DIGITAL FORENSIC RESEARCH CONFERENCE

Using Micro-Services and Artificial Intelligence to Analyze Criminal Evidences

By:

Iaslan Silva (Federal University of Rio Grande do Norte),

João Marcos Valle (Federal University of Rio Grande do Norte),

Gabriel Souza (Federal University of Rio Grande do Norte),

Jaine Budke (Federal University of Rio Grande do Norte),

Daniel Araújo (Federal University of Rio Grande do Norte),

Bruno Carvalho (Federal University of Rio Grande do Norte),

Nélio Cacho (Federal University of Rio Grande do Norte),

Henrique Sales (Federal University of Rio Grande do Norte),

Frederico Lopes (Federal University of Rio Grande do Norte),

and Rivaldo Silva Júnior (Ministerio Publico do Rio Grande do Norte)

From the proceedings of

The Digital Forensic Research Conference

DFRWS USA 2021

July 12-15, 2021

Forensic Science International: Digital Investigation 37 (2021) 301197

Contents lists available at ScienceDirect

Forensic Science International: Digital Investigation

journal homepage: www.elsevier.com/locate/fsidi

DFRWS 2021 USA - Proceedings of the Twenty First Annual DFRWS USA

Using micro-services and artificial intelligence to analyze images in criminal evidences

Iaslan Silva [a], João Marcos do Valle [a, *], Gabriel Souza [a], Jaine Budke [a], Daniel Araújo [a], Bruno Carvalho [a], Nélio Cacho [a], Henrique Sales [a], Frederico Lopes [a], Rivaldo Silva Júnior [b]

[a] Digital Metropolis Institute, Federal University of Rio Grande Do Norte, Natal, Brazil
[b] Ministério Público Do Rio Grande Do Norte, Brazil

ARTICLE INFO

Article history:

Keywords:
Computer vision
Digital forensics
Crime evidence
Machine learning
Microservice

ABSTRACT

With the advancement of digital crimes, the field of digital forensic science grows more and more, and with this growth, the search for faster and more accurate solutions to aid the investigation process becomes a necessity. In the context of the Brazilian judicial system, during a criminal investigation, forensic specialists extract, decode, and analyze the evidence collected to allow the prosecutor to make legal demands for a prosecution. These specialists have a very short time to analyze to find criminal evidence and the process can take a long time. To solve this problem this paper proposes to use a micro-services-based application with artificial intelligence to process large amounts of images contained in criminal evidence using open-source software. The image classification module contains some pre-trained classifiers, considering the needs of forensic analysts of the Rio Grande do Norte District Attorney's Office (MPRN). The models were built to identify specific types of objects, for example, firearms, ammunition, Brazilian identity cards, text documents, cell phone screen captures, and nudity. The results obtained show that the system achieved good accuracy in most cases. This is extremely important in the context of this research, where false positives should be avoided in order to save analysts' working time. Moreover, the proposed architecture was able to speed up the image classification process using Apache Spark.

1. Introduction

In the Brazilian justice system, for a prosecutor to establish legal requirements for an indictment, it is usually necessary that forensics specialists extract, decode and make an analysis of collected evidences (cell phones, tablets, hard drives, etc.). Commonly, the collected media consists of a large amount of data, while leads (important evidences that help to solve a crime) are usually found in specific files. The analyst may have to verify all the evidence to find a single lead. In addition, the analyst usually has access to the evidence just a few days before the ruling of a judge, and so, the available time to the analysts is relatively short. Thus, the speed that the criminal evidences are analyzed is of paramount importance. In this context, it is necessary a system that gives indications of files that can present leads to the analysts, such as suspicious images, and deliver a faster result.

Additionally, with the overall cheapening of data storage and technological advances, mobile device storage capacity has increased by a large margin in the past years. This results in more data present in these devices when they are apprehended. And, there is also the case that in an investigation, several devices (sometimes multiple dozens) are apprehended in a single operation, having huge amounts of data for forensics analysts to analyze one by one. That not being enough, the evidence found in those devices can lead to new investigations, resulting in the apprehension of more devices, which can easily become a big data problem. Also, with the constant usage of new technology, people will become more versatile in it, using new methods of communication, such as through images and screenshots instead of just text messages, and several others wanting to deceive criminal investigators and forensic analysts. And finally, since there is no digital triage method, the analysts can be exposed to several psychologically disturbing images and videos, with no previous warning and no way to prepare themselves for what is to come.

Given these problems, the State Attorney's Office of the state of

* Corresponding author.
E-mail address: joaomarcos@ppgsc.ufrn.br (J.M. do Valle).

https://doi.org/10.1016/j.fsidi.2021.301197

Rio Grande do Norte (MPRN) together with the Federal University of Rio Grande do Norte (UFRN) created the project INSIDE (*Integration, aNalySis, vIsualization of Data for invEstigation*). Such project developed the INSIDE platform, a big data platform for crime analysis capable of powering scalable, real-time, and data-driven applications. This platform was conceived to process big data in the context of criminal evidence and speed up criminal evidence analysis procedures, a very time-consuming process.

This work proposes the development of a microservice architecture for the image classification module. The data used in the tools are extracted from digital evidences at the Forensic Laboratory of the prosecutor. In the development of this software, the following steps were performed: the creation of datasets with examples of images that represent the classes to be used, the selection and use of appropriate existing classifiers, the application of OCR (Optical Character Recognition) techniques for extracting information from documents, and the development of a facial recognition module, nudity module.

