Object-Based Image Comparison Algorithm Development for Data Storage Management Systems

Kirill Smelyakov [1], Oleksandr Prokopenko [1] and Anastasiya Chupryna [1]

[1] *Kharkiv National University of Radio Electronics, 14 Nauky Ave., Kharkiv, 61166, Ukraine*

Abstract

Continuous development of big image storage management systems requires development and introduction of efficient searching and image comparison algorithms. In the scope of this article we analyze modern object detection models and image search algorithms based on their metadata and processed images object localization area comparison. The main advantages and disadvantages of such algorithms are discovered. Article offers objects data extraction algorithm from image using CNN, storage model of this information in image file service fields as metadata, cascade object-oriented image comparison and fast search algorithm using modern solutions in the field of parallel computing. A series of experiments is being set up to apply the proposed algorithms for comparing and searching images in large data storages. The results of experiments, analysis of the effectiveness of the application, conclusions and recommendations for the practical application of the proposed algorithms are given.

Keywords

Image, Metadata, Image Storage, Machine Learning, Image Detecting and Classification, Image Comparison and Search Algorithms, Convolutional Neural Network

1. Introduction

Currently, one can see an exponential growth trend in the number and volume of personal, corporate and commercial image storage. Many such repositories, such as photobanks, contain millions of images. The efficiency of the management system for such storages directly depends on the efficiency of comparing and searching for images during search queries. Every day, users are more and more interested in the content of the image, and not its formal parameters. Users try to include this information in their search queries. In such a case, the search algorithms must efficiently search for images given the specified information about the content of the images. And such image processing should be carried out efficiently (according to the criteria of laboriousness and accuracy) in the presence of millions of images in storage. In this regard, the main problem is related to ensuring the efficiency of comparing and searching images by their content in the presence of a large number of objects of the same type in the image, the localization areas of which repeatedly overlap.

Therefore, the purpose of the work is to ensure the efficiency of comparison and search for images by their content (according to the criteria of labor intensity and accuracy) in big data storages for such conditions.

Objectives of the work: to develop methods for extracting information about objects in an image using CNN, a model for storing this information in the service fields of image files, methods for comparing and quickly searching for images using modern solutions in the field of parallel computing, especially for the conditions of the presence of a large number of objects of the same type in the image , whose localization regions repeatedly overlap.

COLINS-2022: 6th International Conference on Computational Linguistics and Intelligent Systems, May 12–13, 2022, Gliwice, Poland
EMAIL: kyrylo.smelyakov@nure.ua (K. Smelyakov); oleksandr.prokopenko1@nure.ua (O. Prokopenko); anastasiya.chupryna@nure.ua (A. Chupryna)
ORCID: 0000-0001-9938-5489 (K. Smelyakov); 0000-0003-0489-6820 (O. Prokopenko); 0000-0003-0394-9900 (A. Chupryna)

2. Related Works

Modern trends in image search by content [1] require the development of appropriate models for their description, as well as algorithms for extracting information, comparison and search based on content [2-4].

Modern applications often operate with big amounts of data. Such amounts of information need to be processed in a manner that differs from classical applications. Storage structure must be adapted to handle huge amount of incoming data, example of such architecture is data lake, which stores all the data in raw format, so that unstructured files, half-structured files like JSON or XML and SQL tables are stored in one repository representing all the information used by application [5].

Data lakes need appropriate tools to manage stored files. While changing a file is forbidden, because precious user's data may be damaged, one can utilize file's metadata to store some additional information about each single file which can be used later on for search and comparison of two files [6]. In this regard, the works [7, 8] show how the use of clarifying metadata in solving problems of medical diagnostics can improve the quality of classification; especially when such metadata is used in the course of applying deep learning algorithms. The use of informative features and additional data in the subject area can contribute to a significant increase in classification accuracy through the use of deep learning methods [9, 10].

In general, modern studies [10, 11] show that the effective management of image storages is associated with the solution of several key tasks. First, with pre-processing, which is used to improve the quality of images, clean them, filter extreme observations, normalize gray scales, and present images in the required format [12, 13]. Second, it is associated with the preliminary detection of regions of interest, as described in [14, 15] and the extraction of information about the content of the image, as a rule, based on the use of CNN [9, 10, 14]. Such CNNs are able to detect areas of object localization, save their coordinates and class [16-18]. The advantage of most modern CNNs is the high speed and accuracy of detection in automatic mode. Their main disadvantage is that the CNN can be trained to work with images of a certain list of classes. Images of other classes cannot be processed by CNN.

Afterwards, during the lifetime of processed information inside an application it can be used to easily manage files, search for certain kinds of information stored, filter out chunks of data that is required by the user for his specific scenario etc. [19, 20].

Image data is harder to work with, since it is not a trivial task for a computer to find if an image is already stored in a database or to find all images that have a person on them. Mostly, these tasks are accomplished by machine learning models that process an image to extract specific information from an image like depicted objects and their locations [21-23].

At the same time, as practice shows, currently widely used models and algorithms are effective for comparing a small number of objects in the image, the localization areas of which do not intersect with each other. In most cases, these algorithms are based on the consideration of box coordinates obtained after applying a convolutional neural network, and also, in some cases, on the use of local feature detectors (FAST, ORB, etc.) in order to compare the corresponding descriptors of image feature points [24 , 25].

In the case when an image contains a large number of objects of the same type with localization areas that repeatedly overlap with each other, it is often impossible to effectively use classical models and algorithms for comparing images in the data storage during a search query, image filtering. To effectively satisfy search queries in such conditions, it is necessary to develop a new model and method that will ensure efficient processing of images and areas of object localization on them, including introduction of parallel data processing schemes [26, 27].

Considering modern developments and achievements, for this it is advisable to consider a cascade model for comparing and searching for images:

- first (once before searching) for each image in the storage, it is advisable to extract information about its content and save it as metadata;
- when searching at the first stage, it is advisable to perform a quick search / filtering of images by metadata; the thing is that with a large amount of metadata used for a search query, the probability of a match decreases non-linearly. In such a situation, one can quickly and without

resorting to more complex algorithms make an adequate sample of images. Otherwise, such a search can be considered as preliminary to reduce the scope of the search;

- when searching at the second stage, one should apply the analysis of the degree of overlap of areas of image localization, and apply the appropriate algorithm for comparing the degree of overlap of these areas.

