Volume 8 • Issue 4 • December 2019



# INTERNATIONAL JOURNAL OF ARTIFICIAL INTELLIGENCE AND EXPERT SYSTEMS (IJAE)

ISSN : 2180-124X

Publication Frequency: 6 Issues / Year

CSC PUBLISHERS
http://www.cscjournals.org

# INTERNATIONAL JOURNAL OF ARTIFICIAL INTELLIGENCE AND EXPERT SYSTEMS (IJAE)

**VOLUME 8, ISSUE 4, 2019**

**EDITED BY**
**DR. NABEEL TAHIR**

ISSN (Online): 2180-124X

International Journal of Artificial Intelligence and Expert Systems (IJAE) is published both in traditional paper form and in Internet. This journal is published at the website http://www.cscjournals.org, maintained by Computer Science Journals (CSC Journals), Malaysia.

IJAE Journal is a part of CSC Publishers

Computer Science Journals

http://www.cscjournals.org

# INTERNATIONAL JOURNAL OF ARTIFICIAL INTELLIGENCE AND EXPERT SYSTEMS (IJAE)

Book: Volume 8, Issue 4, December 2019
Publishing Date: 31-12-2019
ISSN (Online): 2180-124X



IJAE Journal is a part of CSC Publishers
http://www.cscjournals.org


Published in Malaysia

Typesetting: Camera-ready by author, data conversation by CSC Publishing Services – CSC Journals, Malaysia

**CSC Publishers, 2019**

# EDITORIAL PREFACE

The International Journal of Artificial Intelligence and Expert Systems (IJAE) is an effective medium for interchange of high quality theoretical and applied research in Artificial Intelligence and Expert Systems domain from theoretical research to application development. This is the *Fourth* Issue of Volume *Eight* of IJAE. The Journal is published bi-monthly, with papers being peer reviewed to high international standards. IJAE emphasizes on efficient and effective Artificial Intelligence, and provides a central for a deeper understanding in the discipline by encouraging the quantitative comparison and performance evaluation of the emerging components of Expert Systems. IJAE comprehensively cover the system, processing and application aspects of Artificial Intelligence. Some of the important topics are AI for Service Engineering and Automated Reasoning, Evolutionary and Swarm Algorithms and Expert System Development Stages, Fuzzy Sets and logic and Knowledge-Based Systems, Problem solving Methods Self-Healing and Autonomous Systems etc.

The initial efforts helped to shape the editorial policy and to sharpen the focus of the journal. Starting with Volume 9, 2020, IJAE will be appearing with more focused issues related to artificial intelligence and expert system research. Besides normal publications, IJAE intend to organized special issues on more focused topics. Each special issue will have a designated editor (editors) – either member of the editorial board or another recognized specialist in the respective field.

IJAE give an opportunity to scientists, researchers, and vendors from different disciplines of Artificial Intelligence to share the ideas, identify problems, investigate relevant issues, share common interests, explore new approaches, and initiate possible collaborative research and system development. This journal is helpful for the researchers and R&D engineers, scientists all those persons who are involve in Artificial Intelligence and Expert Systems in any shape.

Highly professional scholars give their efforts, valuable time, expertise and motivation to IJAE as Editorial board members. All submissions are evaluated by the International Editorial Board. The International Editorial Board ensures that significant developments in image processing from around the world are reflected in the IJAE publications.

IJAE editors understand that how much it is important for authors and researchers to have their work published with a minimum delay after submission of their papers. They also strongly believe that the direct communication between the editors and authors are important for the welfare, quality and wellbeing of the Journal and its readers. Therefore, all activities from paper submission to paper publication are controlled through electronic systems that include electronic submission, editorial panel and review system that ensures rapid decision with least delays in the publication processes.

To build its international reputation, we are disseminating the publication information through Google Books, Google Scholar, Directory of Open Access Journals (DOAJ), Open J Gate, ScientificCommons, Docstoc and many more. Our International Editors are working on establishing ISI listing and a good impact factor for IJAE. We would like to remind you that the success of our journal depends directly on the number of quality articles submitted for review. Accordingly, we would like to request your participation by submitting quality manuscripts for review and encouraging your colleagues to submit quality manuscripts for review. One of the great benefits we can provide to our prospective authors is the mentoring nature of our review process. IJAE provides authors with high quality, helpful reviews that are shaped to assist authors in improving their manuscripts.

**Editorial Board Members**
International Journal of Artificial Intelligence and Expert Systems (IJAE)

# EDITORIAL BOARD

**Associate Professor, Afaq Ahmed**
Sultan Qaboos University
Oman

**Dr. Muhammad Ali Imran**
University of Surrey
United Kingdom

# TABLE OF CONTENTS

Volume 8, Issue 4, October 2019

## Pages

# A Study of The Relationship Among Parameters of M/X Solar Flares via Association Rules

**Jéssica de Farias Pereira** *jessica.pereira1764@gmail.com*
*School of Technology*
*University of Campinas*
*Limeira, 13484-332, Brazil*

**Ana Estela Antunes da Silva** *aeasilva@ft.unicamp.br*
*School of Technology*
*University of Campinas*
*Limeira, 13484-332, Brazil*

**André Leon Sampaio Gradvohl** *gradvohl@ft.unicamp.br*
*School of Technology*
*University of Campinas*
*Limeira, 13484-332, Brazil*

**Guilherme Palermo Coelho** *guilherme@ft.unicamp.br*
*School of Technology*
*University of Campinas*
*Limeira, 13484-332, Brazil*

**José Roberto Cecatto** *jr.cecatto@inpe.br*
*Astrophysics Division*
*National Institute of Space Research*
*São José dos Campos, 12227-010, Brazil*

**Abstract**

This paper introduces a method to study the relation among parameters that can cause the origin of M/X solar flares. Solar flares, especially flares of types M and X, make the Earth's atmosphere more ionized and have an effect on radio signals, which can cause disruptions in wireless communications. This situation points out to the need for better identification of the parameters involved in M/X solar flares. The method is based on four categorical parameters and their relations. Relations are demonstrated by association rules which were extracted by the APRIORI algorithm and the most promising rules were filtered by support and confidence metrics. Results of the most promising rules had been compared by application to different periods of the 23rd and the 24th solar cycles.

**Keywords:** Solar Flares, Data Mining, Association Rules.

## 1. INTRODUCTION

A basic characteristic of the Sun is that its activities are described by a cycle of eleven years, on average. During this period, activities range from the minimum through the ascending phase of the cycle towards the maximum. From then on, there is a decay phase of the cycle towards the next minimum. Activities follow this typical behavior to the next cycle. What is observed during a cycle, for instance, is a gradual increase in the number of sunspots on the visible solar disk, from the minimum towards the maximum of the cycle, when that number reaches a top level along a few years, followed by a gradual decrease along the decaying phase towards the next minimum. Similar behavior is observed in the total solar irradiance and frequency of energetic solar phenomena.

Among the energetic phenomena observed during those solar active periods is the flare, which is a sudden release of the magnetic energy hoarded within an active region of the solar atmosphere. It is recorded as a strong glare of radiation observed in a wide spectrum from radio waves up to X-rays or gamma-rays that lasts from a few minutes to a few hours, in case of the most energetic flares. The flares can be classified into five classes (A, B, C, M, and X), according to the X-ray flux level recorded in Watt per square meter on a logarithmic scale, being A the lowest and X the highest intensity flares.

This work focuses on the study of M/X solar flares and the relationship among their parameters. Solar flares, when intense, can trigger events that may affect the Earth and our daily activities [1]. The effects caused by these phenomena interfere in technological systems, such as mobile signals, satellites and global positioning devices [2].

A great challenge regarding the solar flare is its appropriate prediction. Once a solar flare can be predicted with due anticipation, it is possible to minimize the damages it causes. Since the 1960s, in the 20th century, many studies have tried to predict the occurrence of solar flares. However, to adequately predict it, there is still the need to understand which parameters are the most important for its occurrence and how they influence flares' triggering. Such parameters include: radio emission, X-ray emission, the number and area of sunspots, and many others [3-6].

We propose the use of data mining [7] to study some of those most important parameters and their relation in solar flares of classes M and X. Data mining is an area of computing that intends to find patterns in datasets instances. Patterns are occurrences of data that have similar characteristics or behave similarly during a specific event. Three main data mining tasks can be performed in datasets: association, classification, and regression. The problem of finding associations among data can be solved by the association task. As its basic premise, the association task has to find elements that imply the presence of other elements in the same transaction in a dataset. This kind of relationship among elements is called association rule. Association rules are presented, in datasets, under the form: IF X THEN Y, where, one set of attributes in X implies another set of attributes in Y.

The main idea of the paper is to apply an association rule algorithm to find out the association rules that can point out the relationship among parameters present in solar flare of classes M and X. Datasets containing data from 1997 to 2016 were considered as a whole. Besides, datasets from the 23th and 24th solar cycles were also used separately. Datasets and their attributes were chosen according to the main parameters used to predict solar flares [8].

This study aimed to: (i) choose a set of parameters on solar flares; (ii) pre-treat the dataset to allow the application of association rules; (iii) apply the APRIORI algorithm of association rules [9] and ; (iv) choose rules according to the best values of support and confidence which were the metrics used to analyze the rules.

Our main contributions are: (i) discovery of relations throughout association rules among some parameters of M/X solar flares; (ii) comparison of association rules found in different periods of solar cycles.

The paper is divided into six parts: Introduction which presented the motivation of this work; Background, in which the supporting concepts are described; a third section which presents some studies on the relationship among solar activities' parameters; Methodology, in which the methods used in the experiments are described; Results and Discussion, in which the experiments are explained and discussed and Conclusion and Future Works, in which the main points of the method and future studies are mentioned.

## 2. BACKGROUND

This section presents the main concepts which were used to create our methodology. As an interdisciplinary study, concepts of both areas – solar flares and association rules – are presented.

### 2.1 Solar Flares

The activities that occur in the Sun, including solar flares, are distributed into periods called solar cycles. Each cycle has, on average, eleven years of duration [10]. A solar flare is a large burst of electromagnetic radiation originated in an active region of the solar atmosphere that can attain the Earth about eight minutes later. It stems from the release of a large amount of magnetic energy that was previously hoarded inside an active region.

An active region of the Sun is an area with strong magnetic fields. Most solar events, like solar flares, are originated in these active regions, as the magnetic fields in active regions are stronger than the average magnetic field of the Sun. Active regions are more common during the peak of the solar cycle.

Sunspots are dark regions observed in the solar photosphere, usually surrounded by brighter areas, which indicates the presence of a strong magnetic field in the form of a spots-group (Figure 1a). Sunspots are indicators of active regions and are usually surrounded by lighter-shaded areas of magnetic disturbance. They seem dark, because they are almost two thousand degrees colder than the surrounding photosphere (5800 K), and constitute the feet of arch-like magnetic structures observed in the solar atmosphere.

Figure 1 shows the flare which has the number region NOAA 9077, and was named the Bastille day flare, observed on July 14th, 2000, at four distinct wavelengths (white light, 1600 Ångströms (Å), 195 Å, and 171 Å) [11]. The white light (WL) image (Figure 1a) was recorded at 10:28, Universal Time (UT), four minutes after peak time of the flare (10:24 UT), while the three ultraviolet (UV) wavelengths were recorded 20 minutes later. In Figure 1a, the white light image, it is possible to see the dark regions according to the sunspot pattern of the active region with identification number NOAA 9077.

The structure of the flare can be observed in Figures 1b, 1c and 1d, by the two-ribbon structure which forms the flare and limits the bright flaring arcades.

According to Qahwaji and Colak [12], the arrival of solar X-rays which are emitted by solar flares and travel at the speed of light can disrupt point-to-point high frequency radio communications. Besides, concurrent with X-ray emission, solar activity often emits radio noise that can interfere with communications and radar systems and can shorten the life of satellites. This decrease in the useful life of satellites is caused by the emission of high-energy particles [13].

Sello [14] describes the study of the activities of solar cycles, particularly, their prediction, as a challenging task due to the high frequency of solar radiation, the lack of a theoretical quantitative model of the magnetic cycle of the Sun, the noise in obtaining the data and the high variability in the phase and amplitude of cycles. Besides, the temporal aspect of solar activities involves the consideration of a set of time ranges in different kinds of indices of solar activity.

Because of these complex characteristics of solar activities, it is important to study the relations among parameters of solar flares, which is the focus of this research paper, in order to better understand solar phenomena.

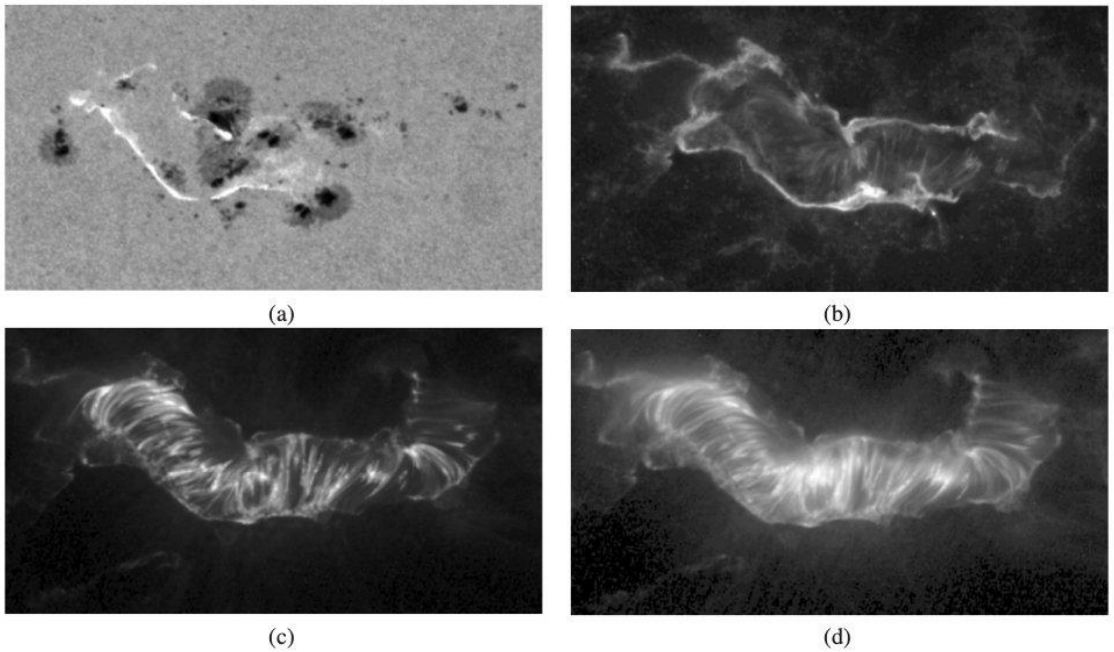



**FIGURE 1:** Multi-wavelength trace observations of the Bastille day flare occurred in the active region NOAA 9077 on July 14th of 2000. (a) WL image taken at 10:28 UT; (b) 1600 Å image taken at 10:48:24 UT; (c) 195 Å image taken at 10:48:38 UT; (d) 171 Å image taken at 10:48:32 UT [11].

### 2.2 Parameters of Solar Flares

Solar flares are regularly observed and recorded by various indexes coded in a numerical form as, for instance: area, magnetic classification of sunspots and the integrated fluxes measured in X-ray [8]. These various indexes, whose values can be found in the databases of the Space Weather Prediction Center [15], represent, in different ways, the characteristics of solar activities.

One of the sunspots classifications is the called Mt. Wilson, which is based on the distribution of magnetic polarities within specific groups, and which has eight sunspot classification classes: alpha, beta, beta-gamma, beta-delta, gamma, delta, beta-gamma delta, and gamma-delta [16]. There are two other sunspot classifications: the Zurich classification [17] and the McIntosh classification. In this study, the Mt. Wilson classification was adopted. The X-ray flux emitted by the Sun is another solar parameter regularly registered, corresponding to the background level of solar X-ray emission. As mentioned before, the X-ray flux, in Watt per square meter, is measured in the wavelength band of 1 to 8 Ångström and classified into five orders of magnitude on a logarithmic scale, as shown in Table 1 [18].

| **X-ray flux (1 to 8 Ångströms)** | **Solar Flare Class** |
|---|---|
| flux < $10^{-7}$ | A |
| $10^{-7}$ < flux < $10^{-6}$ | B |
| $10^{-6}$ < flux < $10^{-5}$ | C |
| $10^{-5}$ < flux < $10^{-4}$ | M |
| flux > $10^{-4}$ | X |

**TABLE 1:** Classification of Flares In Terms of X-ray Flux Level.

### 2.3 Association Rules For Solar Flares

The exploration of frequent patterns refers to the search for recurring relationships in a given dataset. The idea of exploring these patterns is to discover interesting associations among attributes and the frequency with which these associations appear in the dataset [7]. A typical example of the exploitation of these patterns is the market basket transaction. Market basket transactions reflect the purchases made by certain customers in a store [19]. Each transaction is unique and contains the items that were purchased. An example of a market basket transaction could be a set of items purchased by a customer in a supermarket in a specific date. The basket could contain the products: milk, bread, butter and beer, meaning that these products were purchased at the same time by this customer. In this case, the set of items is considered as a transaction, and an identification number, known as transaction identification, would index each transaction.

An association rule represents the relationship found among items in a transaction, using an implication relation. Formally, a set of transactions $T = \{t_1, t_2, \ldots, t_n\}$, and a set of items $I = \{i_1, i_2, \ldots, i_n\}$, where a single transaction $t_n$ contains a subset of items I called item-sets, are the basic concepts of association rules. Therefore, based on these concepts, it is possible to define an association rule as an $X \rightarrow Y$ rule, where X and Y are subsets of I and $X \cap Y = \varnothing$ [20].

By bringing the idea of market basket transactions to the solar flare context, we would have transactions whose items would be parameters of solar flares, as shown by the association rule: {area_huge, beta_gamma_delta, radio_veryhigh} → {Xray_X}.

From the example of the supermarket purchase, an association rule could assume the form of: {bread, butter} → {milk, beer}. In this case, X = {bread, butter} and Y = {milk, beer}. The rule can be described as: a customer who bought bread and butter also bought milk and beer.

For the example of the solar flare context, X = {area_huge, beta_gamma_delta, radio_veryhigh} and Y = {Xray_X}. This means that the combination of these parameters in the antecedent part of the rule results in the Xray_X parameter in the consequent part, which means that this rule represents a class-X solar flare.

According to Han, Kamber and Pei [7], to filter association rules from a dataset, the item-sets of the respective transactions must be selected. The minimum frequency of these item-sets in the general dataset is called support. For such item-set to become an interesting association rule, this minimum frequency number must be established and considered in rule creation, i.e., the item-set only becomes a rule when it reaches a minimum support value. The support of a rule is measured by the ratio of the number of transactions that contain a particular set of items and the total number of transactions. Formally, the support can be described by the formula given by Equation 1.

$$support\ (X \rightarrow Y) = \frac{supp(X \cup Y)}{|N|} \qquad (1)$$

Where: X is the set of items presented in the antecedent of the rule – items in the left side of the rule; Y is the set of items presented in the consequent of the rule – items in the right side of the rule; N is total number of transactions and *supp*(X ∪ Y) is the number of transactions in which item-sets of sets X and Y appear together [19].

In addition to the support, to filter an association rule, it is necessary to establish a minimum confidence level. While the support value can be used to filter out rules that are interesting because they impact the entire database, the confidence value provides the estimated probability of Y given X, i.e., P(Y|X). It shows the probability of, given Y as the consequent of the rule, X is the consequent [19]. Equation 2 shows how to calculate the confidence of a rule that contains X in the antecedent and Y in the consequent.

$$confidence\ (X \rightarrow Y) = \frac{supp(X \cup Y)}{supp(X)} \qquad (2)$$

It is important to explain that an association rule does not imply a cause-effect relation, but it indicates a strong relationship among the items present in the antecedent and consequent parts of the rule.

The objective of this study is to find out which parameters appear together in rules that result into solar flares of the types X and M. To do that, the consequent of the desired rules should contain item-sets that indicate the presence of M/X solar flares.

## 3. WORKS ON THE RELATIONSHIP AMONG SOLAR ACTIVITIES PARAMETERS

Studies that used association rules as a way of studying the parameters of solar flares were not found in the literature. Most of them presented statistical correlations between parameters of solar flares.

It is important to highlight that, unlike the statistical correlation that measures the relationship between two variables as, for example, the linear correlation, the discovery of an association rule indicates the probability of occurrence of a set of items given that some items also occur. Therefore, the idea of reporting the following studies is not about performing a numerical comparison to our work, but to show the importance of some parameters of solar activity and the variety of studies developed to analyze the relationship among these parameters.

Another point about this section is that most of the works cited here used different periods of the solar cycles in their studies, which also prevents a direct comparison between these works and ours. Following are some works which were taken into account because of the relevance of the solar parameters cited by them and the relation cited among such parameters.

According to [8], most studies involving solar activities and their impact on Earth have used the number of sunspots as the main parameter. Later on, the X-ray flux also began to be used. In recent years, many other parameters have been used to study the relationship among parameters in solar activities. Authors describe a correlative study using the monthly data of the following solar activity parameters: sunspot numbers, solar flux, grouped solar flares, coronal index and tilt angle for the solar cycles from 19 to 23.

Feng, Yu and Yang [21] investigated the relationship between grouped solar flares and sunspot numbers during the time interval from Jan-1965 to Jun-2008. Grouped Solar Flares is another type of solar index. The term grouped means that the solar flares observations were made in different locations by different solar observatories and then considered as one [5]. It was found that the relationship between grouped solar flares and sunspot numbers is not only time-dependent but also frequency dependent. According to the authors, the relationship between grouped solar flares and sunspot numbers is a complex nonlinear relationship, despite being highly correlated with each other.

In the study of Meera, Munika and Shrivastava [5], several parameters involved in solar flares were considered to study the relationship among them. The parameters considered were: sunspot numbers, solar flux (10.7 cm flux), grouped solar flare, solar flare index, and coronal index for the last five 11-year solar cycle (20 to 24). According to the authors, the sunspot number is the most famous and popular index for studying the solar activity, being the most analyzed time series in the solar physics. The sunspot number was the solar activity parameter that showed the highest degree of correlation with other parameters. The highest degree of correlation was between sunspot numbers and solar flux.

In the study by Shen, Dun, Zhang, and Jiang [22], the Principal Component Analysis (PCA) method was used to analyze the main parameters of the solar active regions with solar proton events. Using the PCA method, the main parameters of solar active regions were analyzed to get the scores of the principal components of the active regions involved in solar activities. According to the authors, the main parameters to represent the solar activity strength are the relative sunspot number and 10.7 cm radio flux.

The objective of current study is to discover association rules among parameters that are involved in solar flares of classes M and X. As described by the previous studies, some of the most studied solar flare parameters are: X-ray flux, radio flux and number of sunspots. In addition, the Grouped Solar Flares index showed an association with the sunspot number, reinforcing the importance of the sunspot number during solar flares. The parameters X-ray flux, radio flux, number of sunspots and the Mt. Wilson classification were used in this work.