This paper is organized as follows: Section 2 presents some related works in object detection, while Section 3 describes the INSIDE project. In Section 4, we explain the datasets creation, present the classification process and workflow. Section 5 discusses the results of this work. We present our conclusion and point to future works associated with this architecture in Section 6, and finally, the acknowledgement in section 7.

2. Related works

There are many works related to object detection and classification tasks. However, most of them focus on specific objects. Lee et al. (2012) and Han and Jain (2013) propose a biometric model using computer vision to detect and recognize tattoos in individuals. Tattoos are a good instrument in Forensic Science for performing human identification. This refers to the fact of tattoos are registered at the time, scratches, and burns. The dataset used by the authors contains 64,000 images of tattoos. They use CBIR (Content-based image retrieval), an application that scans huge databases to find images that have a certain resemblance to an input image. The images were used in supervised algorithms, and they obtained results with an accuracy rate of 57%.

Carvalho et al. (de Carvalho et al., 2015) present a set of techniques and methodologies to assist the investigation of images. Within the evidence data, an analyst will search for images that prove suspicious behaviors, for example, by looking for pornographic, fake, or forged images. To locate these images, the authors propose some techniques for detecting false images of documents or photomontages, pornographic or child pornography images, and attempts to perform the attribution of sources (verify if an image has a source document).

Brown et al. (Brown et al. (2005) propose a forensic image mining system using a modeling of Component-Based Objects and Scenes to detect and filter out indecent images. They used an SVM classifier to identify certain patches of an image. In their experiments, they tested their classifier using the HSV and YCbCr spaces. In the tests with HSV, the system found 92% of true positive images and 74% of true negative detection, while in the tests using YCbCr space, the system was more efficient in detecting true negative images (95% of negative detection and 79% of positive detection).

Grega et al. (2015) propose a detection system of firearms and knives for CFTV, notifying the operator when one of these objects was detected near of hands or silhouette of a person. The datasets of knife and firearms detection were obtained from CCTV recordings. In their experiments, their method achieved a false alarm rate of 7%, while achieving a specificity of 95% and sensitivity of 81%.

An approach to process large amounts of data is shown by (Hoelz et al., 2009), which employed Artificial Intelligence (AI) techniques to analyze data collected from criminal investigations. In this case, the AI techniques used were multi-agent systems. This system has its objectives to analyze and correlate evidence data, reducing the amount of data required to be verified by the analyst. Results show that the system has the same efficiency as an analyst.

Social media has grown in the past decade. The enormous amount of data shared online represents a valuable source of evidence to investigations if treated properly. Moltisanti et al. (2015) propose a pipeline to explore the mechanisms used to upload images. Results show that in most cases knowing the process used to upload media allows inference of evidences related to authentication and integrity of data. Korus and Memon (1902) address the optimization of the entire data flow from the acquisition to the distribution of images to facilitate the forensic analysis. They present a neural network that can be trained to replace the photo creation and high-fidelity photo rendering pipeline. The neural network approach provided an increased detection of manipulated images.

In 2014, Van Baar (Van Baar et al., 2014) described the process of digital forensics and explained how the approach *Digital Forensics as a Service* (DFaaS), developed in the Netherlands Forensic Institute, helps forensic analysts in the investigation. This approach was based on preview experiences and paved the way for HANSKEN, a big data platform for forensic analysis. Similarly, other tools emerged to facilitate digital evidence analysis. To prioritize relevant data in evidences, the concept of digital triage is key. Sanchez reports that digital triage is not very suitable for complex cases, but also argues that the constant exposure to abusive materials can bring psychological problems for forensic analysts and investigators. Also, the usage of techniques such as the ones proposed by Tahir (Tahir and Iqbal, 2015) can make the investigation process faster and easier for forensic analysts. We argue that digital triage, or automated digital evidence analysis, is fundamental to lessen the burden on analysts.

Furthermore, big data solutions for forensic analysis presented in recent work (Van Beek et al., 2015) show reasonable justification for the usage of big data tools such as Apache Kafka (Thein, 2014; Narkhede et al., 2017) and Apache Spark (Chambers and Zaharia, 2018; The Apache Software Found, 2019).

In previous work, INSIDE (do Valle et al., 2020) proposed a big data platform for processing criminal evidences. Using in it, a lambda architecture which utilizes a set of tools and structures, among them Apache Hadoop HDFS, Apache Spark, and Apache Kafka to process a large amount of data. In the experiments, it was possible to notice positive feedback in the usage of big data tools to process information contained in criminal evidences.

The aforementioned works bring different approaches to image processing. Some are focused on the optimization of processing speed, using various methods such as parallel processing. Others are focused on the classification process of images in digital evidences. It is known that few of the papers found in the literature focus on the classification and detection of important objects for the forensics digital area. To our knowledge, little has been done in the attempt to build a platform capable of timely processing large quantities of image data present in digital evidences to aid forensic analysts in the investigation process. More than that, it is very hard to find an open-source solution to deal with this problem. To address that, this work proposes an architecture using a complete open-source set of tools for image parallel processing and convolutional neural network models to classify large amounts of images stored in digital evidences in a timely manner. The proposed architecture uses Apache Kafka (Thein, 2014; Narkhede et al., 2017) to receive and forward images in the pipeline, and Apache

Spark (Chambers and Zaharia, 2018; The Apache Software Found, 2019) to deliver faster processing using cluster computing. Also, it uses use several convolutional network algorithms to detect relevant objects in images, such as firearms, knives, people, bank receipts, and others.