Such a cascade method will allow you to optimally organize the search for images in large data storages. In addition, the development of search algorithms will make it possible to efficiently compare images in related services, for example, for the purpose of real-time traffic analysis in IoT [28] and many other applications [29].

3. Methods and Materials

This section describes selected machine learning models, dataset, data storage model and reasons why they were chosen in the scope of this article to conduct experiments on offered image search and image comparison algorithms.

3.1. Data Description

As was mentioned in the introduction in the scope of this article we will use image data to show possibility and expediency of smart metadata utilization for big data management in data lake architectures. First and foremost, we had to decide on image formats used for research. As it turned out most of the popular image formats support Exif standard, thus use of Exif metadata tags is making the solution presented in this article compatible for a vast number of file formats like JPEG, TIFF, etc.

Generally there are two different types of tag specified by Exif which can be used to write any arbitrary information.

First one is a tag under Image IFD called “ImageDescription”, its own description states that stored data is: “A character string giving the title of the image. It may be a comment such as "1988 company picnic" or the like. Two-bytes character codes cannot be used.”

Limitations for usage of this tag are obvious, there may be a situation where we need to store some data in form of string, describing for example name of depicted object’s class and in case if it will be used with non-English characters it wouldn’t be properly saved, so it’s better to use second tag.

Another tag used for saving arbitrary information is located under Photo IFD and called “UserComment”. Its description says that this is: “A tag for Exif users to write keywords or comments on the image besides those in <ImageDescription>, and without the character code limitations of the <ImageDescription> tag.”

For experiments we will use images in JPEG format, because this format is most common and a significant part of existing datasets consist of images with this file format. It is worth mentioning that metadata has restrictions in stored data size. Exif metadata are restricted in size to 64 kB in JPEG images because according to the specification this information must be contained within a single JPEG APP1 segment.

Dataset for a project should contain some general images which depict photos that can be found in a common person’s phone, including photos of people, pets, animals, cars, laptops, phones, etc. Specialized datasets do not fit to achieve our goal, because we want to check the ability to run a similar image search for real-life photos that can contain some various depicted objects against a massive amount of other photos.

COCO (Common Objects in COntext) dataset is a large-scale object detection, segmentation, and captioning dataset. It is sponsored by CVD foundation, Microsoft and Facebook. Now it has more than 200 000 labeled images. This dataset is constantly growing and improving its quantity and quality. It has 80 classes of objects depicted in photos, which are pretty generic like person, dog, cat, it fits the goal of our research perfectly since if we want to compare different images by depicted objects, we would rather know that image contains a dog, than a specific dog’s breed. For sake of simplicity, we will take a subset of 5000 images that are used for training purposes in machine learning to make our experiments.

3.2. Machine Learning Model

In order to make some experiments we need to find a machine learning model for object detection on images. YOLO, an acronym for 'You only look once', is an object detection algorithm that divides images into a grid system. Each cell in the grid is responsible for detecting objects within itself. YOLO is one of the most famous object detection algorithms due to its speed and accuracy. The YOLOv5 algorithm can be used for our goals, since it is using the Pytorch framework which allows it to run on a vast majority of modern operating systems using a graphics card for the hardest part of calculations. Also YOLOv5 contains a few types of pretrained models of different size and accuracy which allows us to adapt to certain systems and hardware limitations.

YOLOv5 has the One-Stage Detector architecture, an approach that predicts the coordinates of a significant number of bounding boxes with the results of the analysis and the probability of finding the object, and adjusting their positions afterwards. In general, such an architecture can be represented as follows (Figure 1).

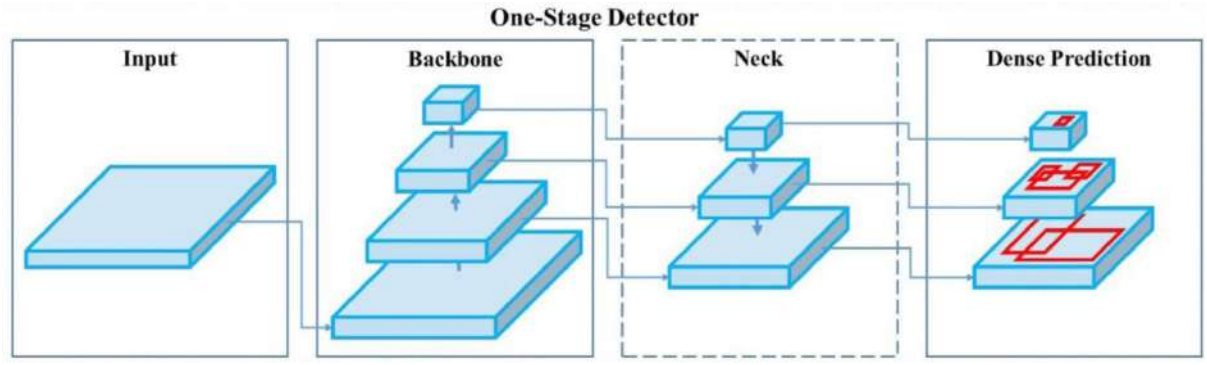

Figure 1: One-Stage Detector's general architecture [30]

The network scales the original image into multiple feature maps using a pass-through connection and other architectural tricks. The resulting feature maps are reduced to a single resolution using upsampling and concatenation. The classes and bounding boxes for the features are then predicted, then the most likely bounding box for each feature is selected using Non-Maximum Suppression.

Each bounding box information is represented by five values:

- Class number;
- X center coordinate;
- Y center coordinate;
- Width;
- Height.

Coordinates, width and height are relative values between 0 and 1, which allows the image to scale without losing object position. YOLOv5 has a confidence threshold, which can improve quality of image comparison, since we can filter out irrelevant objects. Since bounding boxes are rectangles, it is pretty easy to calculate the intersection area of different objects which can be done fast enough.

This object detection model is one of the fastest and most accurate, for example it is 2-2.5 times faster than other popular models like Faster R-CNN, SSD, RetinaNet. This speed is crucial for our use cases when detection should happen as fast as possible.

3.3. Methods

As it was mentioned in the introduction part we will be solving image comparison problems using image metadata and three different comparison algorithms. First of all, it is worth mentioning that once images are labeled and marked with depicted objects data, it becomes relatively easy to filter big amounts of images, leaving only a subset which contains dog and person on them.