## 4. METHODOLOGY

This section presents the preparation of the transactional dataset, the choice of the period of the solar cycle, and the used tools and algorithms.

### 4.1 Preparation of The Transactional Dataset

A transactional dataset contains a transaction as a register of the dataset. This kind of dataset is used when the association task is applied to data to discover association rules. The transaction contains its identification and all its attributes. In the case of solar flares, a transaction can be explained as an occurrence of a solar flare in one specific solar region.

Two main problems were found when creating a transactional dataset for solar region events: one problem regarding data storage and another regarding data format. With respect to the storage problem, it was necessary to consider that the solar data are distributed into several datasets provided by the SWPC [15,23]. Data used in the current work were extracted from two of these datasets: Solar Region Summary (SRS) and Solar and Geophysical Activity Summary (SGAS). Information of how to download data from SRS and SGAS datasets can be found in [24,25].

From the SRS data collection, the following attributes were used: region identification, the sunspot area, and Mt. Wilson magnetic classification. From the SGAS data collection, attributes used were: X-ray data flux and radio flux.

From these two datasets - SGAS and SRS -, it was necessary to merge all chosen attributes into a single dataset. It is important to point out that for one region in the SRS dataset, there can be several related events in the SGAS dataset. This can be noticed in the second and third lines of Table 2, which shows the merged dataset. Transactions 2 and 3 relate to the same region, but have different X-ray classifications. Therefore, these two events were considered as different transactions in the resulting dataset.

| Trans.ID | Date (Y M D) | | | Reg. ID | Spot Area (Millionths of the Solar Disk Area) | Magnetic Classification (Mt. Wilson) | X-ray flux ($W/m^2$) (in the wavelength range of 1 to 8 Ångströms) | Radio flux (10.7cm) |
|---|---|---|---|---|---|---|---|---|
| 1 | 2000 | 3 | 3 | 8898 | 10 | Alfa | $3.7.10^{-5}$ | 220 |
| 2 | 2000 | 3 | 4 | 8882 | 850 | beta-gamma | $3.8.10^{-5}$ | 240 |
| 3 | 2000 | 3 | 4 | 8882 | 850 | beta-gamma | $6.1.10^{-6}$ | 200 |

**TABLE 2:** Example of transactions resulting from the merging of SGAS and SRS datasets.

Besides the data storage problem, another problem faced when creating the transactional dataset was the data format, as solar flares data are presented into diverse formats. Because the APRIORI algorithm demands categorical data as input, it was necessary to preprocess all numerical data, transforming them into categorical data.

As it can be seen in Table 2, the parameters used in the experiments were: sunspot area, magnetic classification of the event, X-ray flux and radio flux. The only categorical parameter is the magnetic classification. The other three parameters (spot area, X-ray and radio fluxes) are numerical. To use them in the APRIORI algorithm, it was necessary to have them transformed into categorical attributes. The attributes were discretized according to the classes showed in Tables 1, 3 and 4. For the X-ray flux, in Watt per square meter, which is measured from the wavelength band of 1 to 8 Ångström, the categorization presented in Table 1 was used [18].

For the background radio flux, a categorization was created according to the flux values observed during solar cycles 23 and 24 [15]. Being quantified in “solar flow unit” (sfu), F10.7 radio flux can range from 50 sfu–300 sfu over a solar cycle. It is considered that the annual mean value of 10.7 cm radio flux larger than 180 sfu corresponds to the solar maximum year, while the annual mean value of 10.7 cm radio flux smaller than 90 sfu corresponds to the solar minimum year [22]. Considering these values, four levels of flux were established: “low”, “medium”, “high”, and “very high” as shown in Table 3.

| **Radio flux (10.7 cm frequency)** | **Category** |
|---|---|
| < 80 | Low |
| 80 a 120 | Medium |
| 120 a 160 | High |
| > 160 | Very high |

**TABLE 3:** Radio Flux Categorization.

For the sunspot area, a categorization was created according to Table 4, following the same way of dividing data into four classes.

| **Spot area (Millionths of the Solar Disk Area - MADS)** | **Category** |
|---|---|
| 0 a 200 | Small |
| 200 a 500 | Medium |
| 500 a 1000 | Large |
| > 1000 | Huge |

**TABLE 4:** Categorization of the sunspot area.

### 4.2 Selection of The Solar Cycles’ Periods

As explained in Section 1, the solar cycle has an average period of eleven years. This eleven-year period has a maximum phase when most of the flares normally occur. Cycles usually have similar behavior, however, there may be differences in events’ behavior between cycles. Considering this fact, three experiments were carried out according to three different periods in order to compare the resulting rules.

For the first experiment, all the data from 1997 to 2016 were used. The main goal of this experiment was to evaluate the association rules that would be generated considering two whole solar cycles (23rd and 24th solar cycles), independently of the period of the solar cycle.

The second and third experiments used specific periods of the 23rd and the 24th solar cycles, respectively. To do so, the most active parts of the cycles were selected. The most active parts are those which show the highest number of flares of classes M and X and are, generally, concentrated between the year of the maximum and the following years [26]. For the 23rd solar cycle, the period from the year 2000 to the year 2003 was used. For the 24th solar cycle, the period from the year 2012 to the year 2015 was considered. Those periods presented the highest levels of solar events, therefore presenting the highest number of M and X solar flares. The goal of the last two experiments was to compare the two sets of rules between the two cycles.

### 4.3 Tools, Algorithms and Metrics

The following tools, algorithms and metrics were applied to the transactional dataset showed in Table 2:

- The APRIORI algorithm was applied to all datasets used in the experiments of Section 5, and libraries of the R framework – a free software environment for statistical computing and graphics creation [27] – were used to apply the APRIORI algorithm;

- For data pre-processing and dataset joining, functions were developed using JAVA language;

- The association rules generated in each set of experiments were analyzed according to the support and confidence metrics.

## 5. RESULTS AND DISCUSSION

As explained in Section 4.2, three experiments were performed according to different periods. The attributes used in all experiments were the same. The attributes used, as explained in Section 4.1, were:

- Classification of the spot area in Millionths of the Solar Disk Area;
- Magnetic classification of the spot by the Mt. Wilson classification;
- Classification of solar flares by the measured flux in X-rays in the band from 1 to 8 Ångströms;
- Classification of solar flares by the radio flux emission at a wavelength of 10.7 cm.

The results of the experiments were described based on the resulting association rules. The rules that presented X or M X-ray classification in their consequent part were selected for analysis. Consider, for example, the rule: {area_huge,beta_gamma_delta,radio_veryhigh}→{Xray_X}. In this example, the antecedent of the rule is a set of three parameters and the consequent, a set of one parameter. The rule can be read as: when a solar explosion of intensity X occurs (X_ray_X), the value of the parameter area is bigger than 1000 MADS (area_huge), the sunspot classification is beta_gamma_delta and radio emission is higher than 160 sfu (radio_veryhigh) (categorizations of the parameters were explained in Section 4.1).

The support and confidence metrics were used to show how often this association occurs and the probability of its occurrence (support and confidence were explained in Section 2.3). The three experiments are explained in the following sections.

### 5.1 Experiment I: Data From 1997 to 2016

For the first experiment, all the data provided by the SWPC from 1997 to 2016 were used. This data period was chosen to assess the kind of association rules that would be generated considering the solar cycles 23 and 24, independently of the period of the solar cycle. Following is the description of the experiment.

- Period: 1997 to 2016;
- Number of considered regions: 1,414;
- Support and Confidence: we selected rules of classes M and X with the highest values of support and confidence. The minimum support was 1%.

Table 5 shows the rules and their support and confidence. The first rule presents the highest support for the rules containing the class-X flares in the consequent part of the rule. From rule number 1 it is possible to conclude that in almost 10% of the transactions of the dataset, the parameters radio-veryhigh and Xray_X appear together in the transactions. The confidence of rule number 1 shows that in about 26% of the rules in which Xray_X is the consequent of the rule, the radio_veryhigh is the antecedent. Rule number 2 presents the highest confidence for the rules containing the class-X flares in the consequent part of the rule.

Rule number 3 presents the highest support for the rules containing class-M flares in the consequent part of the rule. Rule number 4 presents the highest confidence for class-M flares in the consequent part of the rule.

| # | Support | Confidence | Association rule |
|---|---|---|---|
| 1 | 9.83% | 25.78% | *{radio_veryhigh} → {Xray_X}* |
| 2 | 2.26% | 43.24% | *{area_huge, beta_gamma_delta, radio_veryhigh} → {Xray_X}* |
| 3 | 23.97% | 62.89% | *{radio_veryhigh} → {Xray_M}* |
| 4 | 2.47% | 83.33% | *{beta_gamma_delta, radio_high} → {Xray_M}* |

**TABLE 5:** Selected rules for M/X flares for the 1997–2016 period.

### 5.2 Experiment II: Data From The 23rd Solar Cycle

For the second experiment, data from the 23rd solar cycle were considered. The goal was to assess the rules generated by events that occurred in the period between 2000 and 2003 which presented the highest number of solar activities in the23rd solar cycle. The second experiment can be described as follows.

- Period: 2000 to 2003;
- Number of considered regions: 665;
- Support and Confidence: we selected rules of classes M and X with the highest values of support and confidence. The minimum support was 1%.

Table 6 shows the rules and their support and confidence. Similarly to Table 5, in Table 6, the first rule presents the highest support for rules containing class-X flares in the consequent part of the rule. From rule number 1, it is possible to conclude that in almost 8.5% of the transactions of the dataset, the attributes radio_veryhigh and Xray_X appear together in the transactions. The confidence of rule number 1 shows that in almost 26% of the rules in which Xray_X is the consequent of the rule, the radio_veryhigh is the antecedent.

| # | Support | Confidence | Association rule |
|---|---|---|---|
| 1 | 8.42% | 25.45% | *{ radio_veryhigh} → {Xray_X}* |
| 2 | 1.95% | 43.33% | *{area_huge, beta_gamma_delta, radio_veryhigh} → {Xray_X}* |
| 3 | 27.21% | 60.53% | *{radio_low} → {Xray_M}* |
| 4 | 1.20% | 100% | *{area_medium, beta_gamma_delta, radio_medium} → {Xray_M}* |

**TABLE 6:** Selected rules for flares of classes M and X for the 2000–2003 period.

Rule number 2 presents the highest confidence for the all rules which present class-X flares in the consequent part of the rule. Rule number 3 presents the highest support for the rules containing class-M flares in the consequent part of the rule. Rule number 4 presents the highest confidence for the rules containing the M type of flare in the consequent part of the rule.

### 5.3 Experiment III: Data From The 24th Solar Cycle

For the third experiment, data from the 24th solar cycle were considered. The objective was to assess the rules generated by events that occurred between 2012 and 2015. This period presented the highest number of solar activities for the 24rd solar cycle. Following is the description of the experiment.

- Period: 2012 to 2015;
- Number of regions considered: 239;
- Support and Confidence: we selected rules of classes M and X with the highest values of support and confidence. The minimum support was 1%.

Table 7 shows the rules and measures of support and confidence for flares of classes M and X.

| # | Support | Confidence | Association rule |
|---|---|---|---|
| 1 | 13.80% | 27.96% | *{ radio_veryhigh} → {Xray_X}* |
| 2 | 2.09% | 62.50% | *{beta_delta, radio_veryhigh} → {Xray_X}* |
| 3 | 35.14% | 72.41% | *{beta_gamma_delta} → {Xray_M}* |
| 4 | 2.09% | 100% | *{area_huge, beta_gamma_delta, radio_low} → {Xray_M}* |

**TABLE 7:** Selected rules for flares M and X for the 2012–2015 period.

Similarly to Table 6, in Table 7 the first rule presents the highest support for the rules containing the class-X flare in the consequent part of the rule. From rule number 1, it is possible to conclude that in almost 14% of the transactions of the dataset, the parameters radio_veryhigh and Xray_X appear together in the transactions.The confidence value of rule number 1 shows that in almost 28% of the rules that have Xray_X as the consequent of the rule, radio_veryhigh is the antecedent.

Rule number 2 presents the highest confidence for the rules containing the X type of flare in the consequent part of the rule. Rule number 3 presents the highest support for the rules containing the class-M flare in the consequent part of the rule, while rule number 4 presents the highest confidence for the rules containing the class-M flare in the consequent part of the rule.

### 5.4 Comparison of Results

Tables 8 and 9 show a comparison among the generated rules for the three experiments. Table 8 shows all the rules generated with Xray_X in the consequent. Table 9 shows all the rules generated with Xray_M in the consequent.

In Table 8, rule 1 {radio_veryhigh}→{Xray_X} appears in all three experiments as the rule with the highest support level, indicating that, among all the other rules with Xray_X in the consequent, this rule is the most frequent one. For rules 1, 3, and 5, the antecedent is the radio_veryhigh, indicating that these two attributes have the highest association in the dataset. Although their support values are the highest, their confidence remained at around 26%, which indicates that in 26% of the cases, when Xray_X appears in the consequent, radio_veryhigh appears in the antecedent.

Rules 2 and 4 agree that when the X_rayX is the consequent, there is a probability of around 43% that the antecedent part of the rule is formed by the attributes: area_huge, beta_gamma_delta and radio_veryhigh.

Despite not being very frequent, rule 6 shows the highest confidence level, indicating that the probability of an X solar flare is 62.5% when beta_delta and radio_veryhigh appear together. In all rules the attribute radio_veryhigh is present. The area attribute in two rules always assumes huge value. The attribute radio_very_high is present in all rules.

| # | Period | Supp. | Conf. | Association rule |
|---|---|---|---|---|
| 1 | 1997–2016 | 9.83% | 25.78% | *{radio_veryhigh}→{Xray_X}* |
| 2 | 1997–2016 | 2.26% | 43.24% | *{area_huge, beta_gamma_delta, radio_veryhigh}→{Xray_X}* |
| 3 | 2000–2003 | 8.42% | 25.45% | *{radio_veryhigh}→{Xray_X}* |
| 4 | 2000–2003 | 1.95% | 43.33% | *{area_huge, beta_gamma_delta, radio_veryhigh}→{Xray_X}* |
| 5 | 2012–2015 | 13.80% | 27.96% | *{radio_veryhigh}→{Xray_X}* |
| 6 | 2012–2015 | 2.09% | 62.5% | *{beta_delta, radio_veryhigh}→{Xray_X}* |

**TABLE 8:** Comparison of rules with *Xray_X* in the consequent.

Table 9 presents rules that contain Xray_M in the consequent. These rules present an interesting behavior, showing relatively low support, but high confidence, particularly rules 2, 4 and 6. This kind of rule can represent an interesting relation for not being so frequent, but very reliable. This means that when it occurs, there is a high possibility of the association among the attributes of the rule. Another characteristic of these set of rules is that most of them (2, 4, 5 and 6) contain the attribute beta_gamma_delta.

| # | Period | Supp. | Conf. | Association rule |
|---|---|---|---|---|
| 1 | 1997–2016 | 23.97% | 62.9% | *{radio_veryhigh}→{Xray_M}* |
| 2 | 1997–2016 | 2.47% | 83.3% | *{beta_gamma_delta, radio_high}→{Xray_M}* |
| 3 | 2000–2003 | 27.21% | 60.5% | *{radio_low}→{Xray_M}* |
| 4 | 2000–2003 | 1.20% | 100% | *{area_medium, beta_gamma_delta, radio_medium}→{Xray_M}* |
| 5 | 2012–2015 | 35.14% | 72.4% | *{beta_gamma_delta}→{Xray_M}* |
| 6 | 2012–2015 | 2.09% | 100% | *{area_huge, beta_gamma_delta, radio_low}→{Xray_M}* |

**TABLE 9:** Rules with *Xray_M* in the consequent.

If a confidence higher than 80% was considered here, only the rules 2, 4 and 6 would be present. This would imply the presence of the attribute beta_gamma_delta in all rules. If we had a confidence of 100%, only rules 4 and 6 would be considered. For these two rules, the radio attribute assumes medium and low values, respectively. This is an interesting information to be further investigated as, it is well known that, the beta_gamma_delta classification is, generally, present in solar flares of type X. However, when the value of the radio emission is medium or low, even in the presence of a beta_gamma_delta sunspot, the solar flare was of M class.

## 6. CONCLUSIONS AND FUTURE WORKS

This article focused on the study of M/X solar flares and the relationship among their parameters. The parameters here analyzed were: classification of the sunspot area; magnetic classification of the sunspot; classification of solar flares by the measured flux in X-rays, and classification of solar flares by the radio flux emission. Support and confidence metrics were used to analyze all experiments.

Three periods were chosen: 1997–2016; 2000–2003 and 2012–2015. The rules that presented M or X classification in their consequent part were selected for analysis. Among those rules, the ones that presented higher values of support and confidence were filtered and compared for the three-period experiments.

For all experiments, regardless the period of the solar cycle, authors found association rules

relating high radio flux emission (> 160 sfu) with class- X solar flare. This indicates the importance of high radio emission in the occurrence of class-X solar flares.

Regarding the sunspot classification, only the betta_gamma_delta and beta_delta classifications were present in the rules indicating class-X solar flares. Those rules presented the highest values of confidence, indicating the importance of those sunspots' classifications in class-X solar flares. For the spot area, only huge areas were found in the most promising rules of class-X solar flares. However, the rules presented low confidence. Hence, it is only possible to point out that spots with huge areas appeared in class-X solar flare association rules.

For the rules with class-M solar flares in the consequent, all filtered rules presented a level of confidence much higher than rules of class-X solar flares, including two rules with confidence of 100%. One of the differences among rules of M and X solar flares was the radio flux emission. While in the class-X solar flares rules only high emissions of radio appeared, in the class-M solar flare rules, all values of radio flux emissions appeared (low, medium, high and very high). This indicates that it was not possible to point out a value for radio flux and class-M solar flares. However, it is clear that when class-M solar flares occur, radio emission is always found.

In four of the five rules of class-M solar flares, the beta-gamma-delta sunspot classification was found, which reinforces the importance of this sunspot classification for the analysis of class-M solar flares.

The sunspot area parameter was found in two of the five rules with different values (huge and medium). Therefore, it is not possible to observe a very consistent behavior of the spot area parameter for class-M solar flares association rules.

Considering all results, it was possible to establish some relations among the proposed parameters and solar flares of classes M and X, regardless the selected period. It was noticeable that, by using association rules, it was possible to synthesize the relations of frequently used parameters of solar flares of classes M and X.

As mentioned in Section 3, works which used association rules in order to evaluate relations among solar flare parameters were not found. However, a recent work [28], mentions the use of a spatiotemporal association rule mining algorithm in solar images. This kind of association rule involves, besides the parameters´ values of solar images, their time and space properties, which means that, these are association rules which contain parameters´ values as well as the period and the solar region where their occurred. Authors obtained solar images and from them extracted parameters which were submitted to the spatiotemporal association rule mining algorithm. Results showed that the tool would help the understanding of solar properties by the extraction of solar images and the use of temporal association rules in order to evaluate the evolution of the images.

In future studies, authors intend to create a method to study the temporal behavior of the solar cycle. The intention is to use an association rule algorithm which also considered time as an attribute of the rule. The idea is to show, for each part of the solar cycle, the most relevant association rules which could show relations among parameters that resulted in M and X solar flares. This study is similar to [28] in terms of the use of temporal association rules. However, instead of considering solar images and their related descriptions, the magnetic parameters of the active regions of the Sun would be considered. An example of the use of magnetic parameters of active regions in order to predict solar flares can be found in [29].

## 7. ACKNOWLEDGEMENTS

The authors would like to acknowledge the financial support for this research through the grant #2017/14836-4 received from the São Paulo Research Foundation (FAPESP).

The authors thank Espaço da Escrita – Pró-Reitoria de Pesquisa, UNICAMP – for the provided language services.

## 8. REFERENCES

[1] T. Colak, R. Qahwaji. “Automated Solar Activity Prediction: A hybrid computer platform using machine learning and solar imaging for automated prediction of solar flares”. Space Weather. vol. 7, pp 1-12, Jun. 2009.

[2] K. Fox. “Impacts of Strong Solar Flares. 2013”. Internet: https://www.nasa.gov/mission_pages/sunearth/news/flare-impacts.html#.WAGF2uArLIV. [Sep. 1, 2019].

[3] M. Dierckxsens, K. Tziotziou, S. Dalla, I. Patsou, M. S. Marsh, N. B. Crosby, O. Malandraki, G. Tsiropoula. “Relationship between Solar Energetic Particles and Properties of Flares and CMEs: Statistical Analysis of Solar Cycle 23 Events”. Solar Physics. vol. 290(3), pp.841-874, Mar. 2015.

[4] C. Yanmei, W. Huaning. “Correlation between solar flare productivity and photospheric vector magnetic fields”. Advances in Space Research. vol. 42 (9), pp. 1475–1479, Nov. 2018.

[5] G. Meera, R. Munika, A. K. Shrivastava. “Various Solar Activity Parameters and their Interrelationship from Solar cycles 20 to 24”. International Research Journal of Science and Engineering. vol. 5 (5), pp 59-69. May. 2017.

[6] K. D. Leka, G. Barnes. “Solar Flare Forecasting: Present Methods and Challenges” in Extreme Events in Geospace, 1st ed., vol. 1. N. Buzulukova, Ed. Elsevier, 2017. pp 65-98.

[7] J. Han, M. Kamber, J. Pei. “Data Mining: Concepts and Techniques”. 3rd. ed. Morgan Kaufmann, 2012.

[8] G. Meera, V. K. Mishra, A. P. Mishra. “Solar activity parameters and their interrelationship: Continuous decrease in flare activity from solar cycles 20 to 23”. Journal of Geophysical Research, vol. 112. pp 1-10. May, 2007.

[9] R. Agrawal, R. Srikant. “Fast algorithms for mining association rules in large databases,” in Proc VLDB, 1994, pp 487-499.

[10] A. Loskutov, I. A. Istomin, K. M. Kuzanyan, O. L. Kotlyarov. “Testing and forecasting the time series of the solar activity by singular spectrum analysis.” arXiv preprint – ArXiv, nlin/0010027. [Online] Available: https://arxiv.org/abs/nlin/0010027 [Nov. 24, 2019].

[11] F. Zuccarello. “Multi-spectral observations of flares.” Astronomische Nachrichten, vol.337(10), pp.1070. Nov. 2016.

[12] R. Qahwaji, T. Colak. “Automatic Prediction of Solar Flares using Machine Learning: Practical Study on the Halloween Storm,” in Proc. RAST 2007. pp. 739-742.

[13] A. Ajabshirizadeh, N. M. Jouzdani, S. Abbassi. “Neural network prediction of solar cycle 24”. Research in Astronomy and Astrophysics, vol. 11(4), p. 491-496, 2011.

[14] S. Sello, “Time Series Forecasting: A Nonlinear Dynamics Approach. Termo-Fluid Dynamics Research Center.” arXiv preprint – ArXiv, physics/9906035. [Online] Available: https://arxiv.org/abs/physics/9906035 [Nov. 24, 2019].

[15] Space Weather Prediction Center: NOAA / NWS Space Weather Prediction Center. 2015. Internet: http://www.swpc.noaa.gov. [Aug. 17, 2019].