3. Microservice architecture

Digital evidence visualization tools demand much computational power. When running on a local machine, these tools can take too much time to load and even make the analyst's computer unusable. Moreover, we can have thousands of images to be analyzed in an evidence which already could be very time-consuming on a reasonably up-to-date computer, would be even worse on a slow computer. To address this kind of problem, we proposed the development of an architecture capable of efficiently handle digital evidence images and classify them.

As digital evidences have a large range of objects of interest, the image classification process of those evidences requires different applications, or services, that have to be capable of detecting those different objects. Considering that large range, we chose to use a micro-service-based architecture. The micro-services architecture is an approach to developing an application as a set of small independent services. Each of the services is running in its independent process. Services can communicate with some lightweight mechanisms (usually something around HTTP) (Uckelmann et al., 2011). But in the context of digital evidences, there are others factors to consider. First, there are large quantities of data to be analyzed, and the metadata attached to each image is also very important to the forensic analysts, so it cannot be dismissed. Furthermore, there is also the problem of sequential image classification, for example, if a human is detected in an image, that image will need further classifications, such as nudity classification, or a face recognition process. These factors combined with changing IP addresses from hosts and net load balance issues, make HTTP not a viable choice. In this sense, we chose to use Apache Kafka (Narkhede et al., 2017; Thein, 2014) as our message handler and broker, as we feel it makes the process less chaotic.

Apache Kafka is a scalable messaging system, with its main architecture being that of a distributed commit log, suitable for offline and online (Narkhede et al., 2017; Thein, 2014). In recent years, Kafka has been used as a data ingestion tool, to insert data flows into processing platforms (Narkhede et al., 2017). As Kafka is a platform of distributed systems, there may be several clusters of Kafka servers (brokers), in charge of handling messages and replicating the data transferred in the pipeline. Two of Kafka's main APIs are the Producer (Producer) and the Consumer (Consumer), used to publish and receive message flows, respectively.

As evidences present a great amount of data and there is also a necessity for real-time processing, we opted to use Lambda Architecture. This type of architecture is capable of handling online, real-time, and also batch processing. A Lambda Architecture has as its main objective the building of a big data system organized in layers (speed layer, serving layer, and batch layer) (Marz and Warren, 2015). The image classification process happens at the speed layer, as this layer is focused on real-time and fast processing. To achieve fast results, some of the classification modules are executed using Apache Spark (The Apache Software Found, 2019), an open-source tool for cluster processing. We argue that the remote image classification will be faster in a Spark cluster if compared to a local serial processing environment.

The image classification in the speed layer happens in an asynchronous manner, as the images are processed even if they are not already available for the analysts in the serving layer. As long as the evidence is streamed to the pipeline, the images will be processed, and the results will be available in a Kafka topic to be consumed and stored in the application databases. A Kafka topic works as a channel where a Kafka Producer will send specific data to, and a Kafka Consumer will subscribe to this channel, consuming all data in it via a streaming API. It is also important to notice that, in our pipeline, all images are saved in a data lake in the form of a Hadoop HDFS cluster, guaranteeing that they can be accessed at later times if necessary.

It is important to notice that the architecture described supports any kind of image processing or data extraction. We can exemplify the general approach by the following steps:

1. The working flow of the module begins from a message received by a Kafka (Narkhede et al., 2017) consumer from an “Image” topic. The consumed message consists of an image (in a base64 format) that has yet to be classified and the image's metadata.
2. The message is then processed by an Image Classification API, where it will be processed by a Deep Learning (DL) algorithm. This algorithm is trained using a specific dataset to detect illicit objects.
3. In this API, each image goes through a series of classifiers. If a relevant object is detected, the image receives a tag corresponding to the object class. It is possible for an image to belong to more than one class.

For example, if the image is classified as a document, it will be sent to an OCR (Optical Character Recognition) module, where all texts will be extracted and stored for further processing. Or, if in the image, a person object is detected, it will be sent to the Facial Recognition Module, which is trained exclusively for the detection of suspects. In this stage, the person's face will be cropped, processed, and compared to the faces present in a database and saved in it.

4. After the classification process, images are sent to a visualization screen where the analyst can see and work with the classified images.

The Apache Spark (The Apache Software Found, 2019) is an open-source framework for general-purpose cluster computing. Spark provides an API that enables the complete usage of computational cluster power, with parallel processing and fault tolerance. It supports the most common programming languages, such as Python, Java, Scala, and R, and it also includes libraries for multiple useful tasks, such as SQL support, data streaming, and Machine Learning techniques. Spark can run in single or large-scale server environments (Chambers and Zaharia, 2018). Big companies, including Facebook, Microsoft, Netflix, and LinkedIn, are successfully using Spark-based solutions inside their main streaming applications (Luu, 2018).

To improve modularity, the application is composed of several layers, each with a specific purpose. The data workflow within the platform is as follows: data extracted from digital evidence is sent to an Import Module, which provides the transformed data and metadata to a Lambda Layer. The Import Module is capable of handling different types and sizes of data.

3.1. Import Module

The Import Module is responsible for extracting and preprocessing evidence data. This module receives a media extraction in XML format as input. This file contains two types of data: (i) an XML file containing all metadata collected from the evidence, along with all textual data present in the evidence, and (ii) a directory tree with a variety of other types of data from the

evidence (images, audios, videos, databases, configuration files, etc.). The extracted data and metadata from the evidence are then sent to the Speed Layer.