All image comparison algorithms are using a cascade approach and giving us similarity assessment as a coefficient between 0 and 1 including, where 0 is a comparison of images that depict sets of objects which do not intersect and 1 is the result of comparing the image with itself.

Let's start with the simplest trivial algorithm which we will use to compare two photos. We have photo A that has a set of detected objects A = { O11, O12, … O1N } and photo B that has a set of detected objects B = { O21, O22, … O2M }.

First step is to group objects by their class label and count the amount of objects of each class that photos contain. Generally speaking, we will have two dictionaries where keys are class marks and values are the number of objects on image. We want to make the result of comparing A to B the same as comparing B to A. To achieve it, we will take the list of distinct class labels found on two images. After this we can calculate the similarity between two images by each class label. Calculation is as simple as it can be, one image will contain a lesser or equal number of objects with some class label than the other, so we will divide this value by the number of objects with this class label on the other image. We assume that all class labels have the same weight, so to calculate the final similarity coefficient we need to sum all similarities by label and divide this sum by the number of distinct class labels.

This algorithm will also be used as a preliminary filter to two other algorithms, because it does not contain any heavy calculations and can remove all images that do not have any intersection with our target image in object terms, so it does not make any sense to compare object locations on these images. When we are talking about hundreds of thousands of images in a database it will shorten the set of images that should be compared to a few percent of the initial amount of images depending on the threshold used for filtering.

Second algorithm focuses on comparison of two images depending on objects locations. Same as with the first algorithm we will compare objects of the same type, but now we will compare not only their numbers on both images, but the intersection area of two objects. To calculate the coefficient between two objects we need to calculate their intersection area which is a trivial task given that bounding boxes are rectangles and the resulting intersection is also a rectangle.

We can calculate each object's area because their heights and widths are saved in the image's metadata. Then, we can get a coefficient with a given formula: $S_i/(S_1 + S_2 - S_i)$, where S_1 - First object's area; S_2 - Second object's area; S_i - Intersection area (Figure 2).

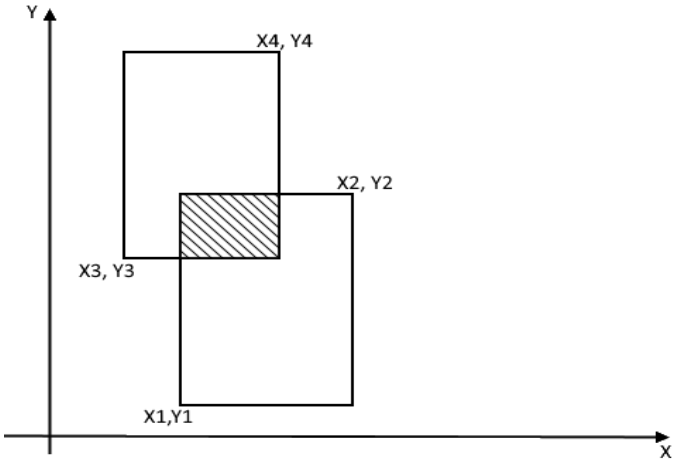

Figure 2: Intersection area of two rectangles

Once, we calculate the coefficient for each pair of objects of some type. Then we must pick pairs of objects that are maximizing the sum of coefficients. This procedure is repeated for each class label and the general sum of coefficients is divided by the number of objects on image. This algorithm allows to get more precise results than first, since it takes into account objects location in addition to their classes and numbers on image.

But, this algorithm contains some significant flaws which can result in not too precise results and slow execution. This algorithm works at its best when there are not a lot of objects of the same type

on image. In case if we have two group photos, the exact intersection areas of pairs of people in two photos may not be big, while the common area people taking on images is likewise. Second concern is an execution time concern since if we have tens of objects of the same types on both images, then search for a maximum sum of coefficients can take significant time, since this is an assignment problem solution which has time complexity $O(n^3)$.

Third algorithm also takes into account objects' locations and solves flaws that the bounding box comparison algorithm has. Instead of comparing specific objects, we will compare areas covered by objects of a certain type on both images. To calculate intersecting areas by objects on two images we will use a matrix with the size of an image, where the value of each pixel in the matrix is equal to a number of images that contain an object of this type in the pixel's location.

In order to calculate similarity coefficient by class type using this matrix we need to calculate the amount of 2's in the matrix, which represents intersecting areas and amount of non-zero values which will represent the common area taken by objects on both images, then we will get a coefficient dividing intersecting area by common area.

Then we will have our coefficient of similarity by class type, example could be seen on Figure 3. After calculating coefficients by all object types that were detected on images we should aggregate these values in a general similarity coefficient between two images. For the current article we are treating different object types as equal, so to get a similarity coefficient we are summing up coefficients by all object types and dividing this sum by a number of distinct object types on these images.

Basically we are utilizing an idea of a scanner that goes through the matrix value by value and finds out if location belongs to both images, only one of the images or none of the compared images. So we remember locations of certain type objects on one image at first and then, going through objects on the second image and calculating final values of areas and coefficients.

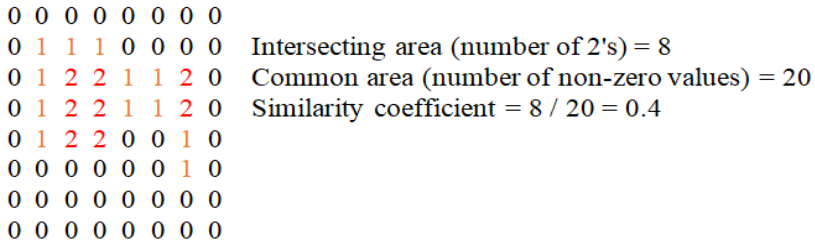

Figure 3: Matrix similarity coefficient calculation example

Main holdup for this algorithm is correct implementation that will allow us to calculate coefficients in acceptable time.

First problem is comparison of images with high resolution, in case the matrix contains a few thousands or even millions of values, it is taking a lot of time to scan through it.

We can fix this issue with scaling. Images contain information about detected objects in their metadata as relative values. So we can compress the matrix to a certain small size, which will be fast enough to calculate. Compression comes with a loss of precision, but as practice shows us, the difference in a calculated result on compressed images and full-sized images is only a few percent. Further in an experimental part of this article, we will show that change in results is neglectable and precision is high enough even when compression coefficient is as low as 0.2 of an original size.