[16] R. Qahwaji, T. Colak. "Automated Prediction of Solar Flares Using Neural Networks and Sunspots Associations". Advances in Soft Computing. vol 39. pp. 316 – 324. Jun. 2007.

[17] T. T. Nguyen, C. P. Willis, D. J. Paddon, H. S. Nguyen. "On Learning of Sunspot Classification," in Intelligent Information Processing and Web Mining, 1st ed., vol. 25. M. A. Kłopotek, S. T. Wierzchoń, K. Trojanowski (eds). Berlin, Heidelberg: Springer. 2004. pp 59-68.

[18] C. Guo B. Xue, Z. Lin. "Study on the Prediction Method of Characteristic Parameters of Solar X-ray Flares". Chinese Astronomy and Astrophysics. vol 37(3). pp. 255–265, Jul. 2013;

[19] P. Tan, M. Steinbach, V. Kumar, V. "Association Analysis: Basic Concepts and Algorithms," in Introduction to Data Mining. P. Tan, M. Steinbach, V. Kumar (eds). London:Pearson, 2005. pp. 328-414.

[20] C. Basu, M. Padmanaban, S. Guillon, L. Cauchon, M. DeMontigny, I. Kamwa. "Association rule mining to understand GMDs and their effects on power systems," in Proc. IEEE Power and Energy Society General Meeting, 2016, pp.1-6.

[21] S. Feng, L; Yu, Y. Yang. "The relationship between grouped solar flares and sunspot activity," in Proc. Bull. Astr. Soc. India, 2013, pp. 237–246.

[22] L. Shen, J. Dun, X. Zhang, Y. Jiang. " Analysis of the Major Parameters in Solar Active Regions Based on the PCA Method". Chinese Astronomy and Astrophysics. vol. 39. pp 212 – 224. Jul. 2015.

[23] SWPC. Data Access: SWPC Data Service. 2017. Internet: http://www.swpc.noaa.gov/content/data-access. [Sep. 10, 2019].

[24] SRS Solar Region Summary. Internet: ftp://ftp.swpc.noaa.gov/pub/forecasts/SRS/README. [Aug. 10, 2010].

[25] SGAS Solar and Geophysical Activity Summary. Internet: ftp://ftp.swpc.noaa.gov/pub/forecasts/SGAS/README. [Aug. 10, 2019].

[26] X. Yang, G. Le, C. Zhang, Z. Yin, W. Zhao. "A Statistical Analysis of X1- and X2- or Higher-Class Flares during Solar Cycles 21∼23". Chinese Astronomy and Astrophysics. vol. 38, pp. 92 –99. Jun. 2014.

[27] THE R FOUNDATION. The R Project for Statistical Computing. Internet: https://www.r-project.org. [Aug. 12, 2019].

[28] Silveira C.R., Cecatto J.R., Santos M.T.P., Ribeiro M.X. (2018) Thematic Spatiotemporal Association Rules to Track the Evolving of Visual Features and Their Meaning in Satellite Image Time Series. In: Latifi S. (eds) Information Technology - New Generations. Advances in Intelligent Systems and Computing, vol 738. Springer, Cham.

[29] Liu, Chang, Deng, Na, Wang, Haimin, and Wang, Jason T. L. "Predicting Solar Flares Using SDO /HMI Vector Magnetic Data Products and the Random Forest Algorithm". United States: N. p., 2017.

# Shallow vs. Deep Image Representations: A Comparative Study with Enhancements Applied For The Problem of Generic Object Recognition

**Yasser M. Abdullah** *yasware@gmail.com*
*Faculty of Engineering/Department of IT*
*Aden University*
*Aden, Yemen*

**Mussa M. Ahmed** *mussa_m7@yahoo.com*
*Faculty of Engineering/Department of ECE*
*Aden University*
*Aden, Yemen*

**Abstract**

The traditional approach for solving the object recognition problem requires image representations to be first extracted and then fed to a learning model such as an SVM. These representations are handcrafted and heavily engineered by running the object image through a sequence of pipeline steps which requires a good prior knowledge of the problem domain in order to engineer these representations. Moreover, since the classification is done in a separate step, the resultant handcrafted representations are not tuned by the learning model which prevents it from learning complex representations that might would give it more discriminative power. However, in end-to-end deep learning models, image representations along with the classification decision boundary are all learnt directly from the raw data requiring no prior knowledge of the problem domain. These models deeply learn the object image representation hierarchically in multiple layers corresponding to multiple levels of abstraction resulting in representations that are more discriminative and give better results on challenging benchmarks. In contrast to the traditional handcrafted representations, the performance of deep representations improves with the introduction of more data, and more learning layers (more depth) and they perform well on large-scale machine learning problems. The purpose of this study is six fold: (1) review the literature of the pipeline processes used in the previous state-of-the-art codebook model approach for tackling the problem of generic object recognition, (2) Introduce several enhancements in the local feature extraction and normalization steps of the recognition pipeline, (3) compare the enhancements proposed to different encoding methods and contrast them to previous results, (4) experiment with current state-of-the-art deep model architectures used for object recognition, (5) compare between deep representations extracted from the deep learning model and shallow representations handcrafted through the recognition pipeline, and finally, (6) improve the results further by combining multiple different deep learning models into an ensemble and taking the maximum posterior probability.

**Keywords:** Shallow Models, Deep Learning Models, Encoding Methods, Object Recognition, BoVW.

## 1. INTRODUCTION

Generic object recognition problem is simply to assign a label for an object image. The image may contain multiple objects (such as elephants, leopards, sunflowers, grand pianos, faces, etc.) and hence multiple labels should be assigned accordingly. This problem is one of the most fundamental problems in computer vision and pattern recognition and has wide range of applications like web content analysis and video surveillance. However, it is a challenging

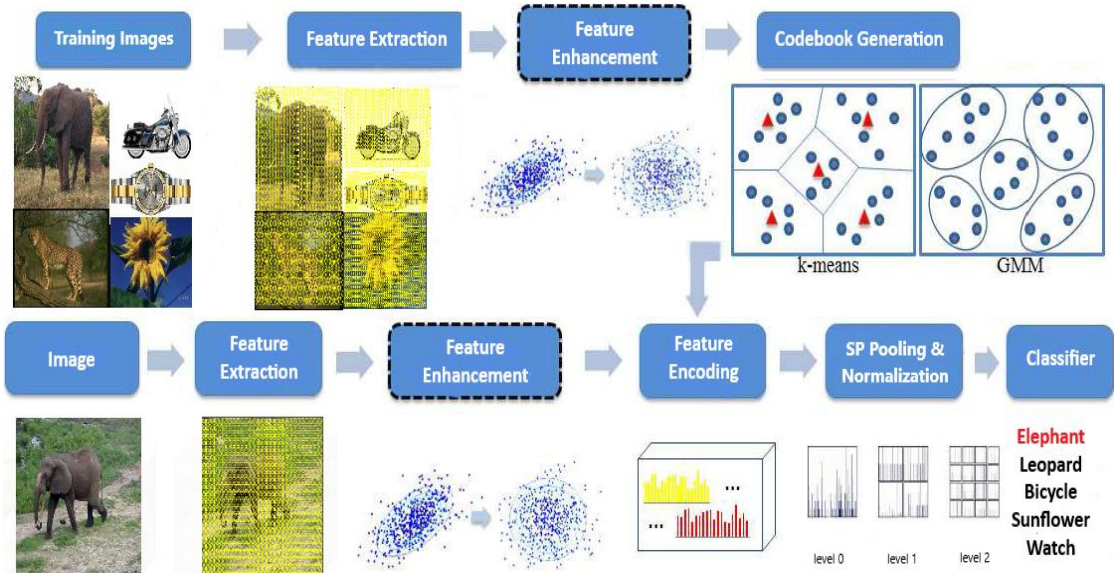



**FIGURE 1:** Bag of Visual Words (BoVW).

problem especially in the presence of intra-class variation, clutter, occlusion, deformation, illumination and viewpoint changes.

The best known shallow framework used for solving this problem is the Bag of Visual Words (BoVW) [1, 2] model which makes use of a pretrained codebook as shown in figure 1. BoVW passes the query image through a pipeline consisting of several steps to get the final representation describing the image. It starts out by extracting local features of the object image. The local features are detected and described to get local feature descriptors. Many feature detection and description methods were proposed in the literature and are reviewed in section 2. In a later step, these features are transferred from their feature space onto the codebook space using an encoding method. For a particular feature descriptor, the encoding methods can produce a single code or a block of codes for each codeword in the codebook. Many encoding algorithms were proposed in the literature and are discussed in section 4. To get a global representation of the image, the coding responses for each codeword is then integrated into a single code (or a block of codes) using a pooling method like sum or max pooling. Pooling process can be improved if performed spatially using a Spatial Pyramid (SP) [3]. At the end of the pipeline, the resultant image representation is normalized using a normalization method like $\ell_2$ or power normalization. At this stage, the image hand-engineered representation is ready and can be fed to a machine learning model (e.g., SVM) to give the class of the query image.

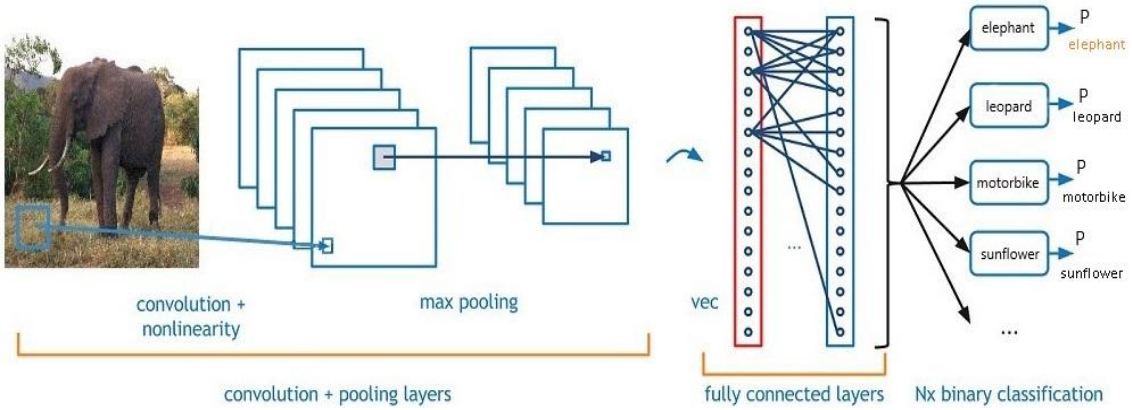



**FIGURE 2:** Deep model architecture (CNN).

To improve the performance of shallow representations, we proposed a new pipeline step in which the previously extracted local features are enhanced (see figure 1). In this step, feature descriptors are square-rooted as suggested by Arandjelović and Zisserman [4] and are reduced in dimensionality and decorrelated using Principal Component Analysis (PCA). Additionally, they are augmented by their spatial location [5]. BoVW model will be discussed in section 3.

On the contrary, a deep learning model doesn't need features to be extracted beforehand but the object image raw pixels can be fed directly to the deep model for both feature extraction and classification in a single step. In this model, features are deeply leant inside the deep model in multiple learning layers and no prior knowledge about the problem domain is required. Deep models are sometimes called generalized feature extractors since they can be used to extract features for different data domains such as text, audio and video. Figure 2 displays a deep model architecture used for object recognition and is called a convnet or Convolutional Neural Network (CNN). CNNs start learning directly from the input image and succeeding layers consume the output of the preceding layer(s). Mainly, they consist of many Learning layers, some pooling layers, and one classification layer. A learning layer can be a convolutional layer or a fully connected layer (FC). Convolutional layers are many in number and can accept and produce multidimensional data whereas FC layers are a few and only accept and produce one-dimensional data. FC layers can exist only at the end of the network whereas convolutional layers are dispersed throughout the network. Inside the learning layers, there exist a nonlinearity to help the network learn interesting functions. They also contain many free parameters that are adjusted during the supervised training process. The classification layer (final layer) computes the loss that the network has incurred in learning non-discriminative representations. The network then updates its free parameters so as to produce more discriminative representations to bring the network loss down. To reduce the dimensionality of the convolutional layer output, a pooling layer is used. In addition to dimensionality reduction, it summarizes local inputs by one statistic that gives the network invariance to simple translations. Deep learning and its models will be discussed in section 5.

In this study, we have conducted many experiments to compare the performance of shallow and deep representations. We evaluated our proposed enhancements to different encoding methods used in the BoVW codebook model and obtained better results than before. Also, we experimented with deep model architectures and compared between different architectures on one hand and between deep and shallow models on another hand. We studied further the discriminative power of deep and shallow features extracted from deep and shallow models respectively. Finally, we improved the results further by combining several deep models into an ensemble and taking the maximum posterior probability. All of experiment details are mentioned in section 6. Conclusion, future work and references are given in sections 7, 8 and 9 respectively.

## 2. LOCAL FEATURES

For any object image, local features can be interest points or interest regions. They are extracted from the appearance of an object and are local in the sense that they describe a local part of the object appearance. An image can produce several hundred (or thousands) of local features. The next subsections review the detection and description methods used to extract and describe image local features.

### 2.1 Feature Detectors

Many feature detectors were proposed in the literature such as Scale-Invariant Feature Transform (SIFT [6]) and Speeded Up Robust Features (SURF [7]). Feature detection is about detecting points (or regions) in an image that are repeatable – i.e., given a different image of the same object, the feature is distinctive enough that we can find it again in the correct location. A lot of methods were used for the detection process. Harris [8] has used the second moment matrix (Harris matrix) which contains image first derivatives in order to detect corner points. He used a corner quality measure that is based on eigenvalues of this matrix (Harris measure). To detect a feature at multiple spatial scales, a Gaussian scale-space for an image is constructed and

features are detected at all scales. To prevent multiple detections for the same feature and select the scale that is most significant for the feature, Lindeberg [9] introduced the concept of automatic scale selection which allows to detect interest points in an image, each with its own characteristic scale. The characteristic scale is the one that gives the maximum of the scale-normalized Laplacian of Gaussian function in scale space. Lindeberg [9] also proposed another detection method that is based on the scale-normalized Hessian matrix of image second derivatives. In this method, an interest point is detected if the trace of this matrix (which is the scale-normalized Laplacian of Gaussian or LoG) is locally maximal in scale space. Lowe [6], in his SIFT detector, approximated the Laplacian of Gaussian (LoG) detector using the Difference of Gaussians (DoG) which is computationally more efficient. Another measure of detection is to use the determinant of the Hessian (DoH) as a feature quality measure. Mikolajczyk and Schmid [10] refined the previous methods, creating robust and scale-invariant feature detectors with high repeatability, which they coined Harris-Laplace and Hessian-Laplace. They used a scale-adapted Harris measure or the determinant of the Hessian matrix to select the location, and the Laplacian to select the scale. Bay et al. [7] noted that the discrete Gaussian filters used in the computation of the scale-normalized Hessian could be approximated by extremely simple box filters involving simple sums and differences of pixels and have used this idea, in their SURF detector, to approximate the determinant of the Hessian. The box filters can be applied very quickly compared to filters with floating-point coefficients. Moreover, if integral images [11] are used for the computation, then the speed of applying the box filter is independent of the filter size resulting in what they called Fast Hessian. Rosten and Drummond [12] proposed a fast detection algorithm for what they called FAST Corners. In their method, a candidate pixel $p$ is compared to a discretized circle of pixels around it; if all the pixels on a contiguous arc of $n$ pixels $(n = 12)$ around the circle are significantly darker or brighter than the candidate pixel, it is detected as a feature. They [13] later extended the FAST idea using a machine learning approach (a decision tree) based on the intensities of the sixteen surrounding pixels of the candidate pixel to yield a detector that is higher in performance and speed. Matas et al. [14] proposed a new type of feature called Maximally Stable Extremal Regions or MSERs. These too are extremely fast to compute and are based on the basic thresholding operation. A region (connected component) is detected as feature if it satisfies two conditions: (1) all the pixels inside that region are either all darker or all brighter than pixels in the boundary, (2) the region should be stable, i.e., it should change little to threshold variations.

It is important to mention that most of the above feature detectors are not robust to large viewpoint changes. If the viewpoint of an object undergoes a large affine transformation, then the region of the same pixels around the interest point in both cases would be different. The fundamental theory of affine-invariant regions was first proposed by Lindeberg and Gårding [15] and applied by other researchers including Baumberg [16], Schaffalitzky and Zisserman [17], and Mikolajczyk and Schmid [18]. An elliptical affine invariant region can be computed by an iterative procedure called affine adaptation. After applying this procedure, the resulting circular region and that region before the transformation are identical but up to a rotation. Mikolajczyk and Schmid [18] proposed to simultaneously detect feature point locations and corresponding affine-invariant regions using an iterative algorithm, resulting in Harris-Affine or Hessian-Affine features according to the type of detector used whether Harris or Hessian.

After determining the feature location and scale, the next problem for the detector is to determine the feature support region - i.e., the set of pixels in the feature neighborhood that should contribute to a feature's descriptor. For scale-invariant features such as Harris-Laplace, Hessian-Laplace, and DoG where features are detected at a characteristic scale, the support region can be a circle drawn with a radius proportional to the feature characteristic scale. For features like MSERs and Hessian-Affine which produce affine-covariant regions, we can use the circular region produced at the end of the affine adaptation process as a support region. However, many description methods assume a square patch is given around the feature location as opposed to a circular one which means that we need to assign a reliable orientation to the feature to define the top edge of the square.

### 2.2 Feature Descriptors

Once a feature's location (and perhaps some additional information such as its scale or support region) has been determined, the next step is to describe the feature with a vector of numbers called a descriptor. Throughout the literature, large number of feature descriptors were proposed. Lowe's SIFT [6] descriptor used a square patch around the feature point as opposed to a circular one which requires a reliable orientation for the square patch to be calculated. He suggested estimating this orientation based on a histogram of pixel gradient orientations over the support region of a scale-invariant circle and taking the dominant gradient orientation as the patch orientation. The square patch is then rotated upwards to normalize its direction, resampled, and smoothed by a Gaussian. The gradient at each resampled pixel is estimated and weighted by a Gaussian. The square is then subdivided into $4 \times 4$ smaller sub-squares with one histogram of eight direction bins in each sub-square. These sixteen histograms are then concatenated to form a $4 \times 4 \times 8 = 128$-dimensional vector which is normalized twice to make it robust to illumination changes. Mikolajczyk and Schmid [19] proposed a SIFT variant called GLOH (Gradient Location and Orientation Histogram). Instead of the square grid, a log polar grid of three rings is created. The outer and second outer rings are subdivided into eight bin locations each whereas the center ring is undivided to give a total of seventeen bin locations. In each bin location, a histogram of sixteen quantized directions is computed. Histograms from different bin locations are concatenated to form a raw $16 \times 17 = 272$-dimensional descriptor. The dimensionality of the descriptor is then reduced using PCA to give a 128-dimensional vector. SURF descriptor is very efficient to compute because it uses Haar wavelets which are simple box filters. As in SIFT, the oriented square at a feature's detected scale is split into a $4 \times 4$ square grid. However, instead of computing gradient orientation histograms in each subsquare, Haar wavelet responses at twenty-five points in each subsquare are computed. The sums of the original and absolute responses in the $x$ and $y$ directions are computed in each sub-square, yielding a $4 \times 4 \times 4 = 64$-dimensional descriptor. Ke and Sukthankar [20] addressed the problem of dimensionality reduction of the SIFT descriptor using PCA in what they coined PCA-SIFT. They collected a large number of DoG keypoints and constructed $41 \times 41$ patches at the estimated scale and orientation of each keypoint. The $x$ and $y$ gradients at the interior pixels of each patch were collected into a $39 \times 39 \times 2 = 3042$-dimensional vector, and PCA was applied to determine a much smaller number of basis vectors (e.g., twenty or thirty-six). Thus, the high-dimensional vector of gradients for a candidate feature is represented by a low-dimensional descriptor given by its projection onto the learned basis vectors. Belongie et al. [21] proposed shape contexts as a method for matching shapes, which were modified by Mikolajczyk and Schmid [19] for feature point description. The approach is similar to GLOH in that a log-polar location grid is constructed at the feature point location. However, instead of using the gradients of all points to construct the histograms in each subregion, only edge points detected with the Canny detector [22] are allowed to contribute their gradient orientations. Each edge point's contribution is further weighted by its gradient magnitude. Johnson and Hebert [23] originally proposed spin images for describing features in range data. Lazebnik et al. [24] proposed modifying them to create feature descriptors for grayscale images. We simply compute a histogram of quantized intensities for each of several rings around the feature location, after the intensities have been normalized. The dimension of the descriptor is the number of intensity bins times the number of rings. Since there are no angular subdivisions of the rings, the descriptor is rotation-invariant.

Another class of descriptors that do not require the patch orientation estimation and explicit orientation, are called Invariant-based descriptors. They are based on invariant functions of the patch pixels with respect to a class of geometric transformations, typically rotations or affine transformations. Schmid and Mohr [25] popularized the idea of differential invariants for constructing rotation invariant descriptors. That is, the descriptor is constructed using combinations of increasingly higher-order derivatives of the Gaussian smoothed image. Moment-invariant [26] descriptors are another type of invariant descriptors that are computed using the image intensities and spatial locations.

## 3. BAG OF VISUAL WORDS MODEL

The first decade of this century has seen the rise of Bag of Visual Words (BoVW) (or Bag of Features (BoF)) in computer vision and has been applied to many problems such as object recognition and image retrieval. BoVW model was developed from the Bag of Words model used in document classification and is probably the most popular and effective shallow framework for object classification. In this model, an object image is passed through a pipeline consisting of typically five steps to give at the end the image final handcrafted representation (see figure 1). The five pipeline steps are (1) local feature extraction, (2) local feature enhancement, (3) feature descriptor encoding, (4) code pooling and finally, (5) normalization. In the first step, patches from an image are extracted and represented using feature descriptors such as SIFT. The extraction process employs different sampling strategies which can be dense or sparse. In dense sampling, a patch is extracted and represented at every $n$ pixels on a fine fixed grid while in sparse sampling, patches are extracted only at interest points or regions detected by a feature detector. Multiscale features can also be extracted by using different patch sizes. It is worth noting that the sets of local descriptors produced for different images may have different cardinality especially if sparse sampling is used or if images have different resolutions.

The second step of the pipeline is optional, but we have seen much improvement in the final results when employed. To enhance the features, we first square-root their descriptors. After that, we reduce their dimensionality by learning from the training data a set of basis vectors using PCA and then projecting the features onto a subset of these basis vectors that has the highest projection variance. Feature descriptors are then decorrelated by whitening their descriptor dimensions. Moreover, the feature spatial location information is embedded inside its descriptor [5]. Let $\mathbf{x}_n$ be a feature descriptor, then the feature is augmented by its normalized spatial location: $[\mathbf{x}_n^{\mathrm{T}}, \frac{x}{h} - 0.5, \frac{y}{w} - 0.5]^{\mathrm{T}}$ where $(x, y)$ is the descriptor $\mathbf{x}_n$ spatial location, and $h \times w$ are the image dimensions.