3.2. Speed layer module

The Speed Layer consists of an Apache Spark cluster for fast processing and specialized applications tasked with handling different types of data. The architecture of the Spark cluster is composed of a master node and working nodes. The master node is responsible for handling processing requests sent to it by specialized applications, scheduling operations on the cluster, managing the processing power of the cluster, and returning processing outputs. The worker nodes receive data from the master node in RDD format, a type of data stored in the distributed memory of the cluster. The workers also receive parallel operations required for execution on the RDDs. The operations are executed using the maximum processing power available in the cluster, thus speeding up the process. At this layer, data is sent to the Spark master using Kafka messages.

3.3. Classification module

The classification module is responsible for classifying images within the evidence. The classification module is composed of several microservices where each microservice is responsible for one object. Present in this module is APIs for classification, object detection, face recognition, text character detection, nudity detection, and age group detection.

4. Materials and methods

4.1. Datasets

The usage of a convolutional neural network to solve classification problems need a great volume of data. Thus, one of the more important phases to obtain better results is the build of a good dataset.

MPRN needs to analyze and classify a great number of images, that are obtained from criminal evidence. This evidence usually contains bank transaction information, geo-referenced data, firearms images, or object images that prove the involvement of the suspect in the investigated crime. There are many classifiers and detectors that can be used, such as YOLO (Redmon et al., 2016; Redmon and Farhadi, 1804) and Fast-RNN (Girshick, 2015). However, some important classes for this project are not pre-trained. Thus, to detect the objects of interest we had to build our own dataset. The datasets contain images of firearms, ammunition, paper documents, the Brazilian identification document (ID cards), and chats screenshots. These classes were defined by the analysts according to their necessity.

The final dataset was composed of 7034 images of fire guns, 2200 images of ammunition and 1200images of ID cards. Nenhuma imagem negativa foi utilizada para compor o dataset desses classificadores.

For the Document classifiers, were used 13,220 document images and for the Screenshots classifiers, we used 8203 images of screenshots of chats. The datasets of both classifiers were, still, incremented with 9495 negative images. Both were trained considering 2000 images (from each dataset: class and negatives) used for validation, and the rest for training.

The training datasets used for the classifiers were collected using three different strategies: (1) keywords searches in Google referring to the searched object, (2) parts of news reports/movies available on YouTube, and (3) images in datasets already collected available with open access.

The collection of Document and IDs datasets was made using strategy (1) and the Screenshots dataset with strategies (1) and (3). For the Firearm and Ammunition datasets, were incorporated images gathered with the three listed strategies. And, lastly, for Nudity and Facial Recognition classifiers, was used a trained model, which did not need a dataset collection.

4.2. YOLO

YOLO (*You Only Lock Once*) (Redmon et al., 2016; Redmon and Farhadi, 1804) is a real-time image detector. This tool is a **convolutional neural network** that addresses the detection task as a regression problem. Neural models like *R—CNN*, *Fast R—CNN* (Girshick, 2015) and *SSD (Single Shot MultiBox Detector)* (Liu et al., 2016) give a new purpose for classifiers to perform object detection. These systems use a classifier for a given object and evaluate it at various locations and scales in a test image. Other systems like *DPM (Deformable parts models)* (Felzenszwalb et al., 2009) use a sliding window approach, in which the classifier interacts in equally spaced locations throughout the image. The *R—CNN* and *Fast R—CNN* (Girshick, 2015) use a different approach, analyzing the region of interest in the image to generate bounding boxes in the image and then use a classifier on those images. After sorting, post-processing is used to refine these bounding boxes by eliminating duplicate detection and re-evaluating those boxes based on other objects in the image. These complex *pipelines* are slow and difficult to optimize since each component must be trained separately.

In contrast, YOLO is a single deep neural network that deals with all these tasks. YOLO is more efficient because it uses a simplified approach, i. e., instead of revisiting the image several times to adjust multiple systems that after a lot of costs (time and processing) perform the task, it analyzes the image only once to determine which objects are present in the image and where they are.

In this work, we used some pre-trained classes from YOLO (Redmon et al., 2016; Redmon and Farhadi, 1804) and made use of models obtained from it to train new classes. The classes used by YOLO are person, bag, handbag, clocks, car, motorcycle, truck, boat, knife, notebook, smartphones, and computer. The additional classifiers developed in this paper were: ammunition, Brazilian identification document, and firearms.

4.3. Inception

Inception V3 was developed based on improvements to the GoogleNet architecture, being the first model to allow CNN layers not to be executed in sequence (Rodrigueset al.). The presence of modules called “Inception” which are responsible for extracting features through convolutional networks, whose function is to learn rich representations of information with few parameters. As a standard, a convolutional layer applies filters in a three-dimensional space, in the case of using images in the RGB color space, for example, we have the height, width, and depth (Ribeiro and Francisco de Paula, 2019).

In this work, we used the transfer learning technique, implemented with the aid of the TensorFlow library for extracting features, using a pre-trained model with Inception V3 (Szegedy et al., 2016) trained with the ImageNet dataset. In this case, the weights of the initial layers are frozen and the network is retrained with the images collected from Document and Chats Screenshot, previously described.

4.4. NudeNet

With the advent of several Deep Learning libraries and a large number of open-source implementations and documents, Image Classification has gained an excellent ally (Bedapudi, 2021). The automatic identification of images that are not suitable/safe for work (NSFW), is an important problem that has been researched and solved by several researchers. One project that is relevant on the open-source scope is the open_NSFW from Yahoo, however, this project is out of date (Mahadeokar and Pesavento, 1511).

NudeNet is an open-source neural network for the classification, detection, and censorship of nudity in images. For the implementation of this library, there was a great deal of work to collect these images and train them to make a good model for the classification of nudity and NSFW. For this project, the NudeNet model is used to assist in the identification of crimes of sexual abuse and pedophilia.