This algorithm's execution time can be improved even further. Matrix comparison algorithms can be paralleled with a great execution time improvement.

First of all, concurrent execution can be implemented on multiple levels of calculations. Highest level is an object type level, comparison of the most part of images will detect multiple types of objects and all similarity coefficients by object type can be done independently, which means that calculation time in perfect conditions shortens from sum of times to calculate similarity by all of object types to time that takes one longest calculation of similarity coefficient by object type.

Next possible improvement is concurrent scanning of the matrix. Since the matrix has rectangular shape, we can split it into smaller segments and delegate intersecting and common area calculations to separate processes.

One of the most interesting ideas is to execute these calculations via graphics card, minimizing execution time. The only concern is the time spent on data exchange between graphics card memory and RAM, which can lose more time than concurrent execution wins. In the scope of this article improvement with graphics card utilization won't be implemented while concurrent calculation of coefficients by different types will be actively used, since it allows using this algorithm and having execution time that is acceptable for use in applications that operate with big data amounts.

3.4. Technologies

As it was mentioned, the main used technology is object detection model YOLOv5 that is the same as the YOLOv4 version, but moved to Pytorch framework - an open source machine learning framework that accelerates the path from research prototyping to production deployment. YOLOv5 is an open-source project so we will use it as a backbone [30].

To run object detection, metadata extraction and image similarity comparison we will modify the Python script from the YOLOv5 repository called detect.py to meet our needs and use it with a few purposes: write object information to image metadata and run image similarity comparison. Pytorch will run object detection on a graphics card for the best performance.

To work with EXIF metadata we will use ExifTool that is a free and open-source software program for reading, writing, and manipulating image, audio, video, and PDF metadata. It is platform independent, available as both a Perl library (Image::ExifTool) and command-line application. ExifTool is commonly incorporated into different types of digital workflows and supports many types of metadata including Exif, IPTC, XMP, JFIF, GeoTIFF, ICC Profile, Photoshop IRB, FlashPix, AFCP and ID3, as well as the manufacturer-specific metadata formats of many digital cameras.

Script modifications include obligatory save of image metadata after object detection and run of image similarity comparison depending on command-line argument.

4. Experiment

Experimental section of this article will be used to reveal main advantages and disadvantages of developed algorithm and assess its effectiveness and usage in real-world scenarios.

4.1. Initial Data

We will run performance experiments and compare algorithms on a part of the COCO 2017 dataset. This part includes five thousand images and depicts 80 types of objects [31, 32]. We will select five out of five thousand images to run our tests and compare algorithms results by execution time and selected results (Figure 4). These images depict different types of images: bear, table with food, crowd, family and tennis player.

Figure 4: Images to compare [31, 32]

4.2. Experiments Plan

First of all we will run image comparison for all of the five selected images with simple comparison, boxes comparison and matrix comparison. Then we will compare their: Execution time; Best result; Similarity; Top-5 picks coefficients spread. We will compare received results to the values received with comparing histograms of two images using correlation metric provided by OpenCV library. This comparison should give us an idea on how efficient and accurate our results in comparison to classic widely-used algorithm.

Tests will be run with confidence threshold at level 0.5 to filter out all objects that YOLOv5 is not confident about, since a lot of untrustworthy objects can decrease similarity coefficients drastically while images look alike.

Then we will focus on matrix image comparison algorithms, trying to figure out the optimal compression ratio for calculations which will leave us with precise results that are given fast. Preliminary we will run object detection on all tested images to have a visual hint on found objects, they're shown on Figure 5.

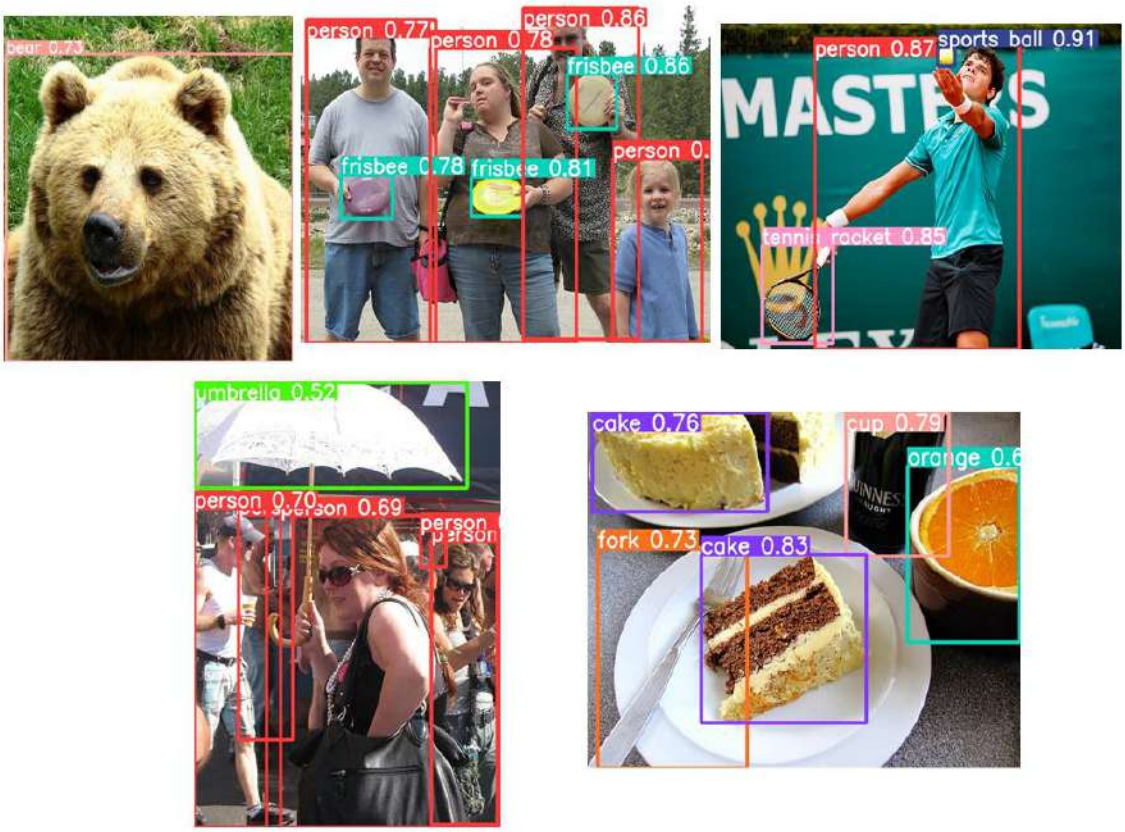

Figure 5: Detected objects on images

4.3. Hardware and software of the testing system

Working station to execute experiments is a laptop HP Pavilion Gaming Laptop 17-cd0xxx. This laptop features Intel(R) Core(TM) i5-9300H CPU @ 2.40GHz as a central processing unit, has 24 gigabytes of RAM, 128 GB SSD and 1 TB HDD.

