Content-Based Lawsuits Document Image Retrieval

Chapter · September 2022

DOI: 10.1007/978-3-031-16474-3_3

Daniela Lopes Freire
University of São Paulo
26 PUBLICATIONS **66** CITATIONS
SEE PROFILE

Leonardo Carneiro Feltran
University of São Paulo
2 PUBLICATIONS **0** CITATIONS
SEE PROFILE

Kelly Ramos
University of São Paulo
2 PUBLICATIONS **0** CITATIONS
SEE PROFILE

João Eduardo Ferreira
University of São Paulo
152 PUBLICATIONS **1,246** CITATIONS
SEE PROFILE

Content-Based Lawsuits Document Image Retrieval

Daniela L. Freire[1](✉), André Carlos Ponce de Leon Ferreira de Carvalho[1], Leonardo Carneiro Feltran[1], Lara Ayumi Nagamatsu[1], Kelly Cristina Ramos da Silva[1], Claudemir Firmino[1], João Eduardo Ferreira[1], Pedro Losco Takecian[2], Danilo Carlotti[1], Francisco Antonio Cavalcanti Lima[2], and Roberto Mendes Portela[2]

[1] University of Sao Paulo, Sao Paulo, Brazil
{danielalfreire,andre}@icmc.usp.br,
{leonardo.feltran,lara.nagamatsu,kelly.ramos.silva,cfirmino}@usp.br,
{jef,danilopcarlotti}@ime.usp.br

[2] TJSP - Justice Court of São Paulo State, Sao Paulo, Brazil
plt@ime.usp.br, {franciscol,robertomp}@tjsp.jus.br

Abstract. The São Paulo Court of Justice has the highest number of lawsuits of all courts. The lawsuits are composed of raster-scanned documents enclosed in unstructured volumes, of which some are unreadable document images. Natural Language Processing techniques fail to extract from some of these documents due to the low quality of images. This article proposes a methodology to automatize the retrieval of document images from lawsuit databases based on the contents of the images. We developed a hybrid algorithm for feature extraction from document images and used a distance metric to retrieve similar images. The TJSP's database was used to validate our proposal, resulting in a system that allows finding similar images with an accuracy above eighty percent.

Keywords: Document image processing · Content-based image recognition · Deep learning techniques · Convolutional neural networks models · Feature extraction

1 Introduction

According to data from the report produced by the National Council of Justice [9], the São Paulo Court of Justice (in Portuguese, Tribunal de Justiça de São Paulo - TJSP) has the most significant number of lawsuits of all courts in the world, the congestion highest indicator (84%) and the longest average processing time (seven years and five months) among state courts in Brazil. Its lawsuits collection corresponds to 25% of the total lawsuits in Brazilian Justice, including lawsuits of the higher spheres, federal courts and superior courts. With that in mind, São Paulo Court has the largest workforce, comprising 2,500 magistrates and approximately 40,000 civil servants in 320 units. Despite the considerable

G. Marreiros et al. (Eds.): EPIA 2022, LNAI 13566, pp. 29–40, 2022.
https://doi.org/10.1007/978-3-031-16474-3_3

expense of the Judiciary and the public coffers, the grossing of the amount of money collected by the TJSP cases filed subject to the collection of costs in 2020 was R$ 2,021.92. [12].