In the third step of the pipeline, the set of local features extracted in previous steps are encoded using an encoding process which makes use of a previously generated codebook (also called dictionary or vocabulary). The codebook is generated in an offline process using an unsupervised learning method such as $k$-means clustering [27, 28] or Gaussian Mixture Model (GMM, [29]). Local feature descriptors from all training images are clustered in feature space into $k$ clusters to form the codewords of the codebook. Hence, the inputs to the encoding process are the image local feature descriptors and the previously learnt codebook. The encoding process in turn produces a coding matrix per image, one coding vector for each local descriptor, by using one of different encoding methods. Feature coding is a key component of object recognition and has been widely studied in the past several years. Various encoding methods were proposed and they differ in how they activate the codewords for a given local feature. These algorithms encode a feature descriptor by producing a response for each codeword in the dictionary. The coding response can have different dimensionality for different encoding methods. Encoding methods will be discussed in section 4.

The fourth step of the pipeline (pooling process) converts the image coding matrix into a vector (called the pooling vector) which forms the image global representation. For each codeword, codes from multiple local features are integrated dimension-wise into a single code (or a block of codes) using one of the classic pooling methods. Common examples include sum pooling and max pooling. Let $C_n^m$ be the response of descriptor $x_n$ for codeword $m$, and $h^m$ be the pooling response for codeword $m$. In sum pooling, responses corresponding to the same codeword are summed for all descriptors, i.e., $h^m = \sum_n c_n^m$, whereas in max pooling, the maximum of the codes is taken, i.e., $h^m = \max_n c_n^m$.

In the last pipeline step, the image raw representation is normalized to form the image final representation. Different normalization methods exist such as $l_1$-normalization, $l_2$-normalization, power normalization, and intra-normalization [30]. In $l_1$-and $l_2$-normalization, the representation dimensions are divided by the corresponding $l_1$-, or $l_2$- norm respectively. In power normalization,

| **Encoding Algorithm** | HA | SA | LSA | FV | VLAD | SPC | LCC | LLC | App. LLC | SC | GSC | LTC | SVC |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\mathbf{c}_n^m$ **dimensionality** $(R)$ | 1 | 1 | 1 | $2D$ | $D$ | 1 | 1 | 1 | 1 | 1 | 1 | $H+1$ | $D+1$ |
| **No. of codewords activated** | 1 | $M$ | $K$ | $M$ | 1 | $*$ | $*$ | $M$ | $K$ | 1 | $K$ | $*$ | 1 |

**TABLE 1:** Coding response for different encoding methods. Top row: Coding Response dimensionality per codeword. Bottom row: number of codewords activated by a single feature descriptor. Star symbol ($*$) indicates a sparse set of codewords activated per feature descriptor and might be variable.

we apply to each dimension of the representation the following function: $sign(\boldsymbol{p}(i))|\boldsymbol{p}(i)|^{\alpha}$ , where $\boldsymbol{p}(i)$ is the $i^{\text{th}}$ dimension of the representation, and $0 \leq \alpha \leq 1$. Sign-square-root normalization is a special case of power normalization when $\alpha = 0.5$. Intra-normalization carries out normalization in a block by block manner and can be done using $l_1$- or $l_2$- normalization. Intra-normalization can be applied when the coding process produces a block of codes per codeword. We proposed different normalization strategies for different encoding methods which are based on $l_2$- and sign-square-root normalization and are detailed in section 6.1.

### 3.1 Spatial information

Generally, the BoVW model is orderless and hence discards all the information about the location of the patches and this in turn incurs a loss of information. The dominant approach to include spatial information is to use the Spatial Pyramid (SP). Inspired by the pyramid match kernel of Grauman
and Darrell [31], Lazebnik et al. [3] proposed to partition an image into a set of regions in a course-to-fine manner. They propose to partition the image into $1 \times 1$, $2 \times 2$, and $4 \times 4$, for a total of $21$ regions. Each region is described independently and the region-level pooling vectors are then concatenated into an image-level pooling vector. Marszalek et al. [32] suggested a different partitioning strategy in which the image is partitioned into $1 \times 1$, $1 \times 3$, and $2 \times 2$, for a total of $8$ regions.

### 3.2 Classification

After running the image through the recognition pipeline, the image final representation is ready for classification. Many machine learning models were tried throughout the literature but the most popular one is the SVM which gives better results when used in conjunction with BoVW model. The SVM searches the space of linear decision boundaries to find the one that gives the maximum margin between two binary classes. It optimizes a constrained convex quadratic problem that uses the dot product between input features (linear kernel) and hence the kernel trick can be used. The nonlinear kernel function transfers the input features into a higher dimensional space and computes the dot product there in that space. Since the mapping process is done implicitly, we benefit from the increase in dimensionality by noting that problems become more likely to be linearly separable in high dimensional spaces. Moreover, the dot product is computed in the mapped space by computing the kernel function in the original space.

Many non-linear kernels for BoVW model were proposed and used throughout the literature such as the histogram intersection kernel, pyramid match kernel, chi-square kernel, and Hellinger's kernel. Despite their better classification results, they are less efficient than the linear kernel. There exist a class of kernels (additive homogeneous kernels [33]) that are as efficient as linear ones up to the computation of an efficient explicit feature map. Several non-linear kernels can be approximated by explicitly computing their feature maps.

## 4. ENCODING METHODS

Let $\mathbf{X}$ be a set of $D$-dimensional local descriptors extracted from an image, i.e., $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_N] \in \mathbb{R}^{D\times N}$, $\mathbf{B} = [\mathbf{b}_1, \dots, \mathbf{b}_M] \in \mathbb{R}^{D\times M}$ be the codebook of $M$ codewords and $\mathbf{C}_n = [(\mathbf{c}_n^1)^{\mathrm{T}}, \dots, (\mathbf{c}_n^M)^{\mathrm{T}}] \in \mathbb{R}^{RM}$ be the coding vector produced by the encoding method corresponding to feature descriptor $\mathbf{x}_n$, where $\mathbf{c}_n^m$ is the coding response of the $m^{\mathrm{th}}$ codeword and $R$ is the coding response dimensionality which can be a scalar or a block of codes. Each local descriptor $\mathbf{x}_n$ may activate one or more codewords during the encoding process. The dimensionality of the coding response per codeword and the number of activated codewords for a feature descriptor are different for different encoding methods and are given in table 1.

### 4.1 Hard Assignment (HA)

This coding method [1] forms the baseline encoding upon which other encoding methods improve. It is also called by other names like Vector Quantization (VQ) or histogram encoding. HA encodes a feature descriptor $\mathbf{x}_n$ by assigning the whole coding weight to the nearest codeword. HA is defined as:

$$c_n^m = \begin{cases} 1, & if\ m = \underset{i}{argmin}(\|\mathbf{x}_n - \mathbf{b}_i\|_2) \\ 0, & otherwise \end{cases}, i = 1, \dots, M \tag{1}$$

### 4.2 Soft Assignment (SA)

Instead of assigning all the coding weight to the closest codeword, this method [34-36] distributes the coding weight among all the codewords in a soft manner proportional to how far each codeword is from the feature descriptor. It achieves this by making use of a distance function like the Gaussian kernel. In contrast to HA, SA takes into account the uncertainty of codewords in case two or more codewords are strong candidates for a given feature descriptor. SA is defined as:

$$c_n^m = \frac{\exp(-\beta\|\mathbf{x}_n - \mathbf{b}_m\|_2^2)}{\sum_{i=1}^{M}\exp(-\beta\|\mathbf{x}_n - \mathbf{b}_i\|_2^2)}, \tag{2}$$

where $\beta$ is a smoothing factor controlling the softness of the assignment.

### 4.3 Localized Soft Assignment (LSA)

Unlike SA, this method [37] only considers the $K$ closest codewords in the neighborhood of the feature descriptor. Let $\mathbf{B}_n = [\,\hat{\mathbf{b}}_k]_{k=1}^K$, be the $K$ closest codewords to the feature descriptor $\mathbf{x}_n$. LSA is defined as:

$$c_n^m = \begin{cases} \psi(\mathbf{x}_n), & if\ \mathbf{b}_m \in \mathbf{B}_n \\ 0, & otherwise, \end{cases} \tag{3}$$

$$\psi(\mathbf{x}_n) = \frac{\exp\left(-\beta\|\mathbf{x}_n - \mathbf{b}_m\|_2^2\right)}{\sum_{k=1}^{K}\exp\left(-\beta\left\|\mathbf{x}_n - \hat{\mathbf{b}}_k\right\|_2^2\right)},$$

### 4.4 Fisher Vector

Fisher coding [38] is inspired by the Fisher kernel which describes a signal with a gradient vector derived from its generative probability density function [39]. In the context of object recognition, the signal is the image, the gradient vector is used for feature coding, and the probability density function is a Gaussian Mixture Model (GMM). A GMM of $M$ components with parameters $\boldsymbol{\theta} = \{\,\boldsymbol{\theta}_m\}_{m=1}^M$, is given by:

$$p(\mathbf{x}_n|\boldsymbol{\theta}) = \sum_{m=1}^{M} w_m u_m(\mathbf{x}_n|\boldsymbol{\theta}_m), \tag{4}$$

where $u_m(\mathbf{x}_n|\,\boldsymbol{\theta}_m)$ denotes Gaussian □:

$$u_m(\mathbf{x}_n|\,\boldsymbol{\theta}_m) = \frac{exp\left\{-\frac{1}{2}(\mathbf{x}_n - \boldsymbol{\mu}_m)^{\mathrm{T}}\boldsymbol{\Sigma}_m^{-1}(\mathbf{x}_n - \boldsymbol{\mu}_m)\right\}}{(2\pi)^{D/2}|\boldsymbol{\Sigma}_m|^{1/2}}, \tag{5}$$

and $\{w_m, \boldsymbol{\mu}_{\mathrm{m}}, \boldsymbol{\Sigma}_{\mathrm{m}}\}$ denote the weight, the mean vector, and the covariance matrix of Gaussian $m$ respectively and can be estimated using Expectation Maximization (EM) algorithm. We require $\forall_m$: $w_m \geq 0$ and $\sum_{m=1}^{M} w_m = 1$ to ensure $p(\mathbf{x}_{\mathrm{n}}|\,\boldsymbol{\theta})$ is a valid distribution. We assume a diagonal

covariance matrix so $\boldsymbol{\Sigma}_m$ is reduced to a variance vector denoted by $\boldsymbol{\sigma}_m^2$. Supposing all the features are independent from each other, an image can be expressed as the log-likelihood of all the features:

$$l(\mathbf{X}\,|\boldsymbol{\theta}) = \sum_{n=1}^{N} log\ p(\mathbf{x}_n|\boldsymbol{\theta}). \tag{6}$$

We can define Fisher vector as the gradient of $l$ with respect to $\boldsymbol{\theta}$ normalized by the square root of the inverse of Fisher information matrix $\mathbf{F}_{\boldsymbol{\theta}}$, i.e.,

$$\boldsymbol{\mathcal{g}} = \mathbf{F}_{\boldsymbol{\theta}}^{-1/2}\mathbf{g}, \tag{7}$$

where $\mathbf{g} = \{\partial l/\partial \boldsymbol{\mu}_m, \partial l/\partial \boldsymbol{\sigma}_m\}_{m=1}^{M}$. The gradient with respect to the weight parameter was omitted since it adds a little information [38]. Note that the Fisher information matrix $\mathbf{F}_{\boldsymbol{\theta}}$ has an approximated closed-form solution, and $\mathbf{F}_{\boldsymbol{\theta}}^{-1/2}$ normalization corresponds to the whitening of the dimensions. FV encoding is defined as:

$$\mathbf{c}_n^m = \left[\boldsymbol{\mathcal{g}}_{\mu_m}^{\mathrm{T}}, \boldsymbol{\mathcal{g}}_{\sigma_m}^{\mathrm{T}}\right], \tag{8}$$

$$\boldsymbol{\mathcal{g}}_{\mu_m}^{\mathrm{T}} = \frac{1}{\sqrt{w_m}}\gamma_n(m)\left(\frac{\mathbf{x}_n - \boldsymbol{\mu}_m}{\boldsymbol{\sigma}_m}\right),$$

$$\boldsymbol{\mathcal{g}}_{\sigma_m}^{\mathrm{T}} = \frac{1}{\sqrt{2w_m}}\gamma_n(m)\left(\frac{(\mathbf{x}_n - \boldsymbol{\mu}_m)^2}{\boldsymbol{\sigma}_m^2} - 1\right),$$

$$\gamma_m(m) = \frac{w_m u_m(\mathbf{x}_n|\,\boldsymbol{\theta}_m)}{\sum_{i=1}^{M} w_i u_i(\mathbf{x}_n|\,\boldsymbol{\theta}_i)}.$$

### 4.5 Vector of Locally Aggregated Descriptors (VLAD)

In FV, each codeword is represented by its first and second order statistics. In contrast, VLAD [30, 40] can be viewed as a hard version of FV and it only keeps the first order statistics. It encodes a feature descriptor $\mathbf{x}_n$ by the vector difference to its closest codeword. VLAD is defined as:

$$\mathbf{c}_n^m = \begin{cases} (\mathbf{x}_n - \mathbf{b}_m), & if\ m = \operatorname{argmin}_i(\|\mathbf{x}_n - \mathbf{b}_i\|_2) \\ 0, & otherwise \end{cases}_{,\, i = 1, \ldots, M} \tag{9}$$

### 4.6 Sparse Coding (SPC)

SPC [41] finds a sparse set of codewords with corresponding coefficients $\mathbf{r}_n$ that can reconstruct the input feature descriptor $\mathbf{x}_n$. SPC solves an $l_1$ regularized least-square optimization problem defined as:

$$\arg\min_{\mathbf{r}_n} \|\mathbf{x}_n - \mathbf{B}\mathbf{r}_n\|_2^2 + \lambda\|\mathbf{r}_n\|_1 \tag{10}$$
$$\text{s.t } \mathbf{1}^{\mathrm{T}}\mathbf{r}_n = 1.$$

Sparsity is attained in $\mathbf{r}_{\mathrm{n}}$ due to the $l_1$ norm which is known to induce sparsity. SPC is defined as:

$$c_n^m = \mathbf{r}_n(m), \tag{11}$$

where $\mathbf{r}_n(m)$ is the reconstruction coefficient of descriptor $\mathbf{x}_n$ for the codeword $\mathbf{b}_m$.

### 4.7 Local Coordinate Coding (LCC)

In practice, SPC tends to be local, i.e., it reconstructs a feature descriptor $\mathbf{x}_n$ using codewords in its neighborhood. But this locality cannot be ensured theoretically. Yu et al. [42] suggested a modification that explicitly encourages the coding to be local. They pointed out that under certain assumptions locality is more important than sparsity for successful nonlinear function learning using the obtained codes. LCC solves the following $l_1$ regularized least-square optimization problem:

$$\arg\min_{\mathbf{r}_n} \|\mathbf{x}_n - \mathbf{B}\mathbf{r}_n\|_2^2 + \lambda \sum_{m=1}^{M} |\mathbf{r}_n(m)|\, \|\mathbf{x}_n - \mathbf{b}_m\|_2^2 \tag{12}$$

$$\text{s.t } \mathbf{1}^{\mathrm{T}}\mathbf{r}_n = 1,$$

and LCC is defined as:

$$c_n^m = \mathbf{r}_n(m), \tag{13}$$

where $\mathbf{r}_n(m)$ is the reconstruction coefficient of feature descriptor $\mathbf{x}_n$ for the codeword $\mathbf{b}_m$.

### 4.8 Locality-Constrained Linear Coding (LLC)

The computational cost of LCC and SPC is high because their solution relies on iterative optimization. To address this problem, LLC [43] changed the term $|\mathbf{r}_n(m)|$ in (12) into a differentiable term $\mathbf{r}_n(m)^2$ (as in (14)) to yield an optimization problem that has a closed-form solution. LLC optimizes the following $l_2$ regularized problem:

$$\arg\min_{\mathbf{r}_n} \|\mathbf{x}_n - \mathbf{B}\mathbf{r}_n\|_2^2 + \lambda \sum_{m=1}^{M} (\mathbf{r}_n(m) \exp(\|\mathbf{x}_n - \mathbf{b}_m\|_2)/\sigma)^2 \tag{14}$$

$$\text{s.t } \mathbf{1}^{\mathrm{T}}\mathbf{r}_n = 1,$$

where $\sigma$ is used for adjusting the weighted decay speed for the locality function (exponential term). The coding vector $\mathbf{r}_n$ corresponding to the feature descriptor $\mathbf{x}_n$ is given by:

$$\mathbf{r}_n = \tilde{\mathbf{r}}_n / \mathbf{1}^{\mathrm{T}}\tilde{\mathbf{r}}_n, \tag{15}$$

$$\tilde{\mathbf{r}}_n = (\mathbf{D}_n + \lambda \mathbf{S}_n) \setminus \mathbf{1},$$

$$\mathbf{D}_n = (\mathbf{B} - \mathbf{x}_n\mathbf{1}^{\mathrm{T}})(\mathbf{B} - \mathbf{x}_n\mathbf{1}^{\mathrm{T}})^{\mathrm{T}}$$

$$\mathbf{S}_n = daig(\mathbf{1} \odot trace(\mathbf{D}_n)),$$

where $\odot$ denotes element-wise multiplication. LLC is defined as:

$$c_n^m = \mathbf{r}_n(m) \tag{16}$$

To further enhance the encoding speed, approximated LLC was proposed. The second term in the above optimization problem (14) was omitted, and a feature is reconstructed just from the closest $K$ codewords. Let $\mathbf{B}_n = [\,\hat{\mathbf{b}}_k]_{k=1}^K$ be the closest codewords to the feature descriptor $\mathbf{x}_n$ and $\hat{\mathbf{r}}_n$ be their corresponding coefficient codes, then the approximated LLC reduces the above optimization problem to:

$$\arg\min_{\hat{\mathbf{r}}_n} \|\mathbf{x}_n - \mathbf{B}_n\hat{\mathbf{r}}_n\|_2^2 \tag{17}$$

$$\text{s.t } \mathbf{1}^{\mathrm{T}}\hat{\mathbf{r}}_n = 1,$$

Approximated LLC is defined as:

$$c_n^m = \begin{cases} \hat{\mathbf{r}}_n(k), & if\ \mathbf{b}_m = \hat{\mathbf{b}}_{\boldsymbol{k}} \\ 0, & otherwise \end{cases}, k = 1, \dots, K \tag{18}$$

where $\hat{\mathbf{r}}_n(k)$ is the reconstruction coefficient of the feature descriptor $\mathbf{x}_n$ for the codeword $\hat{\mathbf{b}}_{\boldsymbol{k}}$.

### 4.9 Salient Coding (SC)

SC considers a representation for each codeword that is salient when combined with max pooling. The idea behind SC [44] is that a codeword should receive a strong response if it is much similar with a feature than other codewords. That is, if a codeword is much closer to a feature belonging to this codeword than the other $K$ closest codewords, then the response on this codeword is strong. This response is likely to be preserved during max pooling. As a result, the codeword can independently describe this feature without the help of other codewords. SC is defined as:

$$c_n^m = \begin{cases} \psi(\mathbf{x}_n), & if\ m = \mathrm{argmin}_i(\|\mathbf{x}_n - \mathbf{b}_i\|_2) \\ 0, & otherwise \end{cases}, i = 1, \dots, M \tag{19}$$

$$\psi(\mathbf{x}_n) = \sum_{k=2}^{K} \left(\left\|\mathbf{x}_n - \hat{\mathbf{b}}_k\right\|_2 - \left\|\mathbf{x}_n - \hat{\mathbf{b}}_1\right\|_2\right) / \left\|\mathbf{x}_n - \hat{\mathbf{b}}_k\right\|_2,$$

where $\psi(\mathbf{x}_n)$ denotes the saliency degree, and $[\hat{\mathbf{b}}_k]_{k=1}^{K}$ are the $K$ closest codewords to the descriptor $\mathbf{x}_n$.

### 4.10 Group Salient Coding (GSC)

In SC, weak responses to a codeword are suppressed by the response of the feature that is closest to this codeword compared to others. This results in some features not represented. To alleviate the suppression effect, GSC [45] proposed an idea that is based on group coding. Different group sizes can be formed with a feature $\mathbf{x}_n$. In a group of size $k$, we only consider the feature $\mathbf{x}_n$ and the $k$ nearest codewords. A saliency response for $\mathbf{x}_n$ is calculated for each group size and the response is shared among all the codewords in the group. Each feature $\mathbf{x}_n$ will have multiple coding vectors, one coding vector for each group size. The final coding vector is obtained by taking the maximum of the responses for each codeword over all group sizes. GSC is defined as:

$$c_n^m = \begin{cases} \max\limits_k \{\mathbf{v}_m^k\}, & if\ \mathbf{b_m} \in g(\mathbf{x}_n, k) \\ 0, & otherwise \end{cases}_{\ k=1,\dots,K} \tag{20}$$

$$\mathbf{v}_m^k = \begin{cases} \psi(\mathbf{x}_n), & if\ \mathbf{b}_m \in g(\mathbf{x}_n, k) \\ 0, & otherwise, \end{cases}$$

$$\psi(\mathbf{x}_n) = \sum_{j=1}^{K+1-k} (\left\|\mathbf{x}_n - \hat{\mathbf{b}}_{k+j}\right\|_2 - \left\|\mathbf{x}_n - \hat{\mathbf{b}}_k\right\|_2),$$

where $g(\mathbf{x}_n, k)$ denotes the set of the $k$ nearest codewords to $\mathbf{x}_n$, and $K$ is the maximum group size.

### 4.11 Local Tangent Coding (LTC)

LTC [46] assumes that the codewords and feature descriptors are embedded in a smooth manifold. The main components of LTC are manifold approximation and intrinsic dimensionality estimation. Under the Lipschitz smooth condition, the nonlinear function $f(\mathbf{x}_n)$ can be approximated by a local linear function as:

$$f(\mathbf{x}_n) \approx \sum_{m=1}^{M} (\mu_m^n f(\mathbf{b}_m) + 0.5\mu_m^n \nabla f(\mathbf{b}_m)^{\mathrm{T}} (\mathbf{x}_n - \mathbf{b}_m)), \tag{21}$$

where $\mu_m^n$ is a scalar coefficient associated with codeword $\mathbf{b}_m$ that is used for representing the feature $\mathbf{x}_n$, and is obtained by LCC. The above approximate function (21) can be viewed as a linear function of a coding vector $[\mu_m^n, \mu_m^n (\mathbf{x}_n - \mathbf{b}_m)^{\mathrm{T}}]_{m=1}^{M} \in \mathbb{R}^{M(D+1)}$. LTC argues that there is a lower intrinsic dimensionality in the feature manifold. To get it, PCA is applied to the high dimensional term $\mu_m^n (\mathbf{x}_n - \mathbf{b}_m)$ using a projection matrix $\mathbf{U}_m = [\mathbf{u}_1, \dots, \mathbf{u}_H] \in \mathbb{R}^{D \times H}$ trained from the training data $(H < D)$. Theses directions correspond to the local tangent directions of the manifold. Therefore, the final coding vector for LTC is defined as:

$$\mathbf{c}_n^m = [\alpha \mu_m^n, \mu_m^n (\mathbf{x}_n - \mathbf{b}_m)^{\mathrm{T}} \mathbf{U}_m], \tag{22}$$

where $\alpha$ is a positive scaling factor to balance the two types of codes.