Graphics card installed on the working station is Nvidia's GeForce GTX 1650 built on Turing architecture with 4GB of GDDR5. Graphics card contains 896 CUDA Cores that are used for machine learning. CUDA primitives power data science on GPUs.

NVIDIA provides a suite of machine learning and analytics software libraries to accelerate end-to-end data science pipelines entirely on GPUs. This work is enabled by over 15 years of CUDA development. GPU-accelerated libraries abstract the strengths of low-level CUDA primitives. Numerous libraries like linear algebra, advanced math, and parallelization algorithms lay the foundation for an ecosystem of compute-intensive applications.

Operating system is Windows 10 Professional, which allows us to run our experiments with all latest updates and newest drivers for the hardware. As it was mentioned earlier, to run our scripts we use the Python 3.7 version with Pytorch framework.

5. Results

Tables 1, 2 and 3 show general results for average execution time, selected images and highest found similarity coefficient for tested images for developed algorithms. Table 4 shows received results for histogram comparison by correlation metric. Execution time in table does not include time to read and parse metadata for three offered algorithms, also box and matrix algorithms show execution time without preliminary filtering. Execution time for histogram comparison takes into account time to build, normalize and compare histograms for 5000 images in dataset. Time to read image data from storage on machine is not accounted as well as time to read metadata for first algorithms, because this time highly depends on hardware and can be improved a lot by using caching techniques and we want to compare algorithms execution time.

Table 1
Simple algorithm execution results

	Bear (285.jpg)	Family with frisbees (100238.jpg)	Tennis player (170474.jpg)	Crowd (250137.jpg)	Table with food (496954.jpg)
Execution time (ms)	718 ms	843 ms	812 ms	796 ms	796 ms
Highest similarity coefficient	1	0.75	1	0.916667	0.75
Top-5 picks coefficients spread	1.0 - 1.0	0.75 - 0.70833	1.0 - 1.0	0.916666 - 0.9	0.75 - 0.5

Table 2
Box comparison algorithm execution results

	Bear (285.jpg)	Family with frisbees (100238.jpg)	Tennis player (170474.jpg)	Crowd (250137.jpg)	Table with food (496954.jpg)
Execution time (ms)	0 ms	135 ms	109 ms	126 ms	31 ms
Highest similarity coefficient	0.907722	0.293879	0.30171	0.318123	0.145482
Top-5 picks coefficients spread	0.907 - 0.45	0.293 - 0.264	0.301 - 0.268	0.318 - 0.243	0.145 - 0.123

Table 3
Matrix comparison algorithm execution results

	Bear (285.jpg)	Family with frisbees (100238.jpg)	Tennis player (170474.jpg)	Crowd (250137.jpg)	Table with food (496954.jpg)
Execution time (ms)	141 ms	303 ms	309 ms	366 ms	455 ms
Highest similarity coefficient	0.910818	0.42502	0.262802	0.588056	0.159144
Top-5 picks coefficients spread	0.91 - 0.491	0.425 - 0.331	0.262 - 0.23	0.588 - 0.426	0.159 - 0.089

Table 4
Histogram comparison algorithm execution results

	Bear (285.jpg)	Family with frisbees (100238.jpg)	Tennis player (170474.jpg)	Crowd (250137.jpg)	Table with food (496954.jpg)
Execution time (ms)	1786 ms	1754 ms	1609 ms	1617 ms	1724 ms
Highest correlation coefficient	0.834534	0.962971	0.937198	0.980637	0.799085
Top-5 picks coefficients spread	0.834 - 0.772	0.963 - 0.959	0.937 - 0.328	0.980 - 0.971	0.799 - 0.773

As expected, a simple algorithm produces a lot of similar results for images that depict exact or very close configuration of objects. Bear image comparison produced a dozen images with one detected bear on them. One of the top results was not too close because two bears were falsely detected as one bear. For a picture of a family with frisbees we can see an image of the frisbee team as the best match and coefficient that is less than one because the number of detected people and frisbees are different. Another tennis player's image is selected as the best match for a depicted tennis player with a tennis racket and ball. Picture of the crowd with woman with umbrella on front has similar picture as best match, the only difference is locations and whole picture, because here we can see only legs, while in a tested image we can see people full-height, this image probably won't be taken as the best match by location-dependent algorithms.

Let's notice that execution time for introduced image comparison algorithms were counted purely, which means that time to read metadata or read image data to build histogram was not included in results. This is one important point because time to read info from image metadata is much faster than to build and normalize image's histogram. Reading and parsing metadata takes 1.4 seconds in average for 5000 images while reading and building histograms for the same dataset takes 22.5 seconds average if images are downscaled first, however we can improve this time if histograms will be pre-calculated and stored in metadata or cache.

Next part of the results presents figures with top picks to see the best match found by each algorithm. Also top 5 picks by each algorithm are shown for an image with a tennis player, because received results illustrate the difference between three offered algorithms the best way (Figure 6 - Figure 9).