The lawsuits are composed of raster-scanned documents in PDF files-usually, several kinds of documents per file enclosed in unstructured volumes without any digital index. These documents have perspective distortions and uneven illumination because they were captured in an unrestrained environment and produced by a flatbed scanner, fax machine, or mobile devices, such as smartphones or tablets. Usually, Natural Language Processing (NLP) tools are used for extracting text from, classification and processing document images. However, many lawsuit document images are illegible or of low quality, so it negatively affects the performance carried out by NLP tools because extracted texts from them usually contain errors. Content-based image retrieval (CBIR) is equivalent to providing human vision to the computer for similarity-based image retrieval. CBIR can classify image database images according to the degree of similarity with the query image [4]. Image retrieval depends on its content, known as features, which can be low-level or high-level. Low-level features include colour, texture, and shape, and high-level features describe the features captured by a human-like brain. Several features are extracted in the image recovery process to identify the image with more fidelity. Examples of other features extracted from images are colour histogram, colour averaging, colour structure descriptor and texture for feature extraction [7]. Deep learning techniques, especially Convolutional Neural Networks (CNNs), have been successfully used to improve the performance of feature extraction and the classification efficiency of CBIR [10]. Compared to other feature extraction and classification algorithms, CNNs do not require pre-processing. They are responsible for developing their filters (unsupervised learning), which is not the case with other more traditional algorithms. The lack of initial parameterization and human intervention is a significant advantage of CNN.

This article proposes a CBIR to automatize the retrieval of document images from lawsuit databases. We use a TJSP database composed of 2,136 unrecognized document images by NLP tools. The first contribution is a hybrid algorithm for feature extraction of document images that combines two traditional computer vision techniques for feature detection (ORB and HOG) with one CNN (MobileNet). The second contribution is a content-based retrieval tool that uses the hybrid algorithm, specific for lawsuit document images. We validated our proposal by performing several experiments, reaching an accuracy above 80% in proposed models, and statistically confirmed the results with ANOVA and Tukey statistical tests. The rest of this article is organised as follows: Sect. 2 exposes the proposal of a content-based lawsuits document image retrieval; Sect. 3 describe the application of proposal; Sect. 4 reports experiments and statistics to validate the proposal; and, Sect. 5 presents our conclusions.

2 Proposal

This section describes our proposal of a methodology for automating the retrieving of similar images from lawsuits databases. Our goal is to offer an alternative for document processing when extracting information from document images is not feasible by NLP tools, i.e., when images are entirely blurred images or with stains or scribbles. The retrieving similar images occurs in three steps: Preprocessing, Features Extraction, Similarity Search, and Result Presentation. Figure 1 shows these stages. Preprocessing step of the document processing

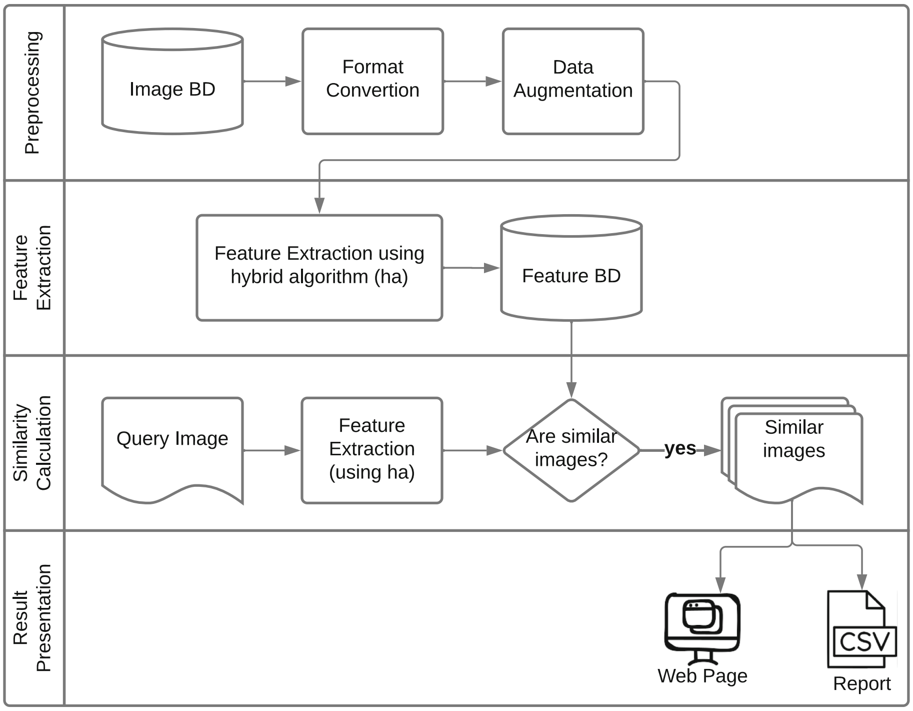

Fig. 1. Steps of retrieval.

pipeline has two activities: Format Conversion and Data Augmentation. In Format Conversion, the unrecognized documents images are read from a repository and converted from `.pdf` to `.jpeg` format. In Data Augmentation activity, since the unrecognized document images database is usually small (less than 1000 images), the data augmentation increases the database by applying several transformations such as rotation, width and height shift, shear, and zoom whitening, brightness, so on.