### 4.12 Super Vector Coding (SVC)

SVC [47] is a simple version of LTC. Unlike LTC, SVC uses just the closest codeword $\mathbf{b}_*$ to represent $\mathbf{x}_n$ and obtains $\mu_*^n$ via HA, i.e., $\mu_*^n = 1$. Moreover, SVC does not apply PCA to the term $\mu_m^n (\mathbf{x}_n - \mathbf{b}_m)$ in (21). In SVC, equation (21) simplifies to:

$$f(\mathbf{x}_n) \approx \left(f(\mathbf{b}_*) + 0.5\nabla f(\mathbf{b}_*)^{\mathrm{T}} (\mathbf{x}_n - \mathbf{b}_*)\right), \tag{23}$$

consequently, the coding vector simplifies to:

$$\mathbf{c}_n^m = \begin{cases} [\alpha, (\mathbf{x}_n - \mathbf{b}_m)^{\mathrm{T}}], & if\ m = \mathrm{argmin}_i (\|\mathbf{x}_n - \mathbf{b}_i\|_2) \\ 0, & otherwise \end{cases}_{\ i=1,\dots,M} \tag{24}$$

# 5. DEEP LEARNING

## 5.1 Review

Deep Learning is about learning a task in a hierarchy of layers. The first layer learns from the input and succeeding layers learn from previous ones. Multilayer Perceptron (MLP), proposed in the past, is an architecture that can be built to go deeper but due to many problems it did not do. First of all, backpropagation [48] does not work well for deep MLPs. This is mainly due to the type of nonlinearities used. The traditional sigmoid and hyperbolic tangent activation functions kill the flowing gradients when neurons are saturated and hence layers down in the hierarchy do not receive theses gradients in order to update their weights. Moreover, proper initialization of weights is critical for backpropagation to work well. Improper initialization might cause most neurons down in the hierarchy to output zero as their activation which too kills the gradients from flowing. Second, the introduction of the Universal Approximation theorem [49] – which states that a single hidden layer can be used to approximate any continuous function to any desired precision – diverted people from going deeper with this model. Third, deep models are data hungry, and need much labeled data to train which was very difficult to get at that time. Lastly, deep models need much computation to train because most of the training is matrix multiplication and CPUs were not optimized for this operation.

Recently, the introduction of new activation functions [50-52], better initialization strategies [51, 53, 54], big data like ImageNet [55], better optimization methods like Adam optimization [56], better regularization methods [57-59], and finally the introduction of GPUs, all made deep model training possible.

Deep learning has changed the view of how problems are solved. In end-to-end deep learning, the input to the model is the signal itself, not features extracted from that signal. This produces a unified framework for the two steps that were considered separate for a long time: feature extraction and classification. Learning the features along with the classification boundary in one step produces more discriminative features far better than hand engineered ones and gives much better classification results.

## 5.2 Convolutional Neural Networks (CNNs)

A Convolutional Neural Network (CNN) is a powerful machine learning technique from the field of deep learning. CNNs were inspired by the experiments of Hubel and Weisel [60-62] in which they discovered that different neurons in the cat visual cortex fire for different light edge orientations. They also showed that locality is preserved in processing, i.e., nearby cells in the cortex represent nearby regions in visual field. They hypothesized through their experiments that the visual cortex has a kind of hierarchical organization where simple cells feed to other complex cells which in turn feed to hypercomplex cells. Fukushima [63] introduced the neurocognitron which models the visual cortex in computers. It is composed of a hierarchy of layers where each neuron in upstream layers looks at a small region of input in the downstream layer. Since backpropagation was not invented yet, Fukushima trained his model with an unsupervised procedure. LeCun et al. [64] built on Fukushima's work and trained his network (LeNet-5) with backpropagation. Many years later, Hinton and Salakhutdinov [65] described an effective way of initializing the weights that allows deep autoencoder networks to learn a low-dimensional representation of data. Krizhevsky et al. [66] designed a convolutional neural network that they called AlexNet. It is very similar to LeNet except that it is bigger, deeper, uses Rectified Linear Unit (ReLU) activation function, and is trained on GPUs with more training data (ImageNet). This network has won the ILSVRC2012 competition and reduced the error rate by a large margin. Since then, these models have drawn people attention and many deeper and wider CNNs were proposed. Common CNN examples include VGG [67], GoogleNet [68], ResNet [69, 70], DenseNet [71], etc. These networks have millions of parameters and need proper training.

CNNs are a special kind of multi-layer neural networks which have two fundamental differences: (1) neurons share weights, where many neurons in a layer share the same weights (2) they have sparse connections, i.e., not every neuron in upstream layer is connected to every other neuron in downstream layer. CNNs are composed of many layers of different types like convolutional

layers, pooling layers, FC layers, softmax and classification layers. A convolutional layer is where downstream neurons are convolved (dot product operation) with a filter bank to give rise to many output feature maps, one map for each filter. Convolution filters need not to be flipped and this operation is similar to correlation. After filtering, a non-linearity such as ReLU is applied on the filtered input. It thresholds negative activations and passes positive activations as they are. A pooling layer down-samples each feature map by computing some statistic over it. Common statistics include average and the popular max pooling. Other pooling methods were recently proposed like fractional max-pooling [72] and spatial pyramid pooling [73]. At the end of the network, there exists the FC layers which are like the traditional MLP and it runs for a few layers before the softmax layer. The softmax layer uses the softmax function to convert class scores into class probabilities. The classification layer takes the class probabilities to compute the cross-entropy loss function to be optimized. Other types of layers that prevents the network from overfitting the training data were proposed like batch normalization [74] layer (which also speeds up the training process) and dropout layer [59].

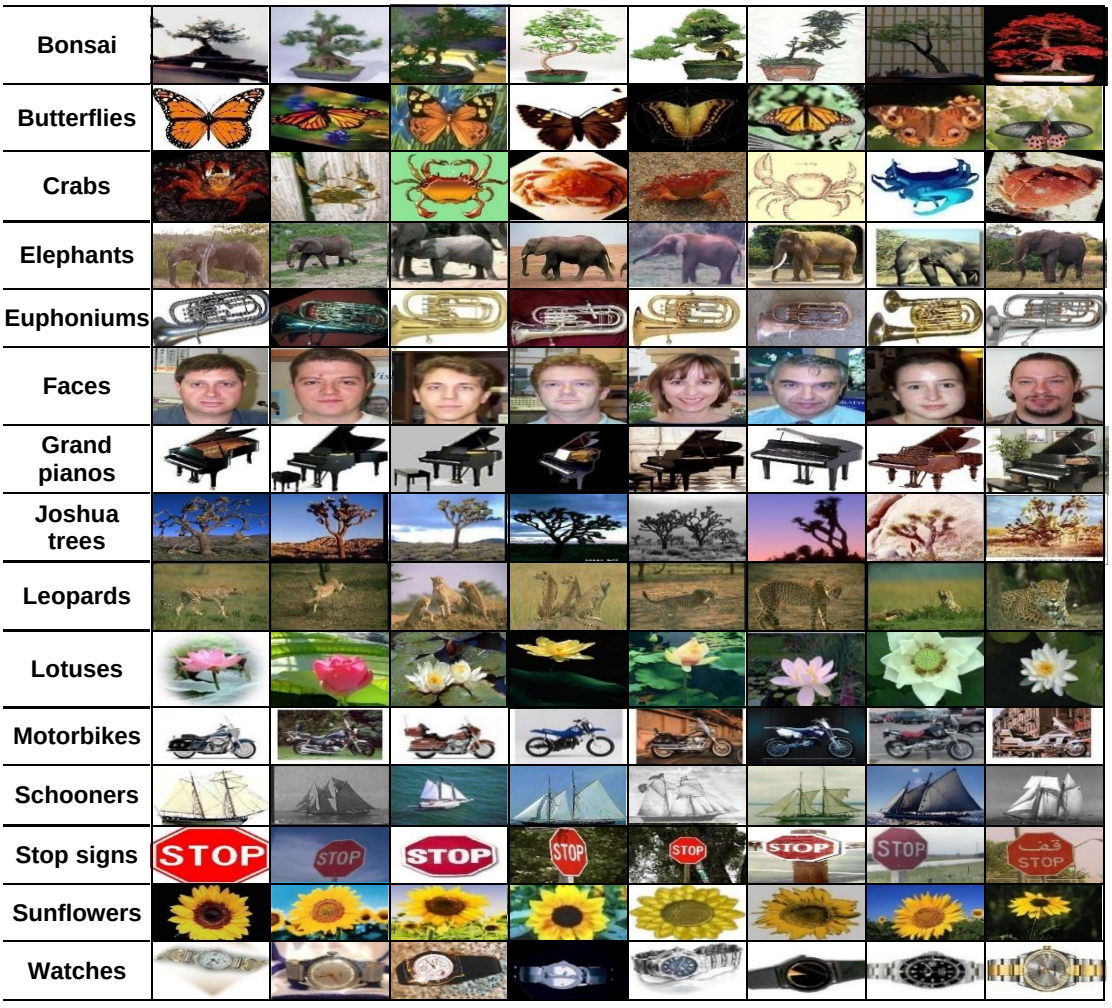



**FIGURE 3:** Example Images from The Image Dataset.

## 6. EXPERIMENTS

In this section, we describe four types of experiments carried out to evaluate the performance of shallow and deep image representations. The first experiment compares the performance of shallow image representations taking into consideration the enhancements proposed in the feature enhancements and normalization pipeline steps of the BoVW model for different encoding methods. Moreover, we contrast the results obtained to previous results obtained without using the proposed enhancements. The second experiment compares the performance of deep image representations on different CNN architectures finetuned on the training set. The third experiment evaluates the performance of handcrafted representations extracted from the BoVW pipeline and representations extracted from the CNN. The last experiment improves the performance by combining multiple CNNs into an ensemble.

The dataset used in the experiments consists of 2533 images each belonging to one of $15$ object categories. These categories were taken from Caltech-101 dataset and include bonsais, butterflies, crabs, elephants, euphoniums, faces, grand pianos, Joshua trees, leopards, lotuses, motorbikes, schooners, stop signs, sunflowers, and watches. Each category contains different number of images and are of different resolutions. Example images from each category are shown in figure 3.

The dataset was split into two sets: training and testing sets. The first $30$ examples of each class were used as a training set for a total of $450$ images while the rest was used as a testing set for a total of $2083$ images.

To demonstrate the results, we used MATLAB and two external libraries: VLFeat [75] and Lib-linear [76]. The results were reported using top 1 and top 2 accuracy evaluation metrics. Precisions for each category were also reported.

### 6.1 BoVW Pipeline Enhancements Experiments

In these experiments, we evaluate the performance of shallow representations for the enhancements proposed for different encoding methods. Furthermore, we compare these results to previous results that do not use the proposed enhancements.

*Local Feature extraction and enhancement.* All images were converted to grayscale even when color is available. Since all images have a medium resolution $400 \times 500$, images were processed as they are without resizing them. We only used a single descriptor type, i.e., SIFT descriptors extracted densely from an image with a stride of $4$ pixels at $7$ different scales with $\sqrt{2}$ scale increments. The spatial bin size of the SIFT descriptor was fixed to cover $8$ pixels. The feature descriptors were enhanced by running them through the feature enhancement pipeline step where they were square-rooted as suggested in [4], reduced in dimensionality from $128$ to $80$ dimensions using PCA, and then whitened. Moreover, the reduced descriptors were augmented with their normalized spatial location to get $82$-dimensional descriptors. For performance reasons, we didn't use feature augmentation for FV.

*Codebook Generation.* A codebook of size $1024$ codewords was constructed by clustering a random subset of feature descriptors from the training set using $k$-means clustering algorithm. The codebook was fixed in size (to $1024$) for all experiments except for FV, and VLAD encoding methods which use a codebook of size 256 codewords. For VLAD, the codebook was constructed using $k$-means, whereas a GMM was used for FV. For GMM training, the mixture parameters were learnt from the training set using Expectation Maximization (EM, [29]) algorithm. The means of the GMM were initialized by the means of $k$-means algorithm run for a few iterations. The diagonal of the covariance matrix of each Gaussian was initialized by the diagonal of the covariance matrix of the points assigned to initial mean of each Gaussian. The mixing coefficients of each Gaussian were initialized by the proportion of points assigned to each Gaussian.

*Encoding Methods.* We compare the proposed pipeline enhancements to seven popular encoding methods used in the codebook model and for each we provide the parameters used in the encoding process:

- HA: It uses hard quantization and requires computing the nearest neighbor for each feature descriptor.
- LSA: It uses soft quantization and requires the computation of the $k$ nearest neighbors to each feature descriptor. The parameter $k$ was set to 5, and $\beta$ was set to $2 \times 10^{-4}$.
- FV: It represents feature descriptors by their first and second order statistics for each codeword.
- VLAD: It represents each descriptor by the vector difference between the descriptor and its closest codeword. It requires the computation of the nearest neighbor.
- LLC: The approximated version of LCC was used to achieve both sparsity and locality. The parameter $k$ was set to $5$, and $\lambda$ was set to $10^{-4}$.
- SC: It requires the computation of the $k$ nearest neighbors for each descriptor. Each descriptor is encoded just by its closest codeword as in HA. Unlike HA, the coding weight is not constant and depends on the distances of the descriptor from its $k$ nearest codewords. The parameter $k$ was set to $5$.
- GSC: It is a soft version of SC and requires the computation of the $k$ nearest neighbors for each descriptor. The parameter $k$ was set to $5$.

*Spatial information.* To introduce weak geometry to the representation, spatial pyramid was used for all experiments. It divides the image into three levels: $1 \times 1$, $2 \times 2$, and $4 \times 4$ for a total of $21$ regions.

*Pooling and Normalization.* Different pooling and normalization strategies were used for different encoding methods. Region encodings are pooled separately and normalization is done for all regions. The following pooling and normalization strategies were used for each encoding method:

- HA: The resulting coding vectors from different feature descriptors are sum-pooled, sign-square-root normalized, globally $l_2$-normalized, and again sign-square-root normalized. The final representation has $1024 \times 21 = 21{,}504$ dimensions.
- LSA: Same as in HA.
- FV: Average-pooling is used for image coding vectors to get the pooling vector. Each gradient sub-vector of the pooling vector is $l_2$-normalized (intra-normalization), and each region is also $l_2$-normalized. After that, sign-square-root normalization followed by global $l_2$-normalization is done on the whole representation. The dimensionality of the representation is $82 \times 2 \times 256 \times 21 = 881{,}664$.

| Encoding Methods | Top1 Accuracy (%) | Top2 Accuracy (%) |
|---|---|---|
| HA | 96.35 | 98.94 |
| LSA | 96.74 | 98.90 |
| FV | 95.58 | 98.08 |
| VLAD | 96.69 | 98.80 |
| LLC | 97.12 | 99.09 |
| SC | 96.50 | 98.90 |
| GSC | 97.22 | 98.99 |

(a)

| Encoding Methods | Top1 Accuracy (%) | Top2 Accuracy (%) |
|---|---|---|
| HA | 91.60 | 96.45 |
| LSA | 93.95 | 98.03 |
| FV | 95.58 | 98.08 |
| VLAD | 94.38 | 98.22 |
| LLC | 93.61 | 98.03 |
| SC | 93.37 | 97.65 |
| GSC | 92.80 | 96.69 |

(b)

**TABLE 2:** Top 1 and top 2 accuracies of different encoding methods. (a) Using our proposed enhancements in the feature extraction and normalization steps of the recognition pipeline. (b) Previous methods implemented as in their original papers including enhancements introduced afterwards.

- VLAD: Coding vectors corresponding to image feature descriptors are sum-pooled, and the resulting pooling vector is sign-square-root normalized, and $l_2$-intra-normalized. The representation has $82 \times 256 \times 21 = 440{,}832$ dimensions.
- LLC: Max-pooling is used followed by global $l_2$ and sign-square-root normalization. The representation has $1024 \times 21 = 21{,}504$ dimensions.
- SC: Max-pooling followed by same normalization as in HA. The representation has $1024 \times 21 = 21{,}504$ dimensions.
- GSC: Max-pooling followed by the same normalization as in HA. The representation has $1024 \times 21 = 21{,}504$ dimensions.

*Classifier*. In all experiments, multi-class classification is done with $15$ SVM classifiers trained using one-versus-all setting. We did not use nonlinear kernels or any explicit feature mapping. Only linear kernels were used. The SVM regularization parameter was set to $10$.

*Parameters used for previous results.* We compared our enhancements to previous settings in the recognition pipeline. To achieve a fair comparison, we have tested the previous encoding methods with the parameters used in their original papers or with enhancements introduced afterwards. As before, we extracted SIFT descriptors with the same scales and sampling frequency. We do not perform square rooting for features except for FV. Moreover, features are neither augmented nor reduced in dimensionality (except for FV and VLAD which were reduced to 80 dimensions). The same pooling strategies and pyramid levels were used for all encoding methods as before. For normalization, we used $l_2$-normalization for individual regions and global $l_2$-normalization for the whole representation. Moreover, we used $l_2$-intra-normalization for encoding methods that produce a block of codes for each codeword, i.e., FV and VLAD. For classification, we used the SVM classifier as before. Intersection kernel was used for HA and LSA, Hellinger kernel for FV and VLAD, while a linear kernel was used for the rest of encoding methods. For nonlinear kernels, we used explicit feature mapping using additive homogeneous kernels.

*Results and analysis.* Results for the enhancements proposed are summarized in table 2(a). Different settings for spatial information, feature reduction, and pooling and normalization parameters were tested for each experiment. A performance increase was noticed when we introduce the feature enhancement step in the recognition pipeline and a sequence of representation normalization that is based on $l_2$- and sign-square-root normalization as described above. The performance increase was attained using the efficient and inexpensive linear kernel which yield same or better performance than nonlinear kernels using the above parameter settings. The results show that the relatively fast and sparse methods like GSC, LLC, HA, LSA, and SC yield better results than the slow and dense FV method. For this dataset, we see that

among single-dimensional response hard coding methods, SC gives better results than HA. Moreover, the multi-dimensional response hard coding method, VLAD, gives better performance than single-dimensional response hard coding methods like HA and SC. Among sparse soft encoding methods, we see that GSC and LLC produce better result than LSA. Among multi-dimensional response coding methods, we see that VLAD produces better results than FV. Among all methods, we see that GSC produces the best result. Moreover, LLC and GSC produce the best results in terms of top $2$ accuracy. Class precisions for different encoding methods using the proposed enhancements are shown in table 4(a).

Results for previous encoding methods are shown in table 2(b). We see that the FV outperforms other methods. Despite its best accuracy, FV is the slowest among all encoding methods since it needs more computation and memory resources for its multi-dimensional representation. However, we have achieved the best result using the faster and more efficient GSC encoding method using the proposed enhancements. Moreover, we see a large performance gap favoring our proposed enhancements over the previous methods for all encoding methods. Class precisions for previous results are given in table 4(b).

### 6.2 CNN Architectures Experiments

In these experiments, we finetune different pretrained deep CNN architectures on our dataset. Fine-tuning a network with transfer learning is usually much faster and easier than training a network with randomly initialized weights from scratch, and this might be necessary in case you don't have enough training data. The early layers of a pretrained CNN learn low-level features (blobs, corners, edges) that are general to any task, and the last layers learn features that are task specific. Therefore, early layers were retained with their weights, and last layers were replaced with new layers of our own to learn features that are specific to our dataset.

*CNN Architectures.* Different CNNs expect different image sizes, so images were resized to fit the input layer of each network. Moreover, since the dataset contains a mixture of grayscale and RGB images, color preprocessing is important to ensure that all the images have the same number of channels required by the network. In the experiments, all images were converted to RGB. in the experiments, four popular and recent CNNs were used:

- AlexNet: It was named after Alex Krizhevsky [66] and has won the ILSVRC2012 competition on object classification and considered the first CNN-based winner. The network consists of 8 layers, 5 convolutional and 3 FC layers and uses $11 \times 11$, $5 \times 5$, and $3 \times 3$ convolutions. The network has around 60 million parameters and expects input of size $227 \times 227$ RGB image.

- VGG19: It was designed by Simonyan and Zisserman [67] and achieved the second rank in ILSVRC2014 object classification and the first in object localization. It consists of 19 layers, 16 convolutional and 3 FC layers. The network uses only $3 \times 3$ convolutions but lots of filters and is deeper than AlexNet. It contains around $138$ million parameters and expects inputs of size $224 \times 224$ RGB image.

- GoogLeNet: It was designed by Google and has won the ILSVRC2014 competition on object classification. It is deeper than VGG19 and consists of $22$ layers. It features $9$ efficient inception modules where each module uses $5 \times 5$, $3 \times 3$, and $1 \times 1$ convolutions. It does not contain FC layers and contains only $5$ million parameters. The network expects inputs of size $224 \times 224$ RGB image.

- ResNet: It was designed by Microsoft and has won the ILSVRC2015 competition on object classification. It is much deeper and consists of 152 layers. To improve the training process, it introduced a novel architecture with "skip connections" and featured heavy batch normalization.

| CNN | Top 1 Accuracy (%) | Top 2 Accuracy (%) |
|---|---|---|
| AlexNet | $97.95 \pm 0.34$ | $99.30 \pm 0.19$ |
| VGG19 | $98.62 \pm 0.11$ | $99.57 \pm 0.14$ |
| ResNet50 | $98.65 \pm 0.14$ | $99.49 \pm 0.11$ |
| GoogleNet | $98.75 \pm 0.17$ | $99.55 \pm 0.07$ |

(a)

| Experiment | Top 1 Accuracy (%) | Top 2 Accuracy (%) |
|---|---|---|
| Deep features on linear SVM | 99.04 | 99.38 |
| Ensemble of 5 CNNs | 99.47 | 99.52 |

(b)

**TABLE 3:** (a) Top 1 and top 2 accuracies for different deep network architectures. (b) Top row: deeply learnt features extracted from a CNN and trained on an SVM. Bottom row: ensemble of 5 CNN learners combining their predictions by taking the maximum posterior probability.

It expects inputs of size $224 \times 224$ RGB image. In the experiment, we have used only 50 layers (ResNet50).

*Training Parameters*. The last FC, softmax, and classification layers were removed from each network and new FC, softmax, and classification layers were added. The weights of the CNN initial layers were frozen while were allowed to update in last layers. The unfrozen layers were trained on the training set using Stochastic Gradient Descent with Momentum (SGDM) for a maximum of $10$ epochs. The momentum was set to $0.9$ and the effective learning rate was set to $10^{-3}$. The $l_2$-regularization constant for weights of the FC layer was set to $10^{-4}$ with zero regularization for biases. Since these networks contain some stochastic layers, the networks were trained and tested for $5$ times and the average accuracies and standard deviations were reported.