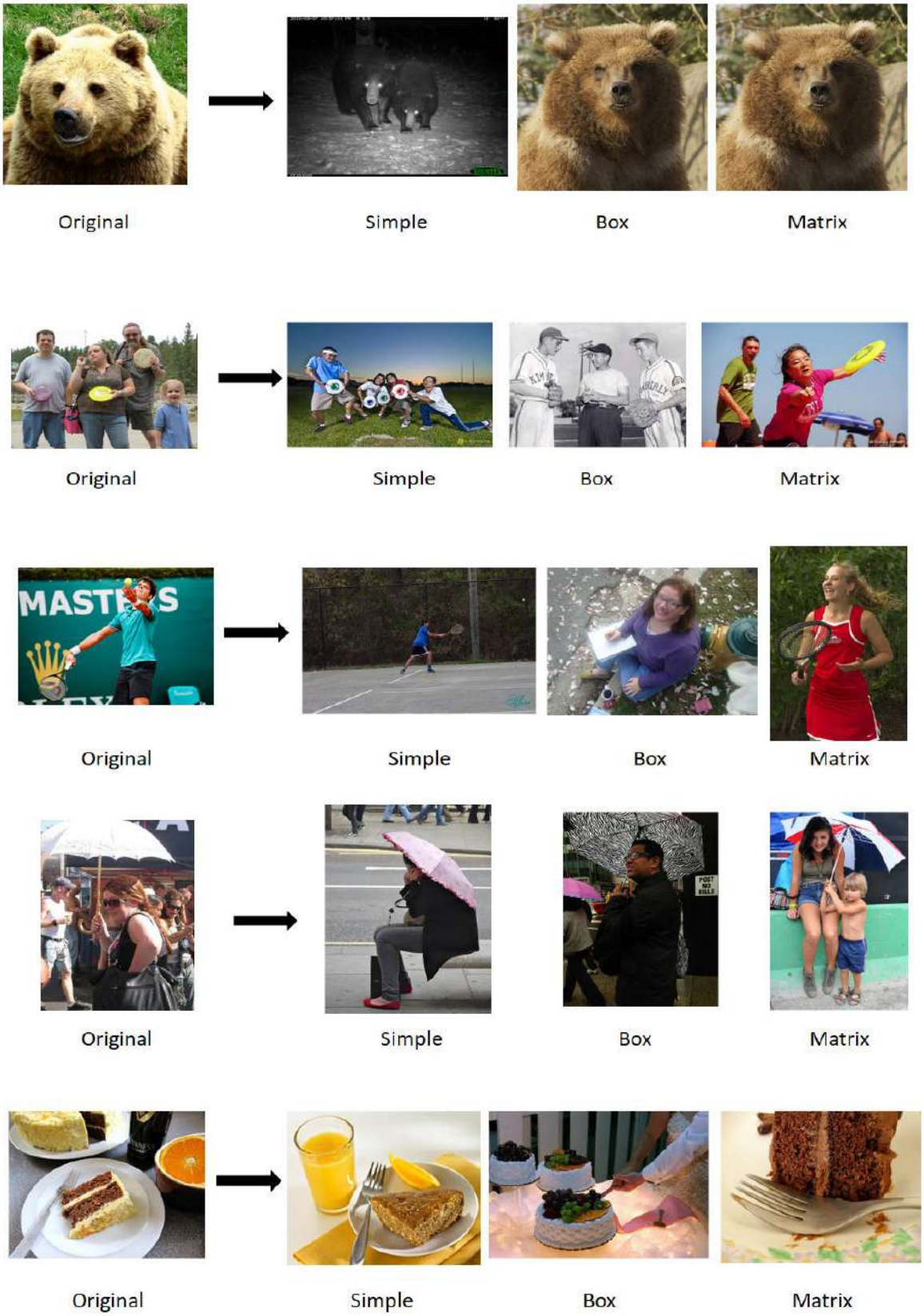

Figure 6: Top matches to images selected by each algorithm (origin – dataset [32])

Figure 7: Top 5 matches to tennis player image by simple algorithm [32]

Figure 8: Top 5 matches to tennis player image by box algorithm [32]

Figure 9: Top 5 matches to tennis player image by matrix algorithm [32]

6. Discussions

First thing to mark in results is execution time. Execution time for simple algorithm averages at 793 ms for 5000 images. Both box and matrix algorithms work with images already filtered by simple algorithm. Box algorithm takes all images that passed 0.1 threshold for simple algorithm since it is fast for small amounts of objects on image, its execution time averaged at 80 ms in our tests for up to 500 images. Matrix algorithm is significantly slower due to scanning images, so we are limiting the amount of images for processing to top-100 when a simple algorithm is done. It averages at 314 ms, the amount of processed images swinged from 50 to 100 samples. In a real application we can use this approach to find similar images, it showed itself as efficient enough for a request execution time. For real scenarios we can introduce further optimizations with caching recently reached files metadata, files storage modifications, etc.

Generally simple algorithm showed highest coefficients as expected, because it does not compute objects location in addition to their types and numbers, of course it does not range images with the same number of objects by their similarity, they are all equal.

We have seen that in 4 out of 5 tests matrix algorithm showed higher similarity coefficients than box algorithm, also different filtering options concluded in a fact that results for matrix algorithm are closer to target image context, while box algorithm sometimes returned images that have objects in the same locations, but do not have any object of a different type.

During experiments all three algorithms showed that depending on the image's content, number and variety of detected objects their approach may be better to get images that look alike from a human point of view, e.g. image of table with food where we had a lot of different objects detected like fork, cup, cake, orange, is matched the best by the simple algorithm from our own point of view it returned results closer to original than location-dependent algorithms for that image. At the same time, box and matrix algorithm showed their dominance for images with a single detected object. Top-picks are much closer to an original than simple algorithms.

It is worth noting that box algorithm is useful for searching images with the same configuration, as an example we can take picture of a family with frisbees, box algorithm's top-5 picks have no frisbees on them, but instead we have images that show us group of three people standing as in original picture. In some sense this algorithm is more straightforward than the matrix algorithm, often it does not capture context, but returns images with the biggest overlap between some of the objects, also this

algorithm depends a lot on each single object unlike other algorithms which depend more on object types than on each single object on them.

Similarity between two images is subjective to each person, but to our mind if we want to choose an algorithm that returns the closest top picks, then the matrix algorithm is the one at least for tested images. From our perspective it became a compromise between a simple algorithm which only cares about context, but not configuration and a box algorithm which is focused mostly on a configuration of objects on image with a little attention to context.

Matrix algorithm became a tool that allows to assess locations of all objects of some type as a whole rather than some individual entities, it is the slowest algorithm of three, but with a few optimizations and tricks it can execute its task in acceptable time.