In the Features Extraction step, the features are extracted from all images and are stored as vectors in `.npy` format files. We developed a hybrid algorithm based on ORB, HOG and MobileNet algorithms for feature extraction of images. The algorithm looks for points of great importance and edges in the images and compiles them into a descriptor vector used to compare the images. The descriptor vectors are written as files in the `.npy` format that is used by the data structure. Then the `.npy` files, which contain the vectors for each image, are read and stored in a kd-tree [1] for future searches. The kd-tree is a data structure in a binary tree that arranges the elements based on their values in each feature vector dimension. It has an average complexity of $O(logn)$ for inserting, removing and searching elements, while in the worst case, it has a complexity of $O(n)$ for these same operations. In the example of Fig. 2, each level of the tree is related to a position of the feature vector, and at each level, the feature space is divided in two according to the median of the values in this position. In this way, the tree will always be balanced, allowing a search that takes practically the same time for all elements and does not require complex calculations to calculate the distances.

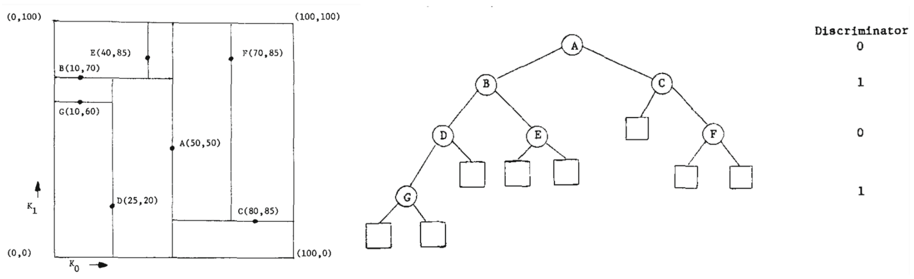

Fig. 2. kd-tree representation. Source: Bentley (1975) [1]

The Similarity Search step is performed using the kd-tree data structure, which will receive the vectors in `.npy` format and arrange them in a tree structure. This structure presents exemplary implementations in the Scikit-learn library [8] for Python, reducing the effort of its development. Two types of search can occur in the data structure, a search to find similar documents in the structure's database or a search based on a new image: *Query Image*. The first type results in a report on the relationship of which images present the most significant similarities to each other. In contrast, the second type presents a set of the N images most similar to the *Query Image*. The similarity between images is measured by the Euclidean distance similarity metric, comparing the feature vectors of the images to each other.

In the Result Presentation step, similar images and their similarity percentages are shown on a web page in the online process or on a report with this information saved into a file `.csv` format in the offline process.

3 Application

This section describes the application of the proposed CBIR in the São Paulo Court of Justice lawsuit document images. First, we contextualize how our research integrates with the document identification project in the TJSP environment and describe the database. Afterwards, we report the preprocessing and the feature extraction of the document images. Finally, we detail the image similarity search and the result presentation

3.1 Contextualization and Database

This experiment is part of the project to identify duplicates of the use of the same proof of payment of a lawsuit payment receipts issued by the TJSP, which gives evidence of fraud in the judicial system. The project's first stage seeks to identify document duplication through information obtained from texts extracted from images of scanned documents, using natural language processing techniques. The proposed pipeline deals with images of payment slip documents from which no information can be extracted and converted into texts. Usually, they are images whose text is unreadable or of low quality, containing much noise, low resolution, or even crumpled and poorly positioned document images.