*Results*. Results for different CNN architectures are shown in table 3(a). They are better than previous results for shallow representation encoding methods. Some CNNs give better results than others on the dataset. For example, GoogLeNet gives the best result among all and is fast to train and test. Second comes Resnet50 which gives comparable result and is also fast to train and test. VGG19 gives good performance but is slow in training and testing. Finally comes AlexNet which gives relatively lower result but is fast at training and testing. Precisions for each class are given in table 4(c).

### 6.3 Shallow vs. Deep Representations Experiment

In this experiment, we compare between hand-engineered shallow features produced using the previous encoding methods and deeply learnt features extracted from a CNN. Here, we treat a CNN as a generalized feature extractor, and activations of the last newly added fully connected layer were used as the image representation. These representations were then fed to a linear SVM for classification. Same parameters for SVM were used as before.

*Results and analysis*. Results for this experiment are shown in table 3(b). Despite the low dimensionality of deep features (15 dimensions in our case), they produce better results than the high dimensional (tens of thousands) handcrafted features. Moreover, the performance gain is achieved without using any normalization method or applying nonlinear kernels. Class precisions are given in table 4(d).

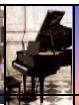
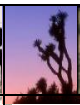
| | Class / Expr. |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| (a) | HA | 93.88 | 86.89 | 79.07 | 97.06 | 91.18 | 99.75 | 98.55 | 97.06 | 100 | 86.11 | 97.01 | 100 | 94.12 | 94.55 | 93.78 |
| | LSA | 92.86 | 88.52 | 83.72 | 94.12 | 97.06 | 99.51 | 92.75 | 97.06 | 100 | 91.67 | 98.31 | 100 | 94.12 | 89.09 | 94.26 |
| | SC | 90.82 | 88.52 | 86.05 | 97.06 | 94.12 | 99.75 | 92.75 | 97.06 | 100 | 88.89 | 98.44 | 100 | 94.12 | 87.27 | 92.34 |
| | GSC | 92.86 | 88.52 | 83.72 | 97.06 | 97.06 | 100 | 94.20 | 97.06 | 100 | 91.67 | 98.83 | 100 | 94.12 | 89.09 | 95.22 |
| | LLC | 92.86 | 86.89 | 86.05 | 97.06 | 97.06 | 99.75 | 94.20 | 97.06 | 100 | 91.67 | 98.96 | 100 | 94.12 | 87.27 | 94.74 |
| | VLAD | 94.90 | 83.61 | 88.37 | 100 | 94.12 | 99.75 | 98.55 | 97.06 | 100 | 91.67 | 96.35 | 100 | 94.12 | 94.55 | 96.17 |
| | FV | 84.69 | 81.97 | 79.07 | 94.12 | 91.18 | 99.51 | 97.10 | 100.00 | 100 | 77.78 | 98.70 | 96.97 | 85.29 | 87.27 | 91.87 |
| (b) | HA | 94.90 | 85.25 | 83.72 | 94.12 | 94.12 | 99.26 | 98.55 | 91.18 | 100 | 86.11 | 84.90 | 96.97 | 94.12 | 94.55 | 94.90 |
| | LSA | 94.90 | 86.89 | 83.72 | 94.12 | 94.12 | 99.51 | 97.10 | 91.18 | 99.41 | 83.33 | 91.80 | 96.97 | 91.18 | 92.73 | 91.87 |
| | SC | 84.69 | 83.61 | 79.07 | 91.18 | 91.18 | 99.75 | 95.65 | 91.18 | 100 | 80.56 | 92.84 | 100 | 94.12 | 83.64 | 91.39 |
| | GSC | 90.82 | 85.25 | 79.07 | 91.18 | 91.18 | 99.51 | 95.65 | 91.18 | 99.41 | 86.11 | 88.93 | 100 | 97.06 | 94.55 | 93.30 |
| | LLC | 95.92 | 86.89 | 79.07 | 94.12 | 94.12 | 99.51 | 97.10 | 91.18 | 100 | 88.89 | 90.36 | 100 | 94.12 | 90.91 | 92.34 |
| | VLAD | 93.88 | 80.33 | 83.72 | 97.06 | 94.12 | 98.77 | 100 | 97.06 | 99.41 | 91.67 | 91.67 | 100 | 94.12 | 92.73 | 95.69 |
| | FV | 84.69 | 81.97 | 79.07 | 94.12 | 91.18 | 99.51 | 97.10 | 100 | 100 | 77.78 | 98.70 | 96.97 | 85.29 | 84.69 | 81.97 |
| (c) | AlexNet | 96.53 | 85.90 | 97.67 | 97.06 | 92.94 | 99.41 | 100 | 98.24 | 97.41 | 90.56 | 99.17 | 100 | 100 | 100 | 95.60 |
| | VGG19 | 98.16 | 92.13 | 97.21 | 98.24 | 95.88 | 99.21 | 100 | 95.29 | 98.82 | 87.22 | 99.53 | 99.39 | 100 | 97.82 | 98.76 |
| | ResNet50 | 97.96 | 92.79 | 95.81 | 97.65 | 96.47 | 99.26 | 100 | 96.47 | 95.41 | 93.89 | 99.77 | 100 | 97.96 | 92.79 | 95.81 |
| | GoogLeNet | 95.92 | 94.43 | 93.49 | 99.41 | 91.18 | 99.60 | 100 | 92.94 | 99.06 | 91.11 | 99.87 | 100 | 100 | 97.45 | 99.33 |
| (d) | Deep features | 96.12 | 87.87 | 96.74 | 98.24 | 94.71 | 99.51 | 100 | 98.24 | 98.00 | 89.44 | 99.27 | 100 | 100 | 100 | 96.17 |
| (e) | CNN ensemble | 97.96 | 96.72 | 97.67 | 100 | 97.06 | 100 | 100 | 100 | 98.24 | 94.44 | 100 | 100 | 100 | 100 | 100 |

**TABLE 4:** Class precisions for different methods. (a) Encoding methods with enhancements in the recognition pipeline. (b) Encoding methods with previous settings without the proposed enhancements. (c) Average precision for deep network architectures trained and tested five times. (d) Deeply learnt features extracted from a CNN and classified on an SVM. (e) Ensemble of 5 CNN architectures.

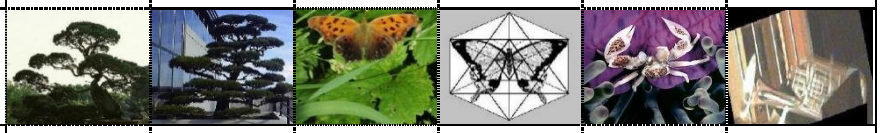
### 6.4 CNN Ensemble Experiment

In this experiment we combine multiple deep CNN learners into an ensemble using maximum posterior probability. The CNN learners are trained as before and are of different architectures. We achieved the best result using only five learners.

*Results*. The ensemble of deep network learners gives the best results (table 3(b)) among all experiments since it selects the most probable class among all the participating learners. Class precisions are given in table 4(e) and misclassified images along with their predictive probability are shown in figure 4.

## 7. CONCLUSION

In this study, we have reviewed the different steps used in the BoVW recognition pipeline used to extract shallow image representations. Moreover, we have provided a concrete mathematical framework for the different encoding methods used in the pipeline. Several enhancements in the local feature extraction and normalization pipeline steps were introduced that give better image representations and perform well than previous methods. We have achieved a top 1 accuracy of 97.22% for shallow representations using the GSC encoding method which outperforms the best encoding method (FV) in the previous results having a top 1 accuracy of 95.58 %.

| **Actual Class** | bonsai | bonsai | butterfly | butterfly | crab | euponium |
|---|---|---|---|---|---|---|
| **Image** |  | | | | | |
| **Output class** | joshua tree | joshua tree | lotus | schooner | lotus | grand piano |
| **Posterior prob.** | 97.27% | 58.80% | 69.78% | 76.86% | 89.69% | 47.74% |

| **Actual Class** | leopard | leopard | leopard | lotus | lotus |
|---|---|---|---|---|---|
| **Image** |  | | | | |
| **Output class** | joshua tree | joshua tree | joshua tree | sunflower | sunflower |
| **Posterior prob.** | 82.52% | 88.81% | 90.17% | 67.30% | 99.13% |

**FIGURE 4:** Misclassified images for the 5 CNN ensemble experiment. Image actual class, predicted class and the posterior probability are shown above.

Instead of handcrafting the image representations, we have demonstrated that better representations can be extracted by learning them directly from the input image while training the CNN deep learning model. We have achieved a top 1 accuracy of $98.75 \pm 0.17\%$ using GoogleNet CNN and boosted the results to $99.47\%$ using an ensemble of only 5 CNNs. As have been seen, the deeply learnt representations are concise, more discriminative, and outperform shallow representations.

## 8. FUTURE WORK

There are many parameters in the recognition pipeline that need to be further explored. These include: different types of feature descriptors, sampling strategies, clustering algorithms, codebook size, encoding methods, pooling methods, normalization methods, and classification models. Each pipeline step might include other parameters that requires further tuning. For CNNs, different architectures need to be explored. Deeper and wider CNNs are the state-of-the-art architectures. Other CNN tunable parameters include: initialization strategies, pooling methods, optimization algorithms, batch normalization, nonlinearities and many others. Moreover, we want to extend these ideas to video data which has a temporal dimension in addition to the spatial dimensions. Instead of extracting spatial SIFT descriptors, we need other descriptors that take the time dimension into consideration like Histogram of Optical Flow (HOF). The rest of the clustering, encoding, pooling, and normalization methods need to be tested on video data. In the same manner, CNNs need to be changed in architecture to include the temporal dimension of video data. There exist many applications for video recognition such as human action recognition and gesture recognition.

## 9. REFERENCES

[1] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray. "Visual categorization with bags of keypoints". *Workshop on statistical learning in computer vision, ECCV*, 2004.

[2] J. Sivic and A. Zisserman. "Video Google: A text retrieval approach to object matching in videos". IEEE International Conference on Computer Vision, 2003.

[3] S. Lazebnik, C. Schmid, and J. Ponce. "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories". IEEE Conference on Computer Vision and Pattern Recognition, 2006.

[4] R. Arandjelović and A. Zisserman. "Three things everyone should know to improve object retrieval". IEEE Conference on Computer Vision and Pattern Recognition, 2012.

[5] J. Sánchez, F. Perronnin, and T. De Campos. "Modeling the spatial layout of images beyond spatial pyramids". Pattern Recognition Letters, vol. 33, no. 16, pp. 2216-2223, 2012.

[6] D. Lowe. "Distinctive image features from scale-invariant keypoints". International Journal of Computer Vision, vol. 60, no. 2, pp. 91-110, 2004.

[7] H. Bay, T. Tuytelaars, and L. Van Gool. "Surf: Speeded up robust features". European Conference on Computer Vision, 2006.

[8] C. Harris and M. Stephens ". A combined corner and edge detector". Alvey Vision Conference, 1988.

[9] T. Lindeberg. "Feature detection with automatic scale selection," International Journal of Computer Vision, vol. 30, no. 2, pp. 79-116, 1998.

[10] K. Mikolajczyk and C. Schmid. "Indexing based on scale invariant interest points". IEEE International Conference on Computer Vision, 2001.

[11] P. Viola and M. J. Jones. "Robust real-time face detection". International Journal of Computer Vision, vol. 57, no. 2, pp. 137-154, 2004.

[12] E. Rosten and T. Drummond. "Fusing points and lines for high performance tracking". IEEE International Conference on Computer Vision, 2005.

[13] E. Rosten and T. Drummond. "Machine learning for high-speed corner detection". European Conference on Computer Vision, 2006.

[14] J. Matas, O. Chum, M. Urban, and T. Pajdla. "Robust wide baseline stereo from Extremal Maximally Stable regions.," Image and Vision Computing, vol. 22, no. 10, pp. 761-767, 2004.

[15] T. Lindeberg and J. Gårding, "Shape-adapted smoothing in estimation of 3-D shape cues from affine deformations of local 2-D brightness structure." Image and Vision Computing, vol. 15, no. 6, pp. 415-434, 1997.

[16] A. Baumberg. "Reliable feature matching across widely separated views." in IEEE Conference on Computer Vision and Pattern Recognition., 2000.

[17] F. Schaffalitzky and A. Zisserman. "Multi-view matching for unordered image sets, or “How do I organize my holiday snaps?"". European Conference on Computer Vision, 2002.

[18] K. Mikolajczyk and C. Schmid. "Scale & affine invariant interest point detectors". International Journal of Computer Vision, vol. 60, no. 1, pp. 63-86, 2004.

[19] K. Mikolajczyk and C. Schmid. "A performance evaluation of local descriptors". IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 27, no. 10, pp. 1615–1630, 2005.

[20] Y. Ke and R. Sukthankar. "PCA-SIFT: A more distinctive representation for local image descriptors". IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2004.

[21] S. Belongie, J. Malik, J. Puzicha, and M. Intelligence. "Shape matching and object recognition using shape contexts". IEEE Transactions on Pattern Analysis and Machine Intelligence, no. 4, pp. 509-522, 2002.

[22] J. Canny. "A computational approach to edge detection". IEEE Transactions on Pattern Analysis Machine Intelligence, vol. 6, no. 6, pp. 679-698, 1986.

[23] A. Johnson, M. Hebert. "Using spin images for efficient object recognition in cluttered 3D scenes". IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 21, no. 5, pp. 433-449, 1999.

[24] S. Lazebnik, C. Schmid, J. Ponce. "A sparse texture representation using local affine regions". IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 27, no. 8, pp. 1265-1278, 2005.

[25] C. Schmid, R. Mohr. "Local grayvalue invariants for image retrieval". IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 19, no. 5, pp. 530-535, 1997.

[26] J. Flusser. "On the independence of rotation moment invariants". Pattern Recognition, vol. 33, no. 9, pp. 1405-1410, 2000.

[27] S. Lloyd. "Least squares quantization in PCM". IEEE Transactions on Information Theory, vol. 28, no. 2, pp. 129-137, 1982.

[28] M. Muja and D. G. Lowe. "Fast approximate nearest neighbors with automatic algorithm configuration". International Conference on Computer Vision Theory and Applications, VISAPP, vol. 2, 2009.

[29] G. McLachlan and D. Peel. Finite mixture models. John Wiley & Sons, 2004.

[30] R. Arandjelovic and A. Zisserman. "All about VLAD". IEEE Conference on Computer Vision and Pattern Recognition, 2013.

[31] K. Grauman and T. Darrell, "Pyramid match kernels: Discriminative classification with sets of image features," in IEEE International Conference on Computer Vision (ICCV), 2005.

[32] M. Marszalek, C. Schmid, H. Harzallah, and J. van de Weijer. "Learning representations for visual object class recognition". Proceedings of the PASCAL Visual Object Classes Challenge, 2007.

[33] A. Vedaldi, A. Zisserman. "Efficient additive kernels via explicit feature maps," vol. 34, no. 3, pp. 480-492, 2012.

[34] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. "Lost in quantization: Improving particular object retrieval in large scale image databases," IEEE conference on Computer Vision and Pattern Recognition, 2008.

[35] V. Gemert, J. Geusebroek, C. Veenman, and A. Smeulders, "Kernel codebooks for scene categorization," European Conference on Computer Vision, 2008.

[36] V. Gemert, J. Geusebroek, C. Veenman, and A. Smeulders. "Visual word ambiguity". IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 32, no. 7, pp. 1271-1283, 2010.

[37] L. Liu, L. Wang, and X. Liu. "In defense of soft-assignment coding". IEEE International Conference on Computer Vision, 2011.

[38] F. Perronnin, J. Sánchez, and T. Mensink. "Improving the fisher kernel for large-scale image classification". European Conference on Computer Vision, 2010, pp. 143-156.

[39] T. Jaakkola and D. Haussler. "Exploiting generative models in discriminative classifiers". Advances in Neural Information Processing Systems, 1999.

[40] H. Jégou, F. Perronnin, M. Douze, J. Sánchez, C. Schmid, and P. Pérez. "Aggregating local descriptors into a compact image representation". IEEE Conference on Computer Vision & Pattern Recognition, 2010.

[41] J. Yang, K. Yu, Y. Gong, and T. Huang. "Linear spatial pyramid matching using sparse coding for image classification". IEEE Conference of Computer Vision and Pattern Recognition, 2009.

[42] K. Yu, T. Zhang, and Y. Gong. "Nonlinear learning using local coordinate coding". Advances in Neural Information Processing Systems, 2009.

[43] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong. "Locality-constrained linear coding for image classification". IEEE Conference on Computer Vision and Pattern Recognition, 2010.

[44] Y. Huang, K. Huang, Y. Yu, and T. Tan. "Salient coding for image classification". IEEE Conference on Computer Vision and Pattern Recognition, 2011.

[45] Z. Wu, Y. Huang, L. Wang, and T. Tan. "Group encoding of local features in image classification". International Conference on Pattern Recognition, 2012.

[46] K. Yu and T. Zhang. "Improved Local Coordinate Coding using Local Tangents". International Conference of Machine Learning, 2010.

[47] X. Zhou, K. Yu, T. Zhang, and T. Huang. "Image classification using super-vector coding of local image descriptors". European Conference on Computer Vision, 2010.

[48] D. Rumelhart, G. Hinton, and R. Williams. " Learning representations by back-propagating errors". Nature, vol. 323, pp. 533-536, 1986.

[49] G. Cybenko. "Approximation by superpositions of a sigmoidal function". Mathematics of Control, Signals, and Systems, vol. 2, no. 4, pp. 303-314, 1989.

[50] V. Nair and G. Hinton. "Rectified linear units improve restricted boltzmann machines". International Conference on Machine Learning, 2010.

[51] K. He, X. Zhang, S. Ren, and J. Sun. "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification". IEEE International Conference on Computer Vision, 2015.

[52] D.-A. Clevert, T. Unterthiner, and S. Hochreiter. "Fast and accurate deep network learning by exponential linear units (elus)". arXiv preprint arXiv: 1511.07289v5, 2016.

[53] S. Kumar. "On weight initialization in deep neural networks". arXiv preprint arXiv: 1704.08863, 2017.

[54] X. Glorot and Y. Bengio. "Understanding the difficulty of training deep feedforward neural networks". International Conference on Artificial Intelligence and Statistics, 2010.

[55] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. "Imagenet: A large-scale hierarchical image database". IEEE Conference on Computer Vision and Pattern Recognition, 2009.

[56] D. P. Kingma and J. Ba. "Adam: A method for stochastic optimization". arXiv preprint arXiv: 1412.6980, 2014.

[57] L. Wan, M. Zeiler, S. Zhang, Y. Le Cun, and R. Fergus, "Regularization of neural networks using dropconnect". International Conference on Machine Learning, 2013.

[58] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger. "Deep networks with stochastic depth". European Conference on Computer Vision, 2016.

[59] G. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. "Improving neural networks by preventing co-adaptation of feature detectors". arXiv preprint arXiv:.1207.0580, 2012.

[60] D. Hubel and T. Wiesel. "Receptive fields of single neurones in the cat's striate cortex". The Journal of Physiology, vol. 148, no. 3, pp. 574-591, 1959.

[61] D. Hubel and T. Wiesel. "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex". The Journal of Physiology, vol. 160, no. 1, pp. 106-154, 1962.

[62] D. Hubel and T. Wiesel. "Receptive fields and functional architecture of monkey striate cortex". The Journal of Physiology, vol. 195, no. 1, pp. 215-243, 1968.

[63] K. Fukushima. "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position". Biological Cybernetics, vol. 36, no. 4, pp. 193-202, 1980.

[64] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. "Gradient-based learning applied to document recognition". Proc. Of IEEE, vol. 86, no. 11, pp. 2278-2324, 1998.

[65] G. Hinton and R. Salakhutdinov. "Reducing the dimensionality of data with neural networks". Science, vol. 313, no. 5786, pp. 504-507, 2006.

[66] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks". Advances in Neural Information Processing Systems, 2012.

[67] K. Simonyan and A. Zisserman. "Very deep convolutional networks for large-scale image recognition". arXiv preprint arXiv: 1409.1556, 2014.

[68] C. Szegedy et al. "Going deeper with convolutions". IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 1-9.

[69] K. He, X. Zhang, S. Ren, and J. Sun. "Deep residual learning for image recognition". IEEE Conference on Computer Vision and Pattern Recognition, 2016.

[70] K. He, X. Zhang, S. Ren, and J. Sun. "Identity mappings in deep residual networks". European Conference on Computer Vision, 2016.

[71] G. Huang, Z. Liu, L. Van Der Maaten, and K. Weinberger. "Densely connected convolutional networks". IEEE Conference on Computer Vision and Pattern Recognition, 2017.

[72] B. Graham. "Fractional max-pooling". arXiv preprint arXiv: 1412.6071, 2014.

[73] K. He, X. Zhang, S. Ren, and J. Sun. "Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition". European Conference on Computer Vision, 2014.

[74] S. Ioffe and C. Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift". arXiv preprint arXiv:.1502.03167, 2015.

[75] A. Vedaldi and B. Fulkerson. "VLFeat: An open and portable library of computer vision algorithms". ACM International Conference on Multimedia, 2010.

[76] “Lib-linear”. Internet: https://www.csie.ntu.edu.tw/~cjlin/liblinear/, [Jun. 19, 2018].

# Smart Proximity: Annotating the Proximity of Entities In A Smart City Ontology

**Hind A. Alawfi** *haawfi@taibahu.edu.sa*
*College of Computer Sciences and Engineering*
*Taibah University*
*Saudi Arabia*

**Khalid S. Aloufi** *koufi@taibahu.edu.sa*
*College of Computer Sciences and Engineering*
*Taibah University*
*Saudi Arabia*

**Abstract**

The smart city concept contributes a new research area that will continue to be the focus of research for a long time. Different works have modelled and presented ontologies for smart cities, especially for data integration processes. In this context, obtaining a model in which the full functionalities of a DL reasoner are employed to generate new knowledge that would be available to the different devices in a smart city. This information can represent a useful picture of the environment around transports, hubs and people, enabling the smart devices in a city to make decisions according to this environment. We present a model of a smart city ontology with different axioms for generating new knowledge from available knowledge using a DL reasoner. This model considers the location and state of proximity between two entities in the environment. To implement our approach, we develop a tool referred to as smart proximity for generating and querying our smart city ontology. We expect the generated knowledge to be useful to many single working devices, especially devices that are available to transportation, and improve several functionalities such as motion, stop, waiting time and connections between two different means of transport.

**Keywords:** Smart City, Ontology, Proximity, Transportation, Reasoning, Decision Support.

## 1. INTRODUCTION

The world population is shifting into an increasing number of urban areas [1][2]. This shift has been projected to increase over time as the number of people securing jobs and other facilities in these cities increases [3][4]. The increased size of these cities raises a number of complex issues as cities need to manage transport and other infrastructures for improved delivery of services [5][6]. This management is critical to ensure that the air population, waste management, resources, health concerns and other resources, such as traffic congestion, can be managed in an ageing infrastructure. This approach will not only enable cities to improve over time but also ensure that they are able to grow in a sustainable manner [1], [2]. To address this changing nature of cities, numerous different ways of managing these expectations have been proposed. One approach is the smart city approach [5], [6]. A consistent definition of a smart city has not been suggested by academia or practitioners; however, it is consistently seen as a way to use new technologies to convert old cities into a smart initiative [7], [8]. All definitions agree that a smart city will be an urban space that is intended to enhance the everyday lives of citizens [9], [10].