We introduced histogram comparison algorithm as benchmark to compare some classic algorithm that is used to get image similarity assessment. Results show that reading image's metadata is 16 times faster than reading image's data and building its histogram. When we are comparing time spent on algorithms execution we see that simple algorithm is 2 times faster than histograms comparison and box and matrix algorithms are 1.5-1.7 times faster.

Experiments with algorithms gave us some of the desired results while leaving a lot of space for improvement. First point that can be an objective for future research is giving each object type some weight, e.g. people are more important than frisbees or tennis rackets, thus we can balance the impact of not very important objects on a comparison. Also coefficients can be given to objects of each image individually depending on the area covered by them. As a possible improvement to accuracy comparison we can compare not only locations of objects on image, but also the model's confidence that the sector contains objects of some type. As it was mentioned earlier, matrix algorithm performance is one of the key points of improvement. Implementation of comparison on a graphics card can increase the speed of similarity coefficient computation, which will allow it to run on a bigger volume of input images. Results of this article's experiments showed us the possibility to use machine learning models with files metadata to manage big amounts of data in an efficient way. Correct use of metadata and machine learning models gives us a tool to implement some non-trivial functionality for applications that operate with a lot of data like applications with data lake architecture where all the data is just stored in a raw format.

7. Conclusions

In this work we did research on machine learning models and metadata use in efficient data management and image comparison algorithms. This work was done to check the possibility and efficiency of this approach for modern big data application architectures such as a data lake.

Image comparison algorithms utilize image's metadata service fields as a storage for information about depicted objects. To accomplish this task we offered a solution that uses object detection machine learning model YOLOv5 during image input processing to find all objects on the image and save them in a compact way in the image's metadata.

We developed three different algorithms of image comparison depending on objects types, their numbers and locations on image. Simple algorithm computes similarity coefficients only based on types and number of objects on both images. Box algorithm computes similarity coefficients based on individual objects locations and their intersections on two images. Matrix algorithm computes similarity coefficients based on intersection area occupied by objects of certain image type, handles images with a huge amount of objects on them. We conducted a series of experiments to compare their execution time and received similarity results among themselves and in comparison to well-known histogram correlation to find out how efficient offered algorithms are in comparison.

As a result of experiments we confirmed that image comparison algorithms based on the use of object detection machine learning models and image files metadata service fields are efficient and accurate enough. In comparison to well-known algorithm suggested approach showed 1.5-2x faster execution time which means that our algorithms can be executed in systems with large image data storages in acceptable time. Experiments also confirmed extreme efficiency of parallel computing for boosting performance of image comparison algorithms as these algorithms for most part can be split

easily for parallel execution. One of the most promising directions to research is using GPU for significant acceleration in executing image comparison.

Offered algorithms showed us that each algorithm can be used to receive the best results in search and comparison depending on objective and image configuration. In our opinion, matrix algorithm provides results closest to human recognition since it gives us a compromise solution between searching for images with the same context and images with similar object positioning.

8. References

[1] X. Liu, G. Cheung, C. -W. Lin, D. Zhao and W. Gao, "Prior-Based Quantization Bin Matching for Cloud Storage of JPEG Images," in IEEE Transactions on Image Processing, vol. 27, no. 7, pp. 3222-3235, July 2018, doi: 10.1109/TIP.2018.2799704.

[2] Y. Zheng et al., "Size-Scalable Content-Based Histopathological Image Retrieval From Database That Consists of WSIs," in IEEE Journal of Biomedical and Health Informatics, vol. 22, no. 4, pp. 1278-1287, July 2018, doi: 10.1109/JBHI.2017.2723014.

[3] A. Preethy Byju, B. Demir and L. Bruzzone, "A Progressive Content-Based Image Retrieval in JPEG 2000 Compressed Remote Sensing Archives," in IEEE Transactions on Geoscience and Remote Sensing, vol. 58, no. 8, pp. 5739-5751, Aug. 2020, doi: 10.1109/TGRS.2020.2969374.

[4] X. Wang et al., "A Storage Method for Remote Sensing Images Based on Google S2," in IEEE Access, vol. 8, pp. 74943-74956, 2020, doi: 10.1109/ACCESS.2020.2988631.

[5] H. Dhayne, R. Haque, R. Kilany and Y. Taher, "In Search of Big Medical Data Integration Solutions - A Comprehensive Survey," in IEEE Access, vol. 7, pp. 91265-91290, 2019, doi: 10.1109/ACCESS.2019.2927491.

[6] P. Kathiravelu, A. Sharma and P. Sharma, "Understanding Scanner Utilization With Real-Time DICOM Metadata Extraction," in IEEE Access, vol. 9, pp. 10621-10633, 2021, doi: 10.1109/ACCESS.2021.3050467.

[7] A. G. C. Pacheco and R. A. Krohling, "An Attention-Based Mechanism to Combine Images and Metadata in Deep Learning Models Applied to Skin Cancer Classification," in IEEE Journal of Biomedical and Health Informatics, vol. 25, no. 9, pp. 3554-3563, Sept. 2021, doi: 10.1109/JBHI.2021.3062002.

[8] A. Ebrahimian, H. Mohammadi, M. Babaie, N. Maftoon and H. R. Tizhoosh, "Class-Aware Image Search for Interpretable Cancer Identification," in IEEE Access, vol. 8, pp. 197352-197362, 2020, doi: 10.1109/ACCESS.2020.3033492.

[9] Vynokurova O., Peleshko D., Peleshko M. (2020) Hybrid Deep Convolutional Neural Network with Multimodal Fusion. In: Babichev S., Peleshko D., Vynokurova O. (eds) Data Stream Mining & Processing. DSMP 2020. Communications in Computer and Information Science, vol 1158. Springer, Cham. https://doi.org/10.1007/978-3-030-61656-4_4

[10] K. Smelyakov, A. Chupryna, M. Hvozdiev and D. Sandrkin, "Gradational Correction Models Efficiency Analysis of Low-Light Digital Image," 2019 Open Conference of Electrical, Electronic and Information Sciences (eStream), 2019, pp. 1-6, doi: 10.1109/eStream.2019.8732174.

[11] P. K. Deb, A. Mukherjee and S. Misra, "Fido: A String-Based Fuzzy Logic Mechanism for Content Extraction from UAV Data Lakes," in IEEE Internet of Things Magazine, vol. 4, no. 4, pp. 24-29, December 2021, doi: 10.1109/IOTM.001.2100084.