The database used in the research project is composed of 139,603 processes that have 416,316 images of documents belonging to the 4th Civil Court of the Regional Forum XII - Nossa Senhora do Ó, in the city of São Paulo. Only 0.5% of the total number of documents were not recognized by the previous text extraction steps, leaving for our experiment 2,136 unrecognized document images, including lawsuit payment receipt images and other kinds of document images such as personal documents, reports, lawsuit decisions.

3.2 Preprocessing and Feature Extraction

We used Python Imaging Library (PIL) [3] to convert `.pdf` files to `.jpeg` format. PIL adds image processing capabilities to the Python interpreter and provides extensive file format support, an efficient internal representation, and fairly powerful image processing capabilities. After, we manually labelled the 2,136 unrecognized document images. As the number of the samples is small, We used the Keras ImageDataGenerator class that provides real-time data augmentation and ensures that the model receives new variations of the images at each epoch (or iteration) without adding them to the original image database. Furthermore, as the images are loaded in batches, much memory is saved.

We developed a hybrid algorithm for feature extraction based on HOG, ORB, and one CNN, MobileNet. We intended to combine the strengths of the three algorithms to obtain a more robust algorithm able to deal with the disturbances of document images. The internal structure of this algorithm is shown in Fig. 3. The input image format changes according to each algorithm. HOG uses greyscale and 400×400 size images. ORG uses greyscale and 600×600 size images. Both algorithms require a larger size to capture more minor features,

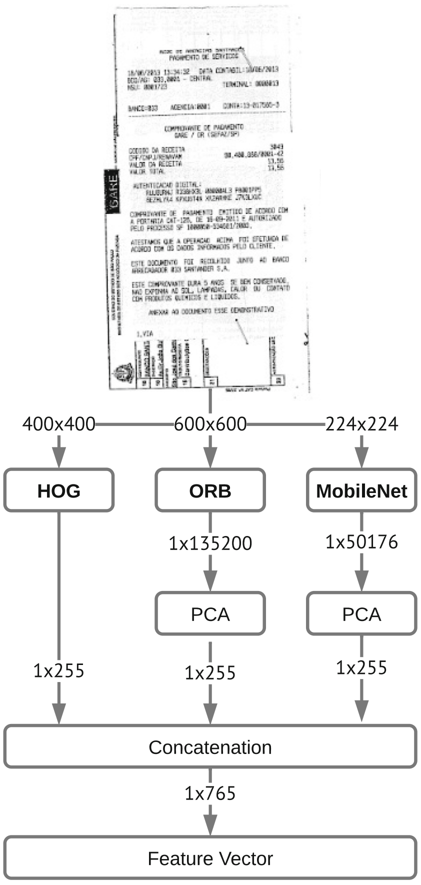

Fig. 3. Feature extractor diagram

such as small numbers and letters. MobileNet processing reduces the dimensionality of the images to a size of 224×224, which is the required size due to knowledge transfer, and colour images. In general, CNNs require inputs to have a specific value scale, as these scales lead to improved training. Keras library allows abstracting this part using a function that transforms the scale to the required format. The feature vectors produced for HOG, ORG, and MobileNet are 1×135200, 1X255, and 1×50176 sizes. As feature vectors from HOG and MobileNet have many positions, we carried out a reduction dimensionality of the HOG and MobileNet output using Principal Component Analysis, or PCA, the most popular technique for dimensionality reduction. PCA is the orthogonal projection of the data onto a lower-dimensional linear space, such that the variance of the projected data is maximized [2]. All output feature vectors with the same size were concatenated, resulting in a vector with 765 positions which is normalized and then converted into a `.npy` format file.