4.5. Face recognition

Facial recognition or face detection consists of identifying patterns and characteristics present on the human face, characteristics such as the shape of the mouth, face, the distance between the eyes. Computer Vision, as already mentioned, aims to replicate human vision, a vision that is able to recognize human faces even in adverse conditions such as high shading, high distance, obstacles in front, or only partial vision (Silva and Cintra, 2015).

In the context of a criminal investigation, it is important that as much information as possible is obtained. Thus, identifying the people in the evidence images becomes important, especially if some of these people have already been registered in the database.

To identify people in the evidences, we used Python's library *face_recognition* to extract feature vectors from images present in the evidence. It extracts a feature vector from a face, which contains 128 different values for different aspects. After extracted, the values are saved in a database for future comparisons with other faces present in evidences or to make cross databases queries and identify possible known criminals in the evidence.

In order to speed up the queries, a clustering algorithm was implemented in the process, where groups are created through similar characteristics so that the search is made in a more agile way, thus not searching the entire database, but rather, searching in the groups that have a greater similarity. The algorithm used to do this process is the K-means. The clustering process has as input all the feature vectors extracted with Face Recognition. After the groups are created, each group also has also an array with 128 different features, and they are saved in the relational database, in their own table. The respective group that each image belongs to is stored next to the feature vectors and the image metadata. Thus, when new images are analyzed, the facial features are extracted, and the facial recognition API will make a comparison between the new image feature array and each group array, with a simple Euclidean distance. Then, after the distances are calculated, the API will first search the person in the group with the lowest result (the closest group), thus, speeding up the queries.

4.6. Training process

To train the Screenshots and Documents classes we used transfer learning, implemented with the help of the TensorFlow library. Transfer learning is a machine learning method where a model developed for a task is reused as the starting point for a model on a second task. The transfer learning method used a pre-trained InceptionV3 (Szegedy et al., 2016) model trained with the ImageNet dataset. In this case, the weights of the initial layers are locked and the neural network is retrained with the images collected from the Document and Screenshot classes, described earlier.

To train the classes that perform object detection, we used YOLO (Redmon and Farhadi, 1804) and InceptionV3, in conjunction with the transfer learning method. For this, we used the pre-trained weights provided by YOLO and we also created a configuration file where we specify how many classes we want to train in this procedure. Some changes were made to this configuration file to work specifically for our training. These changes were $batch = 64$ and $subdivisions = 16$. There is also another parameter, named *max batches*, but, there is a condition to perform the changes to *max batches*, which is to calculate the following: *number of training classes* $\times$ *2000*. This value cannot be less than the number of images used to train the neural network. Finally, the value of the parameter filters is equal to $(classes + 5) \times 3$ (see Fig. 1).

Today, several classification techniques use *Convolutional Neural Networks* (CNNs). During the project, it was noticed that, as our application classifies several types of object and images, the best approach it is to use techniques of object detection and also image classification. During the first tests, we realized that the utilization of classifiers such as Inception for some classes, such as weapons, where the image normally is in a context with other objects (as shown in Fig. 2), the classifiers was not able to understand which object was supposed to be classified. It is important to notice that our application must not only detect if the object is present on the image but also indicate where this object is.

The application has the following classes: Person, Cars, Motorcycle, Watches, Big Vehicles (ambulances, busses, pick-up trucks, trucks, and others), Knives, Firearms, Ammunition, and IDs, those classes being for object detection. And the Nudity, Screenshots, and Documents classes are for image classification.

Inside the classification process, there are several image classifiers as cited above. When a new image arrives, it goes through every classifier. However, some steps are added for images classified as Documents, ID cards, Screenshots, and People. The ones classified as Documents, ID cards, and screenshots go through an OCR algorithm to extract characters that can be processed separately, in order to extract more information.

Images labeled as Person are forwarded to three extra processes: nudity detection, age range classification, and facial recognition. These processes are executed with the idea of getting extra information for the analysts. Nudity detection tied to age range classification, for example, allow us to infer and detect child pornography cases. Facial recognition allows us to verify if pictures of people present in the evidence are already in a database.

The nudity detection API is responsible for detecting, in the images, traces of nudity, and, in the aspect of classification, it can consider a person wearing bathing suits as nudity also. The age range classification API places the person detected in the picture inside an age range. This allows the creation of age-based groups, for example, 0−13 years old as children, from 14 to 18 as teenagers and from 19 to 25 as young adults, and above that as adults. Uniting these two APIs can help forensic analysts to detect crimes of pedophilia in an easier manner. The micro-services of Nudity and Age Range are only activated when the image is classified as having a person in it.

5. Results

The evaluation process is separated into two parts. First, we will evaluate the image classification process, in which we describe the results of different image classifiers. Second, we will evaluate the usage of an Apache Spark cluster in the classification process, and access the difference between serial image processing and image