A smart cities approach often attempts to use e-government, information systems, and technologies to integrate the issues faced by cities and obtain plausible explanations for addressing current issues [3][4]. In this context, the use of information and communication technologies (ICT) is considered a fundamental usage factor, which enables individuals and systems to communicate to improve the experiences, habits and facilities that are provided to individuals who live in urban spaces. Figure 1 illustrates the role of ICT in smart cities [11].

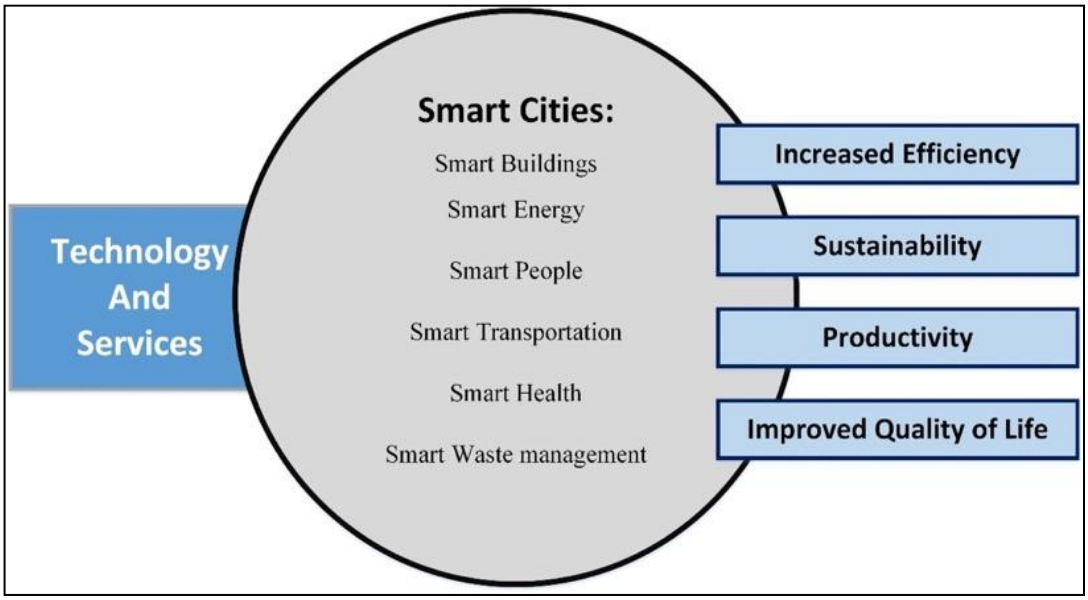



**FIGURE 1:** Role of ICT In Smart Cities [11].

A smart city is often seen as a forward-looking city, where smart combinations of various endowments of environment, mobility, governance, and economy are integrated to create a safe haven for citizens [7], [8]. The city is able to monitor and integrate the conditions to provide an ideal environment for various facilities [12], [13]. Critical infrastructure, such as transport, rail, and communication, can better optimize itself and use resources to undertake maintenance and other activities in a positive manner. Smart city resources can improve the way services are presented to citizens who live within a city. These services include infrastructure and energy management. In particular, improving transport management can be achieved using a smart city initiative since sensors and other information technologies help to organize the components of a transport system [14], [15].

Transportation in smart cities relies on numerous factors. In the first instance, the transport network needs to be integrated in a robust manner to ensure that efficacy of the various networks is maintained [16], [17]. This integration must be coupled with a series of innovations that can integrate various elements of the transport system and ensure that the system can learn from various mistakes that can be improved over time [17], [18].

A factor that is important to consider is that the transport network must be sustainable, which can reduce congestion and traffic [12], [13]. One approach that has been suggested for improvement in smart cities is the use of an integrated approach to ensure that various elements of the transport network can collaborate to ensure a long-term improvement in mitigating traffic and other problems in transport hubs [19]. The implementation of transport hubs and networks that collaborate in an integrated manner must be considered [18], [20]. The development of the transport hubs and networks must enable a citizen to obtain information and options on a continuous basis, which can ensure the sustainability of transport systems and improve the lives of citizens.

The importance of a transport system in a smart city is dependent on the ability of these transport networks to deliver sustainable change, which enables the delivery of new services and ensures that the various elements of a transport network can deliver long-term change and ensure that the sustainability of operations is managed [21]–[23]. However, these smart city initiatives can only be managed if a transport ontology is created, which can improve the semantics and structured data for ease of communication and coordination in smart cities. The goal of smart cities is to provide a more comfortable and safer environment that eases the daily lives of residents. Motivated by this goal, interest in developing more effective smart city applications and ontologies that deliver continuous improvement and address citizens and stakeholder's issues have increased.

One of the smart city areas that has a critical infrastructure with many issues is transportation. Different studies present transportation ontologies for smart cities; most of these studies are within data integration contexts. Most transportation ontologies in the literature focus on specific areas, such as traffic accidents, road traffic management, travel planning, and route planning. Transportation in smart cities can be improved, new knowledge can be produced from available information and smart city devices can facilitate decision-making based on this useful knowledge. The potential to improve the resource usage within the city, increase the effectiveness of the systems, improve the living standards and make smart cities smarter exists.

In this context, the usage of reasoning for generating new useful information is poor. Most previous studies do not consider the distance and proximity between two city objects. No smart city ontology attempts to store axioms for inferring knowledge starting with proximity information.

Different from other models, we take advantage of geospatial data that were furnished by different sensors to gain new knowledge using the reasoning abilities of a standard DL reasoner. With current technology, standard DL reasoners are not capable of inferring new knowledge using geospatial representations of points, lines, and shapes.

Our proposal is to annotate the proximity between two entities in the ontology for which the geospatial location is a known datum. Thus, the reasoner can infer new knowledge from the awareness that some entities are in the same vicinity. This new knowledge can be useful for smart devices in a city, with a focus on the aspect of smart transportation and increases their awareness of the characteristics of the environment in which they move and work.

Some general sample information that can be inferred from the entity proximity in cities is subsequently listed:

- No proximity between people and a bus stop means that the bus can avoid stopping near this bus stop.
- A person with a disability is near a street crossing: the person can have more time to cross the street.
- Ten cars near a parking zone with 10 spots means that the parking zone is probably full.
- Numerous people near a single transportation hub will require police intervention for managing the correct routing of people.
- When I select a destination on a transport, this destination can be translated to a near hub (e.g., “I want to go to the hospital”: a near transportation hub is selected as the destination, and then the person can travel by walking).
- A large truck near a street that is forbidden to trucks can be subjected to sanctions.
- People on a train can be informed whether taxis or shared cars exist near a next station.

We aim to answer the following questions:

- Can the proximity between two smart city objects be represented in an ontology?
- How can reasoning be utilized in ontologies to generate new knowledge?

- How can annotating proximity help smart city devices?

No smart city ontology can infer knowledge starting from proximity information. This task is difficult since geospatial data cannot be handled by standard reasoners.

We simulated the acquisition of several data from city sources by acquiring data about transportation from the open and available General Transit Feed Specification (GTFS) format of data [24] and simulating the acquisition of other data; all data were mapped using the same ontological representation. Due to the abilities of the OWL concept structure, we obtained all data using the same, unified collection of concepts. We subsequently exploited the reasoning abilities of a DL reasoner to obtain new knowledge from the raw data.

This paper is separated into several sections. After the Introduction, there is a section about related work followed by a section about research method describes the methodology for modelling the proximity ontology and annotating the proximity of entities. Then, a section follows about the results and discussion of implementing our smart proximity ontology are presented. The paper concludes with a section about conclusion to discuss the conclusions of our research and future work.

## 2. RELATED WORK

The importance of ontologies as a means for the representation of knowledge within a domain cannot be disregarded. Researchers often look for ways to identify the relationships between a domain and the concepts that define the domain. The transport community can benefit from defining new ontologies, as the data that are being generated from smart cities transport networks will not only be complex but also originate from a variety of sources. Transport data in an integrated smart city will often cause the development of the data from various sources, which can increase the complexity of the data. The complexity of these data will also be challenging, as different types of sensors, which are based on contradictory systems and settings, will create planning problems and delays.

Another problem with transport data in a smart city is the difficulty of mashing up data from various sources [21]. The semantics of the data and the sources of data, will often cause 'mash-ups' [21], which are not easily quantified. These mashups hinder the delivery, understanding, and reliability of the data. An ontology within the transport domain has numerous advantages, which enables a better conceptualization of specifications. Formal ontologies explicitly define semantics, which can be translated into machine-readable languages [21]. One advantage is that a formal ontology can support the different types of semantics and knowledge management issues that are faced by governments and other institutions, which enables the development of various options that can be addressed over time.

The use of these ontologies can produce formal methods, which can increase the sharing and exchange of information without the need for human intervention. These transport ontologies enable different types of systems in a vehicle and household to transmit information between two locations. These systems enable improvement in the data transfer rates, which facilitates the development of improved efficacy in transport networks.

Data management and ontologies need to be undertaken over time in a reusable manner, which enables critical development of large data. The handling of large volumes of data can support interoperability and challenge the different mechanisms that are needed for the smooth operation of smart cities. These transport ontologies have also been designed to scale in size as the system grows and create a system that can enable the development of operations. Numerous transport ontologies have been presented in the literature, which are examined in this review.

These transportation ontologies have been compared in Table 1 to establish the issues that must be addressed.

| Name of Ontology | Proximity Concepts | Complexity of axioms | Evaluation | Knowledge Management Services |
|---|---|---|---|---|
| Content Personalization Transport Ontology [25] | No proximity concepts | Includes detailed axioms | No discussion of evaluation | For decision support |
| Ontology for Transportation Networks [22] | No proximity concepts | Beyond basic taxonomy | No evaluation defined | No services defined |
| Road Traffic Management Ontology [26] | No proximity concepts | Some axioms beyond subclass | Scenario-based evaluation is presented | Decision support |
| Road Accident Ontology [21] | No proximity concepts | No axioms beyond taxonomy | No evaluation defined | No services defined |
| Osmonto [27] | No proximity concepts | Concepts as taxonomy with no additional axioms | No formal evaluation | Route-finding via Open Street Map data |
| GenCLOn [28] | No proximity concepts | Concepts as taxonomy with additional axioms | Compared with model and case studies | Only query answering |

**TABLE 1:** Comparison of The Discussed Transportation Ontologies.

Smart cities are increasingly dependent on the ability of different elements to communicate [23], [29]. Smart cities will have the ability to be monitored by ICT, which enables them to use data from a variety of sources in an integrated manner. The vision of a smart city relies on its ability to transform the lives and mobility of individuals, which can improve the administration [30], [31]. The big data challenges in the smart cities depend on the capability of smart cities to not only monitor the different data sets but also use them in an effective manner [23], [29].

Researchers must study the diverse aspects of a smart city, which can enable a degree of representation to be measured over the use of data. Diverse smart city aspects exist in the development of smart applications [32], [33]. The use of ICT should be considered, as the proximity between two different elements needs to be attained to ensure the long-term success of these projects [34]. The proximity of the objects in smart cities enables numerous advantages [34]. In the first instance, the use of ICT is critical for developing the technologies of smart city applications, which can use the proximity of objects to ensure that applications of smart cities can be implemented. Sensors impaled in vehicles and other objects must be employed to ensure that different location-aware systems can be understood and implemented [35], [36].

Numerous studies have examined the way in which different aspects of a smart city collaborate to not only develop and sustain improvement but also ensure that the development of new sensors can be utilized. The lack of practical knowledge about smart cities can also be disconcerting, as methods in which the proximity in transport networks can be meaningfully understood are need to improve the management of resources [14], [15]. For example, as shown in Table 2, many of the applications for the proximity in transportation are examined, and the success of each application can be considered [14], [15].

| Application | Target | Validity (Scale of 10) |
|---|---|---|
| Smart People | Improvement in qualification, Ability to improve performance | 8 |
| Smart Governance | Participation in decision-making | 10 |
| Smart Living | Health and transport facilities | 6 |
| Smart Economy | Innovation in city | 8 |

**TABLE 2:** Smart City Ontology In Transport.

A smart city transport network will use the proximity information from all resources to ensure that a coordinated response is given to the different types of information that is presented in the smart city. This information needs to be delivered in the most robust manner to improve the efficacy of the information that is being collected [5]. This approach can have numerous advantages for the participants as they can deliver continuous change and delivery of the services in a robust manner [5], [37], which will ensure that the various sensors can improve the services and prompt long-term change [38], [39].

This review has been used to propose an ontology that can include proximity as a key element in addressing the distances and entities in a smart city. This ontology can be developed over time to ensure that the position and distance of an object can be evaluated, which can boost the efficacy and effectiveness of smart city initiatives. The ontology will fill the gap, as a smart city ontology for transportation that can store axioms and infer knowledge from the proximity between two smart city objects, is lacking. This ontology helps to define the locations of objects and improve the interaction among these objects.

## 3. RESEARCH METHOD

### 3.1 Proposed Approach of Smart Proximity

A smart proximity tool is developed for city administrations to make data available in several devices in a city, such as buses, trains and traffic signals. The tool is focused on the smart transportation aspects of a smart city. Figure 2 shows the architecture of the proposed approach for smart proximity.

In particular, the smart proximity tool generates an ontology with information that can produce new knowledge to be furnished to devices in a smart city. This new knowledge is generated starting with proximity information between two entities in the ontology. The new knowledge can be useful for obtaining a portrait of the neighborhood of a device that works in the city, which enable the device to make decisions according to the aspects of its neighborhood and enables the device to present new functionalities. We proceeded as follows:

- We acquired data about transportation in the city of San Francisco (California, USA) that were shared with the General Transit Feed Specification (GTFS) format of data, as these data are available online. The acquired data is used to populate the ontology.
- We modelled an ontology that is related to these data with different concepts regarding entities that move in a city and locations, such as transports, people and local hubs.
- We proceed by defining the format in which GTFS data are shared, and then we present how we mapped these data in an ontological format.
- We added some simulated data, for example, the positions of some pedestrians in the city, as pedestrians are not available in GTFS data. Thus, the information required by the ontology that was not available in the acquired GTFS data was simulated by randomly generating some individuals in a way that was consistent with the remaining data.
- For a proximity representation, the proximity between two entities is annotated using a proximity entity of the type Proximity.

An example of annotating the proximity between a bus and a bus stop of 3 cm is presented as follows:

**Individual: bus1**
**Types: Bus**
**Individual: busStop1**
**Types: BusStop**
**Individual: p1**
**Types: Proximity**
**Facts: hasEntity bus1, hasEntity busStop1, hasDistance 3.0**

- We proceeded by defining some sample axioms in the ontology that can generate new knowledge from the available knowledge; for example, identify a bus near a hub with no people in the vicinity, which is useful as a bus can become aware that it does not have to stop near this hub, as no people have to be picked up.
- We implemented a tool that is referred to as smart proximity to generate and query the ontology.

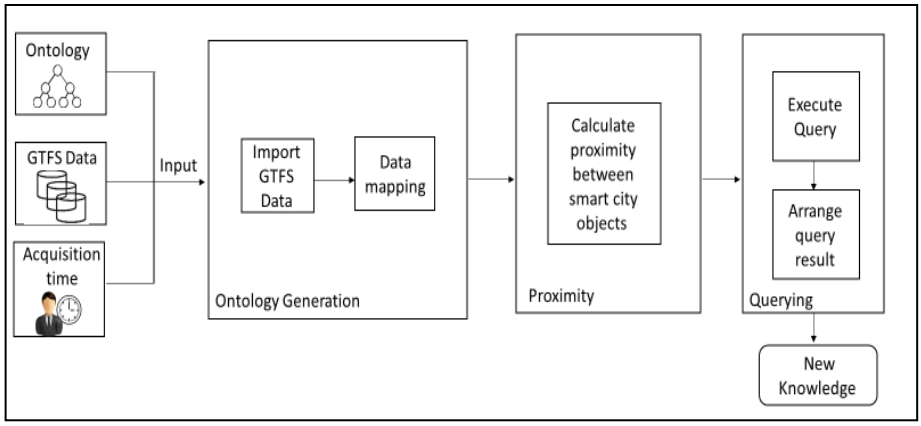



**FIGURE 2:** Smart Proximity Architecture.

### 3.2 Ontology Definition

We defined our smart city ontology by a data-driven approach. We start by identifying the concepts to model within the ontology. Although many concepts must be modelled in an ontology, we focused on the locations of transportation, people, hubs, and traffic signals, and the proximity between them since they are related to the acquired GTFS data. We build our ontology to enable other ontologies to reuse the entire ontology or some of its elements. Figure 3 illustrates the main classes and relations in our ontology.

The defined ontology consists of different concepts regarding entities that move in a city and locations, such as transports, traffic signals, people and local hubs. The following main classes are represented:

- Hub: is a single location in which people gather to enter or exist a mode of transport. A hub can be, e.g., a train station, bus stop, or taxi stand.
- Person: represents a single person walking in a city. We assume that a device exists in the city for tracking a person's location, such as available wearable devices or inference trough cameras.
- Transportation: represents the class of any transportation in the city. Bus, Cable Car, Rail and Taxi are subclasses of this class.
- Traffic Light: represents the set of all detected traffic signals in a city.
- Point: represents a point located on earth with a latitude and a longitude. A point is a concept imported from the WGS84 Geo ontology [40]. Every Hub, Person, Transportation or Traffic Signal is connected to a Point with the hasLocation property.
- Proximity: represents near objects in a city. Thus, each proximity individual represents two points that are located near each other.
- Disability: people can have one or more disabilities; these disabilities are annotated in the ontology. We assume that people with disabilities agree to anonymously share their status via a wearable device.

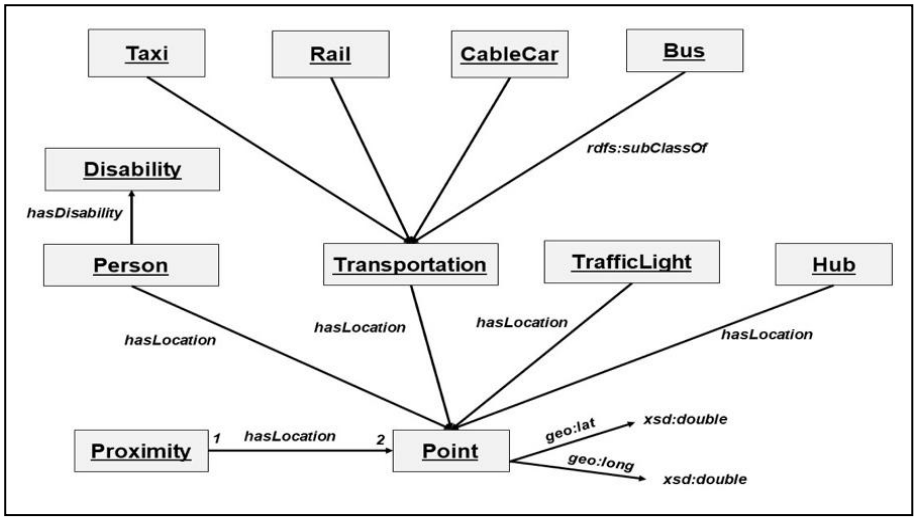



**FIGURE 3:** Main classes and Relations In Our Ontology.

### 3.3 Proximity Representation

In the smart city context, geospatial information is the core concept on which these smart cities are built. Geospatial information connects all smart cities objects, components, and applications. Thus, smart cities that are dependent on a geographical information system (GIS) due to their ability to efficiently model the real world of cities among other benefits aim to highlight the best features of smart cities [41]. The author in [41] argued that smart cities are spatially enabled cities, in which the spatial information of city objects are accessible by governments, stakeholders, and people.

Since everything in a smart city and its entities are based on geospatial information, we represent the proximity concept based on geographic coordinates, which are latitude and longitude. Consequently, our representation of proximity can be applied to other fields other than transportations with the goal of producing new knowledge and efficiently providing solutions. Our proposed tool can enable a user to import our representation of proximity into an external ontology. Thus, this tool can be applied to other smart city fields.

We realize that available DL reasoners cannot understand the proximity between two geo-located points, as the math that would be needed is not supported by current technology. Thus, in our proposed approach; the proximity between two Points is annotated in the ontology by generating an individual of the type Proximity and connecting two near points using the hasLocation property. Figure 4 shows an example of the proximity between two individuals of the type Person and Taxi.

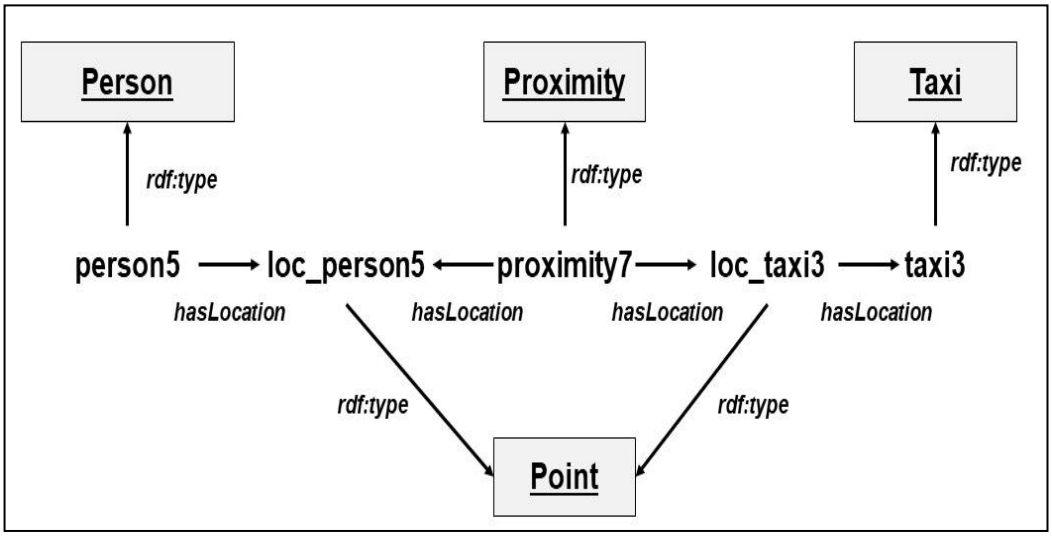



**FIGURE 4:** Example of proximity between two individuals of the type Person and Taxi.

We start by calculating the distance between two points. The distance between two points is calculated if their latitude and longitude are known. We compare the resulting distance with a pre-defined value that represents the maximum distance for two points to be considered near each other. If the distance is a pre-defined value, then we connect these two points with a proximity individual. Thus, all individuals of the proximity entity represent near points.

### 3.4 Algorithm of Mapping GTFS Data

To populate the ontology, we acquired data about transportation in San Francisco, which is one of the smart cities around the world. The data are presented in GTFS format, which is a standard format for sharing transportation schedules and other related information. The files that contain these data are Comma Separated Values (CSV) files. The files of GTFS data contain information about the trips performed by transportation and their times. They also contain information about the geospatial points associated with each trip and information about hubs locations for all trips. Additional details about the files of GTFS data are presented later. Our proposed tool offers the user the possibility to upload an external GTFS Data into our ontology.