3.3 Similarity Calculation and Result Presentation

The proposal CBIR can work out online or offline processes. In both processes, The `extractor.py` script extracts deep features from each database document image of the dataset using our hybrid algorithm. It stores `.npy` format files. In the online process, we developed a simple image-based image search engine using Keras and Flask, a micro web framework written in Python. The `server.py` script runs a web server, in which a query image is provided as input to the server via a Flask web interface. The server extracts the in-depth features from query images and compares them with dataset images using the Euclidean distance metric [11] to retry similar images. The `server.py` algorithm's output is a predefined number of best similar images. On the other hand, in an offline process, the `output.py` script reads a set of query document images from an input path, extracts deep features from these images, compares them with dataset images, finds a predefined number of best similar images, then returns a file `.csv` with query images and their respective similar images.

We tested the proposal CBIR using the dataset composed of 2,136 unrecognized document images to determine if there were images with 100% of similarity. The CBIR returned 32 images identified as equal to human experts.

4 Experiments

This section describes the experiments carried out to develop and evaluate the algorithm for feature extraction. First, we describe an experiment carried out to choose the CNN. After, we describe an experiment for hybrid algorithm evaluation.

4.1 CNN's Choice

We tested four CNNs: VGG16, ResNet50, MobileNet, and MobileNetV2. We performed an experiment with a set of 3448 document images from TJSP. Metrics measured were: execution time, CPU time, percentage of memory usage and accuracy. The execution time comprises the time elapsed to search for a more similar image. User CPU time is the CPU time spent executing the algorithm. The memory usage percentage is the percentage of the total memory size used in executing the algorithm. Accuracy indicates the number of correctly predicted cases concerning all existing cases. Table 1 summarizes the average of the values in the simulations, where the time unit of execution time and CPU time is second. We repeated the execution of each tested algorithm 30 times, collected the metrics and then performed tests to verify if there was a statistically significant difference between the CNNs. The tests applied were ANOVA and Tukey HSD test with a significance level of 0.05. The ANOVA test is a statistical technique that allows differentiating between the variations found in a set of measurements of an experiment derived from random factors due to real differences between the sources of variations (CNNs) under analysis. The parameter *F-value*, in the

ANOVA test, is a resource to find out if the variation between the population means is statistically significant. The *F-value* is a probability in which if its value is less than 0.05; it means that there is a statistical difference between the treatments (our CNNs). The *P-value* is a parameter that indicates the significance of the adding model terms. When there is such a difference in the experiment results, there are multiple comparison techniques to determine which treatments differ and which are similar. One technique is to use the Tukey HSD Test.

Table 1. Metric average of the convolutional neural networks

Metrics	CNNs			
	ResNet50	Vgg16	MobileNet	MobileNetV2
Execution time	**355.50**	365.83	362.40	363.20
CPU time	**4,893.33**	12,154.94	19,520.06	11,393.10
% Memory	23.03	23.03	**22.65**	23.35
Accuracy	0.86	0.85	**0.87**	0.87

Table 2 presents execution time analysis of variance (ANOVA). The resulting values for the *F-value* and the *P-value* were, respectively, 13.796 and 2.8523e–07. The ANOVA from execution time presents the sum of squares as 1,165.28 for the CNNs and 2,139.78 for error. The freedom degree was 3 for the CNNs and 76 for error. Table 3 presents CPU time analysis of variance. The resulting values for the *F-value* and the *P-value* were, respectively, 168.628 and 1.675e–33. The ANOVA from CPU time shows the sum of squares was 2.148961e+09 for the CNNs and 3.228426e+08 for error. The freedom degree was 3 for the CNNs and 76 for error.

Table 2. Execution time

Sources of variation	Degree of freedom	Sum of squares	*F-value*	*P-value* (>*F-value*)
CNNs	3.0	1,165.28	13.796	2.8523e–07
Error	76.0	2,139.78		

Table 3. CPU time

Sources of variation	Degree of freedom	Sum of squares	*F-value*	*P-value* (>*F-value*)
CNNs	3.0	2.148961e+09	168.628	1.675e–33
Error	76.0	3.228426e+08		

Table 4 presents memory percentage analysis of variance. The resulting values for the *F-value* and the *P-value* were, respectively, 21.128 and 4.706e–10. The ANOVA from memory percentage shows the sum of squares was 0.820917 for the CNNs and 0.984293 for error. The freedom degree was 3 for the CNNs and 76 for error. Table 5 presents accuracy analysis of variance. For the CPU time, the resulting values for the *F-value* and the *P-value* were, respectively, 1,497.11 and 1.76e–67. The ANOVA from accuracy shows the sum of squares was 0.002634 for the CNNs and 0.000045 for error. The freedom degree was 3 for the CNNs and 76 for error.