[12] K. Smelyakov, M. Shupyliuk, V. Martovytskyi, D. Tovchyrechko and O. Ponomarenko, "Efficiency of image convolution," 2019 IEEE 8th International Conference on Advanced Optoelectronics and Lasers (CAOL), 2019, pp. 578-583, doi: 10.1109/CAOL46282.2019.9019450.

[13] Rafael C. Gonzalez, Richard E. Woods Digital Image Processing, 4th. ed., Pearson/Prentice Hall, 2018, 1168p. DOI/ISBN: 9780133356724.

[14] K. Smelyakov, D. Tovchyrechko, I. Ruban, A. Chupryna and O. Ponomarenko, "Local Feature Detectors Performance Analysis on Digital Image," 2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T), 2019, pp. 644-648, doi: 10.1109/PICST47496.2019.9061331.

[15] I. Ruban, H. Khudov, O. Makoveichuk, I. Khizhnyak, V. Khudov, V. Podlipaiev, V. Shumeiko, O. Atrasevych, A. Nikitin, and R. Khudov. Segmentation of opticalelectronic images from on-board systems of remote sensing of the Earth by the artificial bee colony method, Eastern-European Journal of Enterprise Technologies, № 2/9 (98), 2019, pp. 37–45. DOI: https://doi.org/10.15587/1729- 4061.2019.161860.
[16] Z. Zhou, Q. M. J. Wu, S. Wan, W. Sun and X. Sun, "Integrating SIFT and CNN Feature Matching for Partial-Duplicate Image Detection," in IEEE Transactions on Emerging Topics in Computational Intelligence, vol. 4, no. 5, pp. 593-604, Oct. 2020, doi: 10.1109/TETCI.2019.2909936.
[17] F. Fang, L. Li, H. Zhu and J. -H. Lim, "Combining Faster R-CNN and Model-Driven Clustering for Elongated Object Detection," in IEEE Transactions on Image Processing, vol. 29, pp. 2052-2065, 2020, doi: 10.1109/TIP.2019.2947792.
[18] C. -Y. Sun, X. -J. Hong, S. Shi, Z. -Y. Shen, H. -D. Zhang and L. -X. Zhou, "Cascade Faster R-CNN Detection for Vulnerable Plaques in OCT Images," in IEEE Access, vol. 9, pp. 24697-24704, 2021, doi: 10.1109/ACCESS.2021.3056448.
[19] E. Zagan and M. Danubianu, "Cloud DATA LAKE: The new trend of data storage," 2021 3rd International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA), 2021, pp. 1-4, doi: 10.1109/HORA52670.2021.9461293.
[20] C. Giebler, C. Gröger, E. Hoos, H. Schwarz and B. Mitschang, "A Zone Reference Model for Enterprise-Grade Data Lake Management," 2020 IEEE 24th International Enterprise Distributed Object Computing Conference (EDOC), 2020, pp. 57-66, doi: 10.1109/EDOC49727.2020.00017.
[21] Q. Bai, S. Li, J. Yang, Q. Song, Z. Li and X. Zhang, "Object Detection Recognition and Robot Grasping Based on Machine Learning: A Survey," in IEEE Access, vol. 8, pp. 181855-181879, 2020, doi: 10.1109/ACCESS.2020.3028740.
[22] G. Cheng, C. Yang, X. Yao, L. Guo and J. Han, "When Deep Learning Meets Metric Learning: Remote Sensing Image Scene Classification via Learning Discriminative CNNs," in IEEE Transactions on Geoscience and Remote Sensing, vol. 56, no. 5, pp. 2811-2821, May 2018, doi: 10.1109/TGRS.2017.2783902.
[23] D. Kollias and S. Zafeiriou, "Exploiting Multi-CNN Features in CNN-RNN Based Dimensional Emotion Recognition on the OMG in-the-Wild Dataset," in IEEE Transactions on Affective Computing, vol. 12, no. 3, pp. 595-606, 1 July-Sept. 2021, doi: 10.1109/TAFFC.2020.3014171.
[24] V. Appana, T. M. Guttikonda, D. Shree, S. Bano and H. Kurra, "Similarity Score of Two Images using Different Measures," 2021 6th International Conference on Inventive Computation Technologies (ICICT), 2021, pp. 741-746, doi: 10.1109/ICICT50816.2021.9358789.
[25] J. Yu, Z. Hu and Y. Zhang, "An Image Comparison Algorithm Based on Contour Similarity," 2020 International Conference on Computer Network, Electronic and Automation (ICCNEA), 2020, pp. 111-116, doi: 10.1109/ICCNEA50255.2020.00032.
[26] J. M. H. Noothout et al., "Deep Learning-Based Regression and Classification for Automatic Landmark Localization in Medical Images," in IEEE Transactions on Medical Imaging, vol. 39, no. 12, pp. 4011-4022, Dec. 2020, doi: 10.1109/TMI.2020.3009002.
[27] C. -Y. Sun, X. -J. Hong, S. Shi, Z. -Y. Shen, H. -D. Zhang and L. -X. Zhou, "Cascade Faster R-CNN Detection for Vulnerable Plaques in OCT Images," in IEEE Access, vol. 9, pp. 24697-24704, 2021, doi: 10.1109/ACCESS.2021.3056448.
[28] D. Ageyev, T. Radivilova, O. Bondarenko and O. Mohammed, "Traffic Monitoring and Abnormality Detection Methods for IoT," 2021 IEEE 4th International Conference on Advanced Information and Communication Technologies (AICT), 2021, pp. 250-254, doi: 10.1109/AICT52120.2021.9628954.
[29] Proniuk, G., Geseleva, N., Kyrychenko, I., Tereshchenko, G., "Spatial interpretation of the notion of relation and its application in the system of artificial intelligence", 2019 3rd International Conference on Computational Linguistics and Intelligent Systems (COLINS 2019), 2019. – CEUR-WS, 2019, pp. 266-276.
[30] YOLOv4-architecture. URL: https://www.researchgate.net/figure/YOLOv4-architecture-34_fig6_349929458.
[31] Docs.ultralytics.com. YOLOv5. URL: https://docs.ultralytics.com.
[32] Cocodataset.org. URL: https://cocodataset.org/#home. https://docs.ultralytics.com.