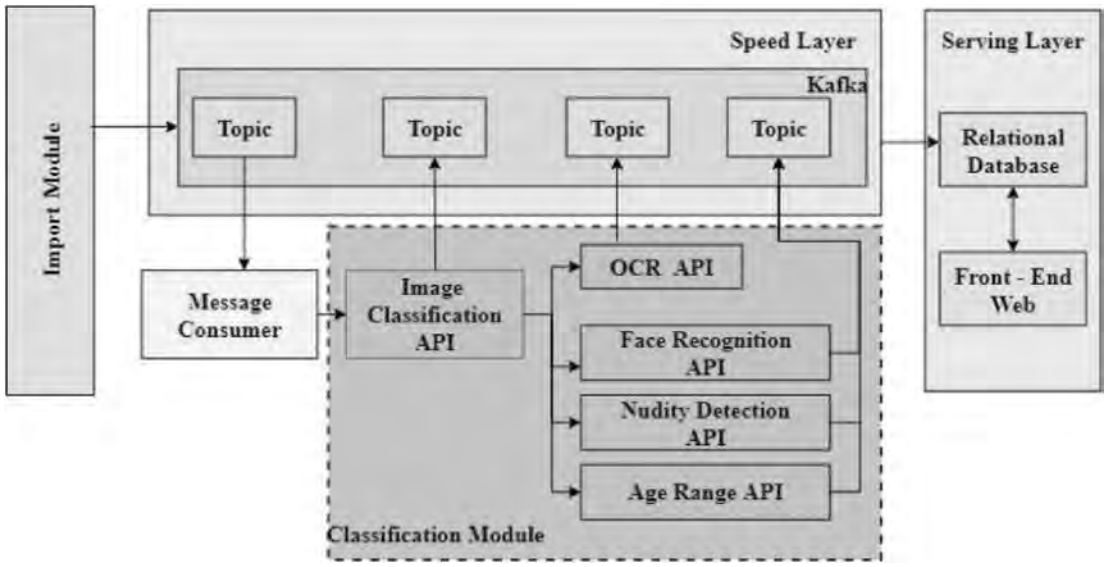

Fig. 1. Inside architecture.

Fig. 2. Example of image that could be sent to the classifier. Font: Internet.

processing in a Spark cluster.

5.1. Classifiers analysis

To assess the accuracy of the classifiers implemented on the platform, tests for Document, Chats Screenshots, and Nudity classifiers were performed using images from real crime evidences provided by the State Attorney's Office of Rio Grande do Norte, in Brazil. The Weapon and Ammunition classifiers were tested using images extracted from movie clips, with the intent of testing the classifiers with different object angles. For the Weapon classifiers were used the first 300 images of weapons that appeared in the video and 300 images, selected randomly, without weapons that also appeared in the video. The ID classifier was tested with images extracted from Google. Every image used for tests was not included in the dataset used for training.

Results are shown in Table 1 in terms of accuracy, precision, recall, and F1 score. We can see that despite the medium level of accuracy, the system achieved good precision for most cases. This is extremely important in the context of a criminal investigation, where false positives must be avoided at maximum. The recall values are mostly low, which indicates that the models should have to learn more to identify objects of interest.

Table 1
Results for class evaluation metrics.

Class	*Accuracy*	*Precision*	*Recall*	*F1 Score*
Fire guns	0.77	0.88	0.62	0.73
Ammo	0.72	0.80	0.55	0.65
ID	0.69	0.80	0.61	0.69
Screenshot	0.98	0.41	0.74	0.52
Document	0.98	0.72	0.79	0.74
Nudity	0.77	0.81	0.71	0.76

The accuracy metric was chosen because it is important to know how well the classifier is performing in general terms. The metrics of precision and recall are important to determine whether the classified images are correct or false positives. The F1 Score shows the harmonization of the precision and recall metrics.

To build the ammunition classification model, we used samples of ammunition from a wide range of firearms. So, for each type of gun, the shape and size of its ammunition is different. For example, a pistol ammo is very distinct of a rifle ammo (Fig. 3). One simple solution for this is to train distinct classifiers for different sizes and/or shapes of gun ammunition (see Fig. 4).

For illustration purposes, in Figs. 5–7 it is possible to see some examples of the images labeled by the fire gun classifier. In those pictures, it is possible to see the classifier detecting small weapons and medium weapons, such as pistols and submachine guns (SMGs), in different angles and with a high level of difficult to be detected.

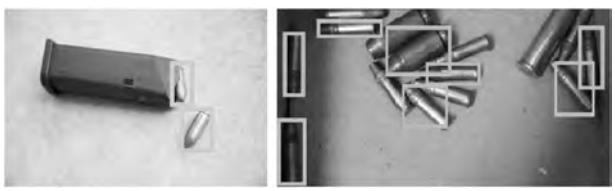

Fig. 3. a) Ammo detected in a cartridge clip. b) Ammo detection of different sizes of ammo.

Fig. 4. Ammo detected in hand and in a cartridge clip.

Fig. 5. Firearms detected in while analysing scenes from the movie Matrix where the character has submachine guns (SMGs) in his hands.

Fig. 6. More firearms detected in scenes from the movie Matrix, this time, a pistol in the characters hand.

Fig. 7. Firearm detected in a picture of a pistol being hold by someone.

Table 2
Face search time measurement.

Linear	2 Groups	3 Groups
1.5088s	0.5079s	0.3201s
5 Groups	**8 Groups**	**10 Groups**
0.2464s	0.2563s	0.1302s
20 Groups	**50 Groups**	**100 Groups**
0.1656s	0.1189s	0.1310s

5.2. Clustering

With a large number of images in the database, the search for faces can be very slow. In order to improve the search time for a face in the database, we used K-means to do the clustering. K-means is a very popular clustering method for partitioning a group of data into K groups (MacQueenet al., 1967). To measure the efficiency of the clustered search we compared it to the linear search time we ran 500 rounds of tests. The tests were performed by searching a database with 7602 stored faces and the table shows the results.

As we can see, there was a big difference when using clustering to search for faces in the database. We noticed that the greater the number of groups used, the greater is the performance gain in the query, as shown in Table 2, when using 10 clusters, we already have a speed gain of almost three times.