Table 4. Memory percentage

Sources of variation	Degree of freedom	Sum of squares	*F-value*	*P-value (>F-value)*
CNNs	3.0	0.820917	21.128	4.706e–10
Error	76.0	0.984293		

Table 5. Accuracy

Sources of variation	Degree of freedom	Sum of squares	*F-value*	*P-value (>F-value)*
CNNs	3.0	0.002634	1,497.11	1.76e–67
Error	76.0	0.000045		

Table 6, Table 7, Table 8, and Table 9 show the pairwise comparison of CNNs with Tukey HSD confidence intervals with a test significance level of 0.05. The group1 and group2 columns are the formed groups; the meandiff column is pairwise mean differences; p-adj is the adjusted *P-value* from the HSD test; the rejected column is "True" if we reject Null for group pair.

Table 6. Execution time

CNNs				
group1	group2	meandiff	p-adj	reject
MobileNet	MobileNetV2	0.7949	0.9000	False
MobileNet	ResNet50	–6.9039	0.0010	True
MobileNet	Vgg16	3.4298	0.1812	False
MobileNetV2	ResNet50	–7.6988	0.0010	True
MobileNetV2	Vgg16	2.6348	0.4029	False
ResNet50	Vgg16	10.3337	0.0010	True

Table 7. CPU time

CNNs				
group1	group2	meandiff	p-adj	reject
MobileNet	MobileNetV2	–8126.9605	0.0010	True
MobileNet	ResNet50	–14626.7325	0.0010	True
MobileNet	Vgg16	–7365.126	0.0010	True
MobileNetV2	ResNet50	–6499.772	0.0010	True
MobileNetV2	Vgg16	761.8345	0.6313	False
ResNet50	Vgg16	7261.6065	0.0010	True

Table 8. Memory percentage

CNNs				
group1	group2	meandiff	p-adj	reject
MobileNet	MobileNetV2	0.1399	0.0012	True
MobileNet	ResNet50	–0.0894	0.0706	False
MobileNet	Vgg16	–0.1214	0.0063	True
MobileNetV2	ResNet50	–0.2293	0.001	True
MobileNetV2	Vgg16	–0.2613	0.001	True
ResNet50	Vgg16	–0.0319	0.7876	False

Table 9. Accuracy

CNNs				
group1	group2	meandiff	p-adj	reject
MobileNet	MobileNetV2	–0.0002	0.7151	False
MobileNet	ResNet50	–0.0084	0.001	True
MobileNet	Vgg16	–0.0136	0.001	True
MobileNetV2	ResNet50	–0.0081	0.001	True
MobileNetV2	Vgg16	–0.0134	0.001	True
ResNet50	Vgg16	–0.0052	0.001	True

The experiment with the simulations for extraction features indicated that ResNet50 reached the best performance in terms of execution and CPU time. In contrast, MobileNet reached the best memory percentage and accuracy performance, c.f. Table 1. The lowest execution time average was 355.50 s with the ResNet50, but others reached similar values for this metric. The ANOVA showed a statistical difference, and the Tukey averages comparison test formed two groups: one with ResNet50 and another with the other CNNs, c.f. shown in Table 2 and Table 6. The lowest CPU time average was 4,803 s with the ResNet50. The ANOVA showed a statistical difference, and the Tukey averages comparison test formed two groups: one with ResNet50 and another with the other CNNs, c.f. shown in Table 3 and Table 7.