We map the GTFS data into ontological propositions to be input into the ontology. Thus, the data are equivalent, but their format will change. The main scope of our ontology is to have at our disposal a single portrait of the locations of hubs, transportations, traffic signals and people in the city in a single instant in time. The locations of transportations in GTFS data are reported for the entire day. Thus, we do not link every information but only the information needed to identify the location of transportation in a single instant in time. Regarding transportation, we cannot directly acquire the location of the transportations as it is not reported in GTFS data for every instant. Thus, we must perform some estimations. We developed an algorithm for mapping GTFS data into the ontology. A pseudocode for this algorithm is presented in Figure 5. The algorithm proceeds as follows:

1. Select the time to perform the mapping. We refer to this time as the acquisition time.
2. Obtain all trips that are active during the selected time by checking if the acquisition time falls between the departure time and the arrival time of a trip.
3. Estimate the travelled distance by the transportation at the acquisition time by computing the time between the first departure time and the acquisition time. The distance is estimated by calculating the ratio between the time passed from departure and the total time of the trip. We use the same ratio to estimate the distance travelled at the acquisition time. For example, 10 minutes passed on a trip of 40 minutes; thus, 25% of the time of

the trip is passed. We estimate that 25% of the distance of the trip was travelled. We use this percentage to identify the point nearest to the calculated ratio. For example, the total distance of a trip is 2000 meters; 25% of 2000 is 500 meters. We look for a point for which the reported distance travelled to the nearest value is 500, that is, the point on which we locate the transportation. Thus, the latitude and longitude of the point are mapped in the ontology as the latitude and longitude of the transportation. Additional transportation information is acquired from the GTFS files.

4. Locations of hubs are mapped from the GFTFS files.
5. People are not available in the GTFS. Thus, we map them using a simulation. The locations of people are randomly generated in a city, with an improved probability in the proximity of a hub. For every four hubs, one hub has people.
6. Traffic signals are not available in the GTFS. Thus, we map them using a simulation. We create a file that contains traffic signal locations that exist in San Francisco. We map this file into the ontology.
7. Taxis are not available in the GTFS. Thus, we map them using a simulation. The locations of taxis are randomly generated in the city, with a better probability in the proximity of a hub. For every two rails, one taxi exists.
8. Every distance between two points is checked. Points with a distance lower than the pre-defined threshold are connected through a Proximity individual.

**Algorithm** MAPPING_OF_GTFS_DATA_PROCESS()

$acq = acquistion\ time\ selected\ by\ the\ user$
$T_o = set\ of\ trips\ in\ the\ trips.txt\ file$
$T_a = \emptyset$
$L = \emptyset$

**for each** $t \in T_o$
**do** $\begin{cases} t_{dep} = first\ departure\ time\ of\ t\ in\ stops.txt \\ t_{arr} = last\ arrival\ time\ of\ t\ in\ stops.txt \\ \textbf{if}\ acq > t_{dep} \wedge acq < t_{arr} \\ \quad \textbf{do}\ \{T_a \leftarrow t \end{cases}$

**for each** $t \in T_a$
**do** $\begin{cases} \tau = \frac{t_{arr} - acq}{t_{arr} - t_{dep}} \\ dist_{tot} = total\ distance\ of\ t's\ journey\ acquired\ from\ shapes.txt \\ dist_{acq} = \tau \cdot dist_{total} \\ t_{loc} = point\ with\ nearest\ distance\ to\ dist_{acq}\ from\ shape.txt \\ L \leftarrow (t, t_{loc}) \end{cases}$

**FIGURE 5:** Pseudocode of Mapping GTFS Data Algorithm.

### 3.5 Reasoning Mechanism About Entities and Proximity

The ontology is considered a picture of a single instant in time of the location of people, hubs, transports and traffic signals in the city; hubs and traffic signals do not move, of course, and their locations are always identical each time. A location for each entity is annotated in the ontology with the proximity between two locations when their distance is below a certain threshold. We

present four use cases for using this knowledge to generate new information that can be useful for the activities of decision-making of devices in a smart city environment.

**Identifying hubs with people and hubs without people**
Monitoring people at hubs is important for transport efficiency issues. Automatic transportation can only stop near hubs that have people. Taxis can efficiently move where they have a greater probability of being needed. Security can be efficiently moved where the risk of social issues is substantial.

If we want to identify hubs with more than 10 people, we can define a class **PopulousHub** as follows:

| | |
|---|---|
| *Class:* | *PopulousHub* |
| *SubClassOf:* | *Hub and hasLocation some (Point and inverse(hasLocation) min 10 (Proximity and hasLocation some (Point and inverse(hasLocation) some Person)))* |

which means a **Hub** in a point that is connected to a minimum of 10 **Proximity** individuals who are each connected to another **Point** that is the location of a **Person**.

Note that we need to set the assumption that all individuals are distinct to true in the ontology. Otherwise, the reasoner can assume that two or more individuals of the type **Person** refer to the same individual, which produces an unexpected inference behaviour.

We may want to infer the situation in which no people exist at the hub. Working the reasoner under the open world assumption, we are unable to make the reasoner automatically infer that no people exist at a hub. When we annotate zero people near the hub, the reasoner considers that some people that are not annotated in the ontology can exist near the hub due to the open world assumption status of the reasoner: If you do not state a proposition in the ontology, it may not be false.

During the mapping phase, we decided to annotate all hubs without people by enabling these hubs to be individuals of the class HubWithNoPeopleAround. Thus, hubs without people are marked during the mapping phase and not using reasoning.

**Identifying transportation near hubs without people**
When transportation approaches a hub, it usually stops; however, it does not have to stop if people do not exist at the hub. To identify transportation with empty hubs, assuming that HubWithNoPeopleAround is already computed in the ontology, we define the new class TransportationWithNoNearStop:

| | |
|---|---|
| *Class:* | *TransportationWithNoNearStop* |
| *SubClassOf:* | *Transportation and (hasLocation some (Point and (inverse (hasLocation) some (Proximity and (hasLocation some (Point and (inverse(hasLocation) some HubWithNoPeopleAround)))))))* |

which means a Transportation in a Point that is linked to a Proximity individual that is also connected to the Point, which is the location of a HubWithNoPeopleAround.

**Triggering a slow stop status for a traffic signal as a person with a disability is near it**
When a person with a disability needs to cross a street, providing this person with additional time to cross is reasonable.

This approach can be achieved by identifying traffic signals with people with some disabilities near them:

*Class:* *SlowStopTrafficLight*
*SubClassOf:* *TrafficLight and (hasLocation some (Point and (inverse (hasLocation) some (Proximity and (hasLocation some (Point and (inverse (hasLocation) some (Person and (hasDisability some Disability)))))))))*

**Obtaining trains with the presence of at least one taxi at a nearby station**
While riding the rails, a passenger may want to be informed using her/his wearable device, if taxis are available at the near station. We can identify trains that have at least one taxi in the proximity of the near station as follows:

*Class:* *RailWithTaxiAtNearHub*
*SubClassOf:* *Rail and (hasLocation some (Point and (inverse(hasLocation) some (Proximity and (hasLocation some (Point and (inverse (hasLocation) some (Hub and (hasLocation some (Point and (inverse(hasLocation) some (Proximity and (hasLocation some (Point and (inverse (hasLocation) some Taxi)))))))))))))))*

### 3.6 Querying the Ontology

In our proposed approach, we use SPARQL-DL queries to query the ontology. When executing a SPARQL_DL query; we display it result in a table. However, one of the problems of the SPARQL-DL queries is that the order of variables within the query expression is not retained when showing the results in a table. Thus, we must reorder the variables as we expressed them in the query.

First, we start by creating a table to record the results. Second, we obtain the order of the variables by extracting them from the query expression. The obtained variables are then written as columns of the created tables. We extract the rows of the query results by examining the results for the selected variable in their order. When we obtain them, we input the results in the correct cell of the row data. Last, we add the rows to the created table that has the correct order of variables as columns.

## 4. SMART PROXIMITY: RESULTS AND DISCUSSION

Smart proximity is a tool that is used to generate a transportation ontology, map GTFS data to the ontology and query it. Our tool provides four use cases that aim to provide new knowledge to smart city devices by allowing them to make decisions based on the entities near them.

We implement our smart proximity tool for generating the ontology and query it using JAVA 8. Our ontology is an OWL/RDF ontology that is defined via Protégé, which is an ontology editor for creating ontologies. Two APIs are employed: OWL API 3.4.4 and Java SPARQL-DL query tool. OWL API 3.4.4 is used to write the ontology and populate our ontology with GTFS data. Although SPARQL-DL API is used to query the ontology and obtain data from it. The JAVA SPARQL-DL query tool developed by the Derivo Company uses HermiT as a reasoner [42]. Figure 6 illustrates the main modules of the tool.

The implementation of our proposed approach undergoes five main phases—data acquisition, ontology definition, data mapping, proximity representation and query processing which are shown in Figure 6.

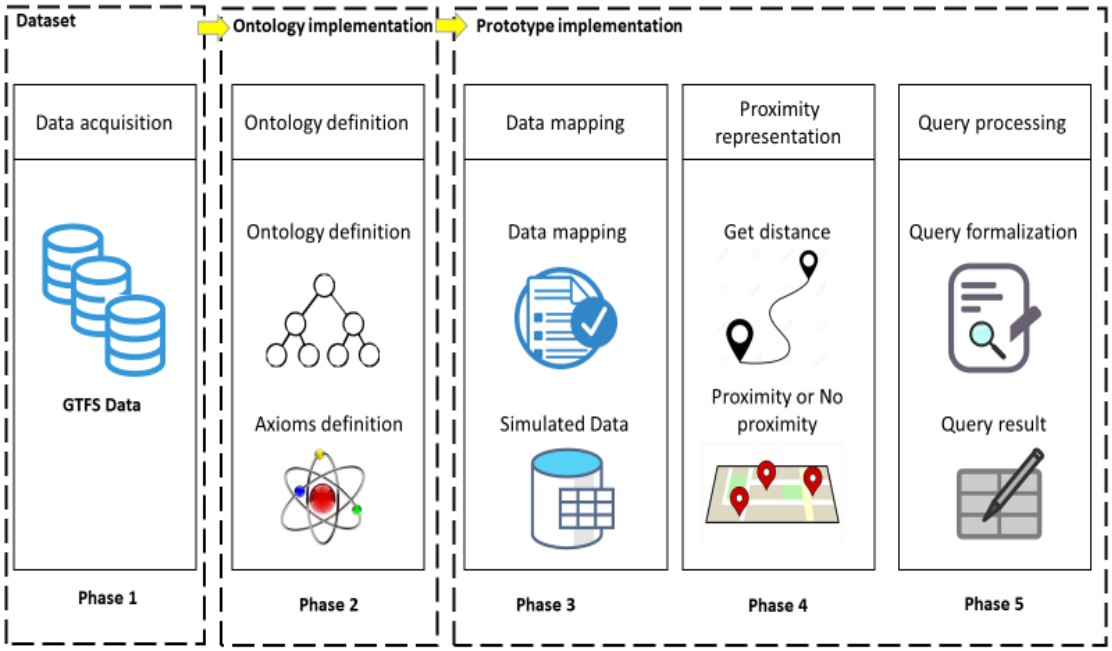



**FIGURE 6:** Main Phases of Implementing Smart Proximity Tool.

In what follows, we describe the implementation details of each phase of our proposed approach.

**Data Acquisition**

As stated earlier, we acquired data about transportation in the city of San Francisco (California, USA) shared using the format of GTFS data. Data available for the city of San Francisco are collected in several CSV files [24]. Data are collected into four files: Trips, Routes, Shapes, Stops and Stop times.

**Smart Proximity Ontology**

We start the ontology definition by defining the classes. We focus on the concepts that are identified and related to the GTFS data and our intended use cases. The defined entities are transportations, hubs, traffic signals, person, disability and proximity, which are defined as the main classes. Transportations include cable car, rail, bus and taxi as subclasses.

We import the point concept from an existing ontology that is referred to as the WGS84 Geo ontology, which provides the basic vocabulary for representing data with latitudes and longitudes [40]. The point concept is related to each entity and instance in our ontology.

We define two object properties in our ontology: hasDisability and hasLocation. The property hasDisability assigns a disability type to a disabled person, while hasLocation assigns a location of entities to the latitude and longitude values in our ontology.

We continue the ontology definition by defining two data properties: hasDistance and hasRouteName. The data property hasDistance allows double values to be assigned to individuals of proximity entity, while the hasRouteName property assigns all transportations with the route names of the trips that they conducted.

The resulting class hierarchy cannot provide the intended new knowledge for our use cases. Thus, we define axioms to acquire the new knowledge for the use cases. These axioms and the reasoning functionalities will produce new useful knowledge. We implement three use cases of

our four use cases: Identifying transportations near hubs without people, identifying traffic signals with people with disabilities and Identifying trains with at least one taxi at a nearby station.

**Smart Proximity Tool**

The tool implementation consists of three main steps: Data processing and mapping, proximity representation and querying.

To perform the mapping, a user must select a time to populate the ontology with GTFS data. We obtain all trips that are active during the selected time. We identify the location of all transportations of the active trips according to the selected time. The location of the transportation is achieved by the setLocationAtGivenTime function.

After executing this function, all transportation will have a specific location with latitude and longitude values associated with it. We import the remaining files according to the selected time and randomly generate people and taxis. A sample code for adding random people around traffic signals is presented in this section.

As stated in section 3, our representation of the proximity starts by calculating the distance between two points, which is achieved via the getDistance function. The formula used in this function is obtained from GeoDataSource [43]. We check if the distance is below the threshold. If it is below the defined threshold, then add the individual of the type proximity.

Once the axioms presented are annotated in the ontology, the information that we need can be extracted from the ontology using a query mechanism that involves DL reasoning as a DL query or a SPARQL-DL query. We consider the latter technology.

After executing the query, we display it result in a table. Due to the SPARQL-DL problem of not following the order of the variables within the query expression, we reorder the result and display it in a table with its specified order by using the following code:

Moving to the user interface of the tool, we implement seven frames as interfaces to be shown to the user. Each frame performs a specific function, and we attempted to make them as user-friendly as possible. The interfaces capture the following tool procedures:

1. Selection of the ontology that contains concepts.
2. Selection of the GTFS data to be acquired.
3. Selection of a time and date for the acquisition of GTFS data.
4. GTFS data are acquired, other data is simulated, and an ontology is generated.
5. The user is asked if she/he wants to execute a query from a pre-defined list or a free query.

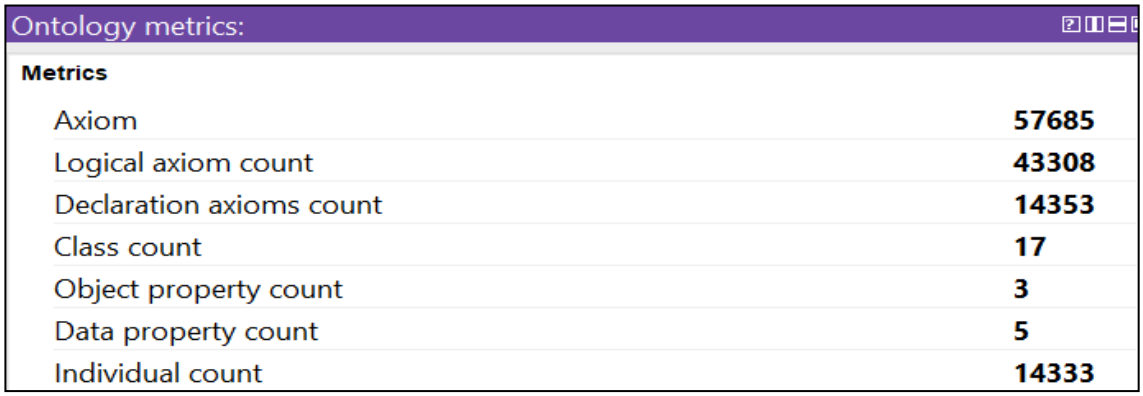



**FIGURE 7:** Ontology Metrics from Protégé.

The relative window is opened; when the user presses “Execute query”, the SPARQL-DL query is executed; and the results of the query are shown. To provide an overview of the number of classes, individuals, axioms and other metrics, we measure these metrics after populating the ontology with GTFS data. Figure 7 shows some of our ontology metrics.

To check the capability of our tool, some scenarios are executed to establish and conclude some important remarks for running the tool. The following scenarios are explained. We start by executing the four use cases with different times for mapping the data. The total measurements results are presented in Table 3.

| Generating ontology parameters | | Execution time | | | |
|---|---|---|---|---|---|
| Time | Date | Query 1 | Query 2 | Query 3 | Query 4 |
| 1:00 | 2019/03/04 | 5 seconds | 1 second (No result) | 11 minutes | 1 second (No result) |
| 1:00 | 2019/10/15 | 3 seconds | 1 second (No result) | 11 minutes | 1 second (No result) |
| 1:00 | 2019/01/29 | 2 seconds | 1 second (No result) | 10 minuets | 1 second (No result) |
| 7:30 | 2019/03/04 | 3 seconds | 3 seconds | 14 minutes | 18 minutes |
| 7:30 | 2019/10/15 | 2 seconds | 1 second | 4 minutes | 5 minutes |

**TABLE 3:** Measurements of Queries Execution Time.

From Table 3, we conclude that the first and second queries require seconds to execute. The third and fourth queries require minutes in the case where they produce results. Thus, we can classify the third and fourth queries as long execution time queries. While we perform these measurements, we noticed that no result is returned from the query in some situations. No results mean that no proximity was observed among the entities specified in the query within the selected time. We also investigate the cases in which the ontology was generated. To successfully generate the ontology, we must ensure that the IRI of the ontology is correct and works. We must enter the time and date for generating the ontology in the correct format.

## 5. CONCLUSION AND FUTURE WORK

We present an ontology for mapping data and generating new knowledge from an existing ontology, which can be useful to smart devices in a city for understanding their surrounding environment and making decisions. This study is the first attempt to manage proximity information among entities in a city to generate new knowledge.

We demonstrated that new and useful information can be generated with the definition of proper OWL axioms in the ontology. We identified and mapped into an ontology the locations of different modes of transportation, which are represented in open GTFS data, in the city of San Francisco. We virtually represented the presence of people, taxis, and traffic signals in the city; this information is unavailable otherwise.

Addressing the proximity between two smart city objects was driven by the gap in the literature, where no smart city ontology has represented the proximity concept and generated new knowledge starting from this representation. This approach was also driven by the benefits and useful knowledge, which can be gained from the proximity awareness between two smart city objects.

Available knowledge is limited to knowledge that can be represented as a collection of OWL propositions in an ontology, while the potential knowledge that can be inferred is bound by the limits of expressivity of OWL axioms. The computational requests of a DL reasoner do not enable

the generation of knowledge from many OWL axioms. However, we consider that this limit does not represent a problem for the usage of the approach in real-time environments.

A future challenge is to collect raw city data in real time and make it available to a smart device. Although we built a large ontology that represents an entire city in the current research, a device only needs information about its surrounding environment. Thus, the information that has to be made available to a device can be filtered a priori to include the minimal necessary knowledge in the picture of the environment, as represented by the OWL ontology. This minimal necessary knowledge is the representation of the individuals who are near the considered device. The representation of a device's surroundings usually does not consist of a large amount of data: entities near a single device can vary from 0 to even 1000; however, these are amounts can be easily managed by a DL reasoner, as we learned from our experiments. Thus, a device can use both the available (and always updatable) conceptualization and the minimal received information to generate new knowledge and make decisions in real time.

Different and interesting future work is represented by the chance to add more useful information from other sources, such as weather stations, urban sensors, and human wearable devices. Several improvements to the user interface of the current tool provides the chance to upload a custom GTFS data source and a more user-friendly interface that can show the content of the ontology without requiring the user to rely on external viewers, such as Protégé. Additional query languages, different from SPARQL-DL, such as SPARQL or DL query, can be supported.

## 6. REFERENCES

[1] M. Batty, "Big data, smart cities and city planning," Dialogues Hum. Geogr., vol. 3, no. 3, pp. 274–279, 2013.

[2] S. Allwinkle and P. Cruickshank, "Creating smart-er cities: An overview," J. urban Technol., vol. 18, no. 2, pp. 1–16, 2011.

[3] G. C. Lazaroiu and M. Roscia, "Definition methodology for the smart cities model," Energy, vol. 47, no. 1, pp. 326–332, 2012.

[4] M. Deakin and H. Al Waer, "From intelligent to smart cities," Intell. Build. Int., vol. 3, no. 3, pp. 140–152, 2011.

[5] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of things for smart cities," IEEE Internet Things J., vol. 1, no. 1, pp. 22–32, 2014.

[6] R. Kitchin, "Making sense of smart cities: addressing present shortcomings," Cambridge J. Reg. Econ. Soc., vol. 8, no. 1, pp. 131–136, 2015.

[7] C. Manville et al., "Mapping smart cities in the EU," 2014.

[8] G. Hancke, B. Silva, and G. Hancke Jr, "The role of advanced sensing in smart cities," Sensors, vol. 13, no. 1, pp. 393–425, 2013.

[9] M. Abdel-Basset and M. Mohamed, "The role of single valued neutrosophic sets and rough sets in smart city: Imperfect and incomplete information systems," Measurement, vol. 124, pp. 47–55, 2018.

[10] A. Caragliu and C. F. Del Bo, "Smart innovative cities: The impact of Smart City policies on urban innovation," Technol. Forecast. Soc. Change, vol. 142, pp. 373–383, 2019.

[11] D. M. Mejia, Integration of Heterogeneous Traffic Data to Address Mobility Challenges in the City of El Paso. The University of Texas at El Paso, 2017.

[12] A. Caragliu, C. Del Bo, and P. Nijkamp, "Smart cities in Europe," J. urban Technol., vol. 18,

no. 2, pp. 65–82, 2011.

[13] M. Batty et al., “Smart cities of the future,” Eur. Phys. J. Spec. Top., vol. 214, no. 1, pp. 481–518, 2012.

[14] V. Fernandez-Anez, J. M. Fernández-Güell, and R. Giffinger, “Smart City implementation and discourses: An integrated conceptual model. The case of Vienna,” Cities, vol. 78, pp. 4–16, 2018.

[15] L. Anthopoulos, P. Fitsilis, and C. Ziozias, “What is the Source of Smart City Value?: A Business Model Analysis,” in Smart Cities and Smart Spaces: Concepts, Methodologies, Tools, and Applications, IGI Global, 2019, pp. 56–77.

[16] R. Giffinger and H. Gudrun, “Smart cities ranking: an effective instrument for the positioning of the cities?,” ACE Archit. city Environ., vol. 4, no. 12, pp. 7–26, 2010.

[17] M. Angelidou, “Smart cities: A conjuncture of four forces,” Cities, vol. 47, pp. 95–106, 2015.

[18] A. M. Townsend, Smart cities: