the project objective of evaluating the viability of the system. However, in order to achieve a complete validation of the legal representation document, it would be beneficial to include the detection and validation of other fields such as signatures, which would require a retraining of the model and adjustments in the image processing stage.
iv) Research and implement new and more advanced OCR technologies to improve the accuracy of character reading. This would have a direct impact on the quality of the extracted information and, therefore, on the accuracy of the validation process.
v) Carry out tests in real production situations. This will allow evaluating the solution under more realistic conditions and collecting additional data to adjust and optimize detection and validation parameters. In addition, feedback from real users can provide valuable information for future improvements.

In conclusion, a solution has been designed and implemented to meet the objectives and requirements defined for the project, although it is true that during its execution, potential actions for improvement have been identified, with which the solution could be implemented in any type of document review platform, beyond the one used as a reference for the project.

APPENDIZ I –MAIN FILE CODE

The following section show the main.py file developed for the execution of the designed solution:

```
import os
from ultralytics import YOLO
import cv2
from pdf2image import convert_from_path

import easyocr
import difflib
import pandas as pd



# definir funciones
def convertir_pdf_a_imagen(archivo_pdf):
    imagenes = convert_from_path(archivo_pdf,
first_page=1, last_page=1)

    nombre_archivo =
os.path.splitext(archivo_pdf)[0]

    for i, imagen in enumerate(imagenes):
        ruta_guardado = f'{nombre_archivo}.jpg'
        imagen.save(ruta_guardado, 'JPEG')

def ocr_nif(img):
    text_detections = ocr.readtext(img)

    for text_detection in text_detections:
        _, text, score = text_detection

    return text, score

def format_nif(text):

    formatted_nif = text[0]

    for i in range(1, len(text) - 1):
        if text[i] in dict_char_to_int.keys():
            formatted_nif +=
dict_char_to_int[text[i]]
        else:
            formatted_nif += text[i]

    if text[-1] in dict_int_to_char.keys():
        formatted_nif += dict_int_to_char[text[-
1]]
    else:
        formatted_nif += text[-1]

    return formatted_nif

dict_char_to_int = {'O': '0',
                    'I': '1',
                    'J': '3',
                    'A': '4',
                    'G': '6',
                    '/': '1',
                    'Z': '2',
                    'S': '5'}

dict_int_to_char = {'0': 'O',
                    '1': 'I',
                    '3': 'J',
                    '4': 'A',
                    '6': 'G',
                    '2': 'Z',
                    '5': 'S'}
```

```
# ------------------------------------------------
# ------------------------------------------------

# CARGAR MODELOS
nif_detector = YOLO('H:/TFM/best_collab.pt')
ocr = easyocr.Reader(['es'], gpu=False)
results = []

# OBTENER REGISTROS DE BBDD
data_mapping = {
    'REPRESENTATIVE_LEGAL_ONE1.jpg': {
        'petition_id': '',
        'dossier_id': '',
        'applicant_NIF': '',
        'representative_NIF': '',
        'file_add': ''
    },
    'REPRESENTATIVE_LEGAL_ONE2.jpg': {
        'petition_id': '',
        'dossier_id': '',
        'applicant_NIF': '',
        'representative_NIF': '',
        'file_add': ''
    }
    # Agregar más entradas según sea necesario
}


# CONVERTIR DOC A IMAGEN
input_path = 'H:/TFM/data/input'
save_path = 'H:/TFM/data/output'

# Obtén una lista de todos los archivos PDF en la
carpeta
archivos_pdf = [archivo for archivo in
os.listdir(input_path) if
archivo.endswith('.pdf')]

# Convierte cada archivo PDF en imágenes
for archivo_pdf in archivos_pdf:
    ruta_archivo_pdf = os.path.join(input_path,
archivo_pdf)
    convertir_pdf_a_imagen(ruta_archivo_pdf)


images = [archivo for archivo in
os.listdir(input_path) if
archivo.endswith('.jpg')]
print(images)