4.2 Hybrid Algorithm Evaluation

We carried out an experiment to evaluate the hybrid algorithm performance regarding the performance of each pure algorithm. We randomly selected 300 document images and applied five distortions in images: rotation, blurring, gamma level change, shift, and zoom level. The tests were performed by populating the data structure with unmodified images, while the query images were of modified images. The accuracy was collected for each level or type of distortion analyzed. Figure 4 presents the results for the four analyzed algorithms, the HOG, ORB, MobileNet and the hybrid algorithm. The best accuracy was reached by ORB, followed by the hybrid algorithm regarding rotation. HOG reached the best accuracy regarding blurring, followed by the hybrid algorithm. All algorithms reached similar accuracy regarding gama level change, shift, and zoom level.

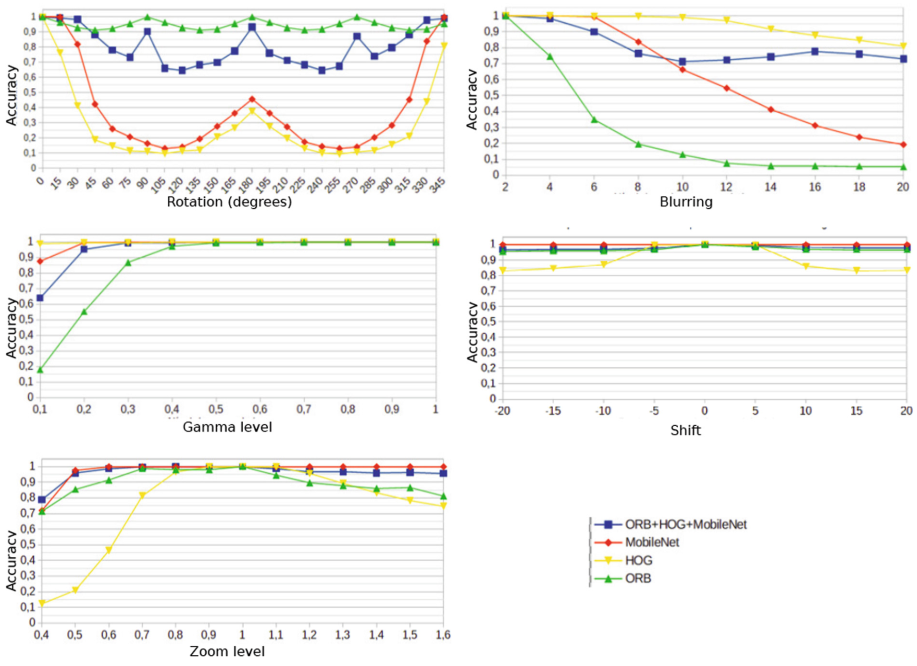

Fig. 4. Algorithm evaluation about distortions

We also applied five types of noises in images: Gaussian Noise, Localvar Noise, Poison Noise, Salt Noise, and Pepper Noise. The Gaussian Noise is a randomly generated noise value that affects the image pixels. In a Gaussian Noise added image, both the true pixel values and random Gaussian noise added values exist. It spreads the noise values by using the Gaussian probability density function [5,6]. Localvar noise is a zero-mean Gaussian white noise. The difference between Gaussian and Localvar noise is that the former is independent of the pixel intensity of an image, and the latter is dependent [13]. Poison Noise depends on the measurement of light, photon direction and quantized nature of light [14]. Salt and Pepper are types of noise that combine black and white spots in an image. Salt noise is added to an image by adding random brightness all over the image. Pepper Noise modifies the image by adding random dark spots [5]. The accuracy collected for each noise analyzed is shown in Fig. 5 for the four analyzed algorithms, the HOG, ORB, MobileNet and the hybrid algorithm. There is also a significant performance increase in noisy images when using the proposed model, being very close to the performance of MobileNet, the best in this regard, reaching a better performance for Salt and Pepper types of noise. The hybrid algorithm generally showed greater robustness than the pure algorithms when considering both distortions and noise, although the accuracy was not the best in every case.