5.3. Processing time (serial x Spark)

Earlier in 2020, we published an article that had YOLO and Spark already integrated, but, with no gain in raw processing time, except for conversion to base64 step. Now, we were able to improve processing time with the tuning of the Spark job.

There are several ways of tuning a Spark job, one of which is resource allocation. This type of tuning means using the right amount of *CPU* and *memory* (the principal resources that Spark cares about) for each job in the cluster. Using YARN (Yet Another Resource Negotiator) with Spark, we defined that the number of Spark executors cores would be 1, and the number of instances would be 2. Also, the memory for each executor and the Spark driver would be 2 GB. We also defined that the amount of memory used in each node would be 7 GB, so the machines would have enough memory to run other processes.

To assess the feasibility of the proposed platform, tests were conducted using real crime evidence data provided by the State Attorney's Office of Rio Grande do Norte, in Brazil.

In the classification context, four rounds of tests were conducted to evaluate the Image Classification API processing serially and using Apache Spark. For each round, a batch of 5000 images was used as input and, to avoid outliers, we processed this batch of images five times, and computed the average processing time. The test results can be seen in Table 3 for both serial and Spark processing. It is important to note that the reported time values are relative to all the steps of the architecture mentioned in Section 3. The serial tests were conducted in a machine running Windows 10 Pro operating system, with 4 cores and 4 threads running at 3.50 GHz and 16 GB of RAM. The Spark tests were conducted in a cluster composed of three machines. Each machine has a Red Hat Enterprise Linux (RHEL) 7.6 operating system with 2 cores and 2 threads and 8 GB of RAM.

Analysing the results, it is easy to see that the processing with the tuned Spark job is almost three times faster than the serial processing. It is also important to note that, even though our Spark

Table 3
Comparison between serial processing and processing using the Apache Spark - Average images per second.

Entry	Serial	Spark
1	0.65	1.82
2	0.65	1.81
3	0.64	1.82
4	0.67	1.81

tests were running on a cluster, which had a total computational power superior to the machine that ran the serial tests, Spark was not able to create multiple jobs in different nodes of the cluster. That means that the threefold gain was achieved using a single node of the cluster. In the future, we aim to improve this processing time even more by using every node of the Spark cluster.

Also, it is possible to see some results of the image classification process and object detection. These results are provided at the end of the pipeline process, and the Figure illustrates the detection of a firearm and two documents. The two images on the right were altered to preserve the privacy of the information.

The overall results show that the proposed approach provides a much faster analysis of images. When the number of images present in an evidence increases to the scale of tens of thousands, the difference in time consumption can make a difference for any investigation. Considering all the processing, the proposed architecture delivers the same result 3 times faster than the serialized approach.

Even though Spark was configured to process the images using all the computational power of the cluster, a limitation in one of its libraries made the consumption from the Kafka topic only able in one of the machines of the cluster. Thus, the classification process was faster even with a computational power smaller than the one used in the serial approach. In theory, as soon as the Spark library limitation is overcome, the architecture will be able to achieve a processing time of approximately six times faster than the serial approach. And with the whole computational power of the cluster used in this work, it may be possible to achieve a nine-fold gain in processing time. Of course, the usage of a more powerful server would increase this advantage while also making it possible to process a larger amount of cases simultaneously.

6. Conclusion

This paper presented the INSIDE's image classification module built to extract evidences at the Forensic Laboratory of the State Attorney's Office of Rio Grande do Norte (MPRN) together with the Federal University of Rio Grande do Norte (UFRN) created in the context of INSIDE (Integration, aNalySis, vIsualization of Data for invEstigation) big data platform that allows crime analysis capable of powering scalable, real-time and data-driven applications.

As a result of this paper, we can conclude that the proposed architecture is a viable option as a system to help the analysis of digital evidences. The architecture ability of reading, processing, and classifying large amounts of images, with agility and precision, allowing the forensic analyst to have an easier and faster process in the analysis of these digital evidences.

The usage of a micro-services-based architecture allows the application to have bigger flexibility regarding the services it uses, seeing that because of the architectural model, new modules can be added when necessary, without interruption of the pipeline. Besides that, old modules services can also be updated in a timely manner, without interfering with the other services.

Also, we build models to identify five specific types of objects: fire guns, gun ammunition, Brazilian id cards, text documents, and smartphone screen pictures. Those objects were chosen considering the necessity of forensic analysts and the availability of images in the database. Results showed that we were able to achieve satisfactory results in terms of precision for most classes.

As future work, we plan to improve the results obtained with the current classifiers and implement new ones. We believe that with more image diversity and more images, we can make the classifiers to be more precise and accurate. Some classes have a great granularity, and in this case, a possibility is a division in smaller classes, as it was done in the fire guns class, which was separated into small, medium, and large-sized weapons and with the ammunition class, which has a variety of ammo models. New classes can be added to the project accordingly with the needs of the MPRN. Likewise, a module for video analysis and processing will be implemented in the future. And, with the utilization of micro-services, the addition of new features is facilitated, allowing the possibility of upgrades without major problems.

Acknowledgment

This work is funded and supported by Ministério Público do Rio Grande do Norte in the context of the INSIDE Project[1] and by the SmartMetropolis Project.[2]

References

Bedapudi, P.. Nude Net: An ensemble of neural nets for nudity detection and censoring. https://praneethbedapudi.medium.com/nudenet-an-ensemble-of-neural-nets-for-nudity-detection-and-censoring-d9f3da721e3. (Accessed 6 February 2021).