for image in images:
    ruta_img = os.path.join(input_path, image)
    print("-------------------------------------")
    print(ruta_img)

    data=data_mapping[image]


    # PRECEDIR CON YOLO LA POSICION DEL NIF
    detections = nif_detector(ruta_img)[0]

    if len(detections) == 0:
        print('No se encontraron NIFs')

        results.append({
            'petition_id': data['petition_id'],
            'dossier_id': data['dossier_id'],
            'applicant_nif ':
data['applicant_NIF'],
            'representative_nif':
data['representative_NIF'],
```

```
            'applicant_extracted_nif ':
'NOT_FOUND',
            'representative_extracted_nif':
'NOT_FOUND',
            'applicant_box_confidence': '-',
            'representative_box_confidence': '-',
            'applicant_ocr_score': '-',
            'representative_ocr_score': '-',
            'applicant_similarity': '-',
            'representative_similarity': '-',
            'verification_decision': 0
        })
        continue

    print('detecciones:')
    detection = detections.boxes.data.tolist()
    print(detection[0])
    print(detection[1])



# RECORTAR NIF A PARTIR DE LA PREDICCION
    img = cv2.imread(ruta_img)

    image_name, image_ext =
os.path.splitext(image)
    new_image_name = f"{image_name}_1{image_ext}"
    image_path = os.path.join(save_path,
new_image_name)
    x1, y1, x2, y2, confidence1, type =
detection[0]
    nif1 = img[int(y1):int(y2), int(x1):int(x2)]
    cv2.imwrite(image_path, nif1)

    new_image_name = f"{image_name}_2{image_ext}"
    image_path = os.path.join(save_path,
new_image_name)
    x1, y1, x2, y2, confidence2, type =
detection[1]
    nif2 = img[int(y1):int(y2), int(x1):int(x2)]
    cv2.imwrite(image_path, nif2)


# PROCESAR IMAGEN DEL NIF

    nif1_gray = cv2.cvtColor(nif1,
cv2.COLOR_BGR2GRAY)
    _, nif1_gray_th = cv2.threshold(nif1_gray,
180, 255, cv2.THRESH_BINARY)

    nif2_gray = cv2.cvtColor(nif2,
cv2.COLOR_BGR2GRAY)
    _, nif2_gray_th = cv2.threshold(nif2_gray,
180, 255, cv2.THRESH_BINARY)


# OCR DEL NIF
    nif1_text, nif1_score = ocr_nif(nif1_gray)
    nif2_text, nif2_score = ocr_nif(nif2_gray)

# PROCESAR TEXTO OBTENIDO
    nif1_text = nif1_text.upper().replace(' ',
'').replace('-', '').replace('.', '')
    nif2_text = nif2_text.upper().replace(' ',
'').replace('-', '').replace('.', '')

    nif1_text = format_nif(nif1_text)
    nif2_text = format_nif(nif2_text)

    print(nif1_text)
    print(nif2_text)
```

```
# COMPARACION DE STRINGS (DDBB y OCR)

    similarity_ratio_applicant =
difflib.SequenceMatcher(None, nif1_text,
data['applicant_NIF']).ratio()
    similarity_ratio_representative =
difflib.SequenceMatcher(None, nif2_text,
data['representative_NIF']).ratio()

    print(f"Porcentaje de similitud interesado:
{similarity_ratio_applicant:.2f}")
    print(f"Porcentaje de similitud representante:
{similarity_ratio_representative:.2f}")

# UMBRAL DE VERIFICACION
    if similarity_ratio_applicant > 0.75 and
similarity_ratio_representative > 0.75:
        verification_decision = 1
    else:
        verification_decision = 0

# ESCRIBIR RESULTADOS

    results.append({
            'petition_id': data['petition_id'],
            'dossier_id': data['dossier_id'],
            'applicant_nif ':
data['applicant_NIF'],
            'representative_nif':
data['representative_NIF'],
            'applicant_extracted_nif ': nif1_text,
            'representative_extracted_nif':
nif2_text,
            'applicant_box_confidence':
confidence1,
            'representative_box_confidence':
confidence2,
            'applicant_ocr_score': nif1_score,
            'representative_ocr_score':
nif2_score,
            'applicant_similarity':
similarity_ratio_applicant,
            'representative_similarity':
similarity_ratio_representative,
            'verification_decision':
verification_decision
    })
# Crear un DataFrame de pandas con los resultados
results_df = pd.DataFrame(results)

# DataFrame en archivo Excel para cargar en bbdd
excel_file = os.path.join(save_path,
'resultados_nif.xlsx')
results_df.to_excel(excel_file, index=False)
```