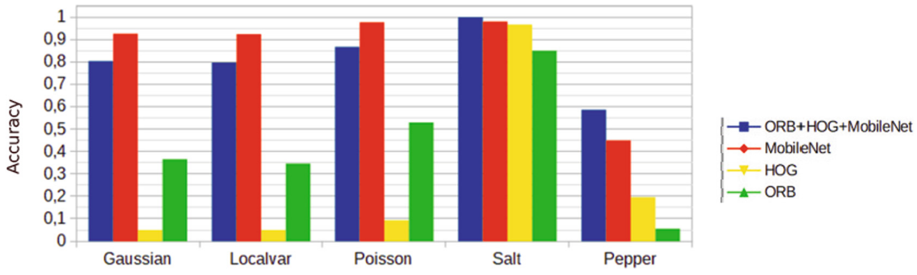

Fig. 5. Algorithm evaluation about noises

5 Conclusion

The lawsuit's document images often contain a lot of noise and distortions because they are captured in an unrestrained environment by different devices such as flatbed scanners, fax machines, smartphones, or tablets. The classification performance of these images with low quality is poor when NLP tools. We proposed a content-based lawsuits document image retrieval to search similar document images to solve this problem. The CBIR contains an algorithm for feature extraction based on three algorithms: HOG, ORB, and one CNN, MobileNet. We carried out exhaustive experiments to choose the best methodologies and techniques for developing the algorithm using the TJSP database. We concluded that the algorithm proposed has greater robustness in extracting features from images with distortions and noise than the pure algorithms.

References

1. Bentley, J.L.: Multidimensional binary search trees used for associative searching. Commun. ACM **18**(9), 509–517 (1975)
2. Bishop, C.M., Nasrabadi, N.M.: Pattern Recognition and Machine Learning. Springer, Heidelberg (2006)

3. Clark, A.: Pillow (pil fork) documentation. Readthedocs (2015). https://Buildmedia.Readthedocs.Org/Media/Pdf/Pillow/Latest/Pillow.Pdf
4. Gudivada, V.N., Raghavan, V.V., Vanapipat, K.: A unified approach to data modeling and retrieval for a class of image database applications. In: Multimedia Database Systems: Issues and Research Directions, pp. 37–78. Springer, Heidelberg (1996). https://doi.org/10.1007/978-3-642-60950-3_2
5. Hoshyar, A.N., Al-Jumaily, A., Hoshyar, A.N.: Comparing the performance of various filters on skin cancer images. Procedia Comput. Sci. **42**, 32–37 (2014)
6. Luisier, F., Blu, T., Unser, M.: Image denoising in mixed poisson-gaussian noise. IEEE Trans. Image Process. **20**(3), 696–708 (2010)
7. Pattanaik, S., Bhalke, D.: Beginners to content-based image retrieval. Int. J. Sci. Eng. Technol. Res. **1**, 40–44 (2012)
8. Pedregosa, F., et al.: Scikit-learn: machine learning in python. J. Mach. Learn. Res. **12**, 2825–2830 (2011)
9. de Justiça Departamento de Pesquisas Judiciárias, C.N.: Justiça em números 2021. Justiça em números 2021 (2021 [Online])
10. Rian, Z., Christanti, V., Hendryli, J.: Content-based image retrieval using convolutional neural networks. In: 2019 IEEE International Conference on Signals and Systems, pp. 1–7. IEEE (2019)
11. Sergyan, S.: Color histogram features based image classification in content-based image retrieval systems. In: 6th International Symposium on Applied Machine Intelligence and Informatics, pp. 221–224. IEEE (2008)
12. de Tecnologia da Informação do Tribunal de Justiça de São Paulo, S.: Tribunal de justiça - estado de são paulo: a justiça próxima do cidadão (2022). https://www.tjsp.jus.br/QuemSomos
13. Tin, H.H.K.: Removal of noise by median filtering in image processing. In: Parallel and Soft Computing, pp. 1–3 (2011)
14. Unser, M.: Texture classification and segmentation using wavelet frames. IEEE Trans. Image Process. **4**(11), 1549–1560 (1995)