Brown, R., Pham, B., de Vel, O., 2005. Design of a digital forensics image mining system. In: International Conference on Knowledge-Based and Intelligent Information and Engineering Systems. Springer, pp. 395–404.

Chambers, B., Zaharia, M., 2018. Spark: the Definitive Guide: Big Data Processing Made Simple. O'Reilly Media, Inc.

de Carvalho, T.J., Pedrini, H., de Rezende Rocha, A., 2015. Visual computing and machine learning techniques for digital forensics. Revista de Informática Teórica e Aplicada 22 (1), 128–153.

do Valle, J.M., Souza, G., Fidelis, S., Araújo, A., Cacho, N., Silva, I., Budke, J., Sales, H., Filgueira, M.W., Lopes, F., et al., 2020. Big data platform for analysing crime evidences. In: 2020 IEEE Sixth International Conference on Big Data Computing Service and Applications (BigDataService). IEEE, pp. 113–119.

Felzenszwalb, P.F., Girshick, R.B., McAllester, D., Ramanan, D., 2009. Object detection with discriminatively trained part-based models. IEEE Trans. Pattern Anal. Mach. Intell. 32 (9), 1627–1645.

Girshick, R., 2015. Fast R-CNN. In: The IEEE International Conference on Computer Vision. ICCV).

Grega, M., Matiolanski, A., Guzik, P., Leszczuk, M., 2015. Automated detection of firearms and knives in a CCTV image. Sensors 16, 47. https://doi.org/10.3390/s16010047.

Han, H., Jain, A.K., 2013. Tattoo based identification: sketch to image matching. In: 2013 International Conference on Biometrics (ICB). IEEE, pp. 1–8.

Hoelz, B.W., Ralha, C.G., Geeverghese, R., 2009. Artificial intelligence applied to computer forensics. In: Proceedings of the 2009 ACM Symposium on Applied Computing, pp. 883–888.

P. Korus, N. Memon, Neural imaging pipelines-the scourge or hope of forensics?, arXiv preprint arXiv:1902.10707.

Lee, J., Jin, R., Jain, A., Tong, W., 2012. Image retrieval in forensics: tattoo image database application. IEEE MultiMedia 19 (1), 40–49.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., Berg, A.C., 2016. Ssd: single shot multibox detector. In: European Conference on Computer Vision. Springer, pp. 21–37.

Luu, H., 2018. Beginning Apache Spark 2: with Resilient Distributed Datasets, Spark SQL, Structured Streaming and Spark Machine Learning Library. Apress.

MacQueen, J., et al., 1967. Some methods for classification and analysis of multivariate observations. In: Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, vol. 1, pp. 281–297. Oakland, CA, USA.

Mahadeokar, J., Pesavento, G.. Open sourcing a deep learning solution for detecting nsfw images. https://yahooeng.tumblr.com/post/151148689421/open-sourcing-a-deep-learning-solution-for. (Accessed 6 February 2021).

Marz, N., Warren, J., 2015. Big Data: Principles and Best Practices of Scalable Real-Time Data Systems. Manning Publications Co., New York.

[1] http://inside.imd.ufrn.br.

[2] https://smartmetropolis.imd.ufrn.br/.

Moltisanti, M., Paratore, A., Battiato, S., Saravo, L., 2015. Image manipulation on facebook for forensics evidence. In: International Conference on Image Analysis and Processing. Springer, pp. 506–517.
Narkhede, N., Shapira, G., Palino, T., 2017. Kafka: the Definitive Guide: Real-Time Data and Stream Processing at Scale. " O'Reilly Media, Inc.".
J. Redmon, A. Farhadi, Yolov3: an Incremental Improvement, arXiv preprint arXiv: 1804.02767.
Redmon, J., Divvala, S., Girshick, R., Farhadi, A., 2016. You only look once: unified, real-time object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 779–788.
Ribeiro, A.M., Francisco de Paula, S., 2019. Rede inception v3 voltada a identificação do glaucoma: comparação entre métodos de otimização. In: Anais da VII Escola Regional de Computação Aplicada à Saúde. SBC, pp. 1–6.
L. F. Rodrigues, et al., Comparação entre redes neurais convolucionais e técnicas de pré-processamento para classificar células hep-2 em imagens de imunofluorescência.
Silva, A.L., Cintra, M.E., 2015. Reconhecimento de padrões faciais: um estudo. In: Encontro Nacional de Inteligência Artificial e Computacional, 2015. Proceedings ENIAC, pp. 224–231.
Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z., 2016. Rethinking the inception architecture for computer vision. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2818–2826.
Tahir, S., Iqbal, W., 2015. Big data—an evolving concern for forensic investigators. In: 2015 First International Conference on Anti-cybercrime (ICACC). IEEE, pp. 1–6.
The Apache software foundation - Apache Spark documentation. https://spark.apache.org/docs/latest/ accessed on: 09 set 2019.
Thein, K., 2014. Apache kafka: next generation distributed messaging system. International Journal of Scientific Engineering and Technology Research 3 (47), 9478–9483.
Uckelmann, D., Harrison, M., Michahelles, F., 2011. An architectural approach towards the future internet of things. In: Architecting the Internet of Things. Springer, pp. 1–24.
Van Baar, R., Van Beek, H., Van Eijk, E., 2014. Digital forensics as a service: A game changer. Digit. Invest. 11, S54–S62.
Van Beek, H., van Eijk, E., van Baar, R., Ugen, M., Bodde, J., Siemelink, A., 2015. Digital forensics as a service: game on. Digit. Invest. 15, 20–38.