REFERENCES

[1] Amazon S3. [Online]. Available: https://aws.amazon.com/es/s3/getting-started/?nc=sn&loc=6&dn=1

[2] Dillon Reis, J., Kupec, J., Hong, J., & Daoudi, A. (May 17, 2023). Real-Time Flying Object Detection with YOLOv8. Available: https://arxiv.org/abs/2305.09972

[3] EasyOCR documentation. [Online]. Available: https://www.jaided.ai/easyocr/documentation/

[4] EnriqueAV. (May 12, 2018). Detección de objetos con YOLO: implementaciones y cómo usarlas. Available: https://medium.com/@enriqueav/detecci%C3%B3n-de-objetos-con-yolo-implementaciones-y-como-usarlas-c73ca2489246

[5] Great Learning Team. (Jul 21, 2020). YOLO object detection using OpenCV. Available: https://www.mygreatlearning.com/blog/yolo-object-detection-using-opencv/

[6] Hui, J. (Mar 28, 2018). Object Detection Speed and Accuracy Comparison: Faster R-CNN, R-FCN, SSD, and YOLO. Available: https://jonathan-hui.medium.com/object-detection-speed-and-accuracy-comparison-faster-r-cnn-r-fcn-ssd-and-yolo-5425656ae359

[7] Hui, J. (Mar 18, 2018). Real-time Object Detection with YOLO, YOLOv2 and now YOLOv3. Available: https://jonathan-hui.medium.com/real-time-object-detection-with-yolo-yolov2-28b1b93e2088

[8] Hui, J. (Mar 28, 2018). Understanding Region-based Fully Convolutional Networks (R-FCN) for object detection. Available: https://jonathan-hui.medium.com/understanding-region-based-fully-convolutional-networks-r-fcn-for-object-detection-828316f07c99

[9] Jifeng Dai, Yi Li, Kaiming He, Jian Sun. (Jun 21, 2016). R-FCN: Object Detection via Region-based Fully Convolutional Networks. Available: https://arxiv.org/abs/1605.06409

[10] Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi. (May 9, 2016). You Only Look Once: Unified, Real-Time Object Detection. Available: https://arxiv.org/abs/1506.02640v5

[11] Joseph Redmon, & Farhadi, A. (Apr 8, 2018). YOLOv3: An Incremental Improvement. Available: https://arxiv.org/abs/1804.02767

[12] Portal de Ayudas del Ministerio de Economía y Competitividad. [Online]. Solicitudes. Available: https://portalayudas.mineco.gob.es/THD/solicitudes/Paginas/Solicitudes.aspx

[13] Saarthi AI. (May 9, 2019). How to Build Your Own OCR. Medium. Available: https://medium.com/saarthi-ai/how-to-build-your-own-ocr-a5bb91b622ba

[14] Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun. (Jan 6, 2016). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. Available: https://arxiv.org/abs/1506.01497v3

[15] Ultralytics YOLOv8 Documentation. [Online]. Available: https://docs.ultralytics.com/

[16] Visual Studio Code. [Online]. Available: https://code.visualstudio.com/

[17] Computer Vision Annotation Tool (CVAT). [Online]. Available: https://www.cvat.ai/

[18] OpenCV Documentation. [Online]. Available: https://docs.opencv.org/4.x/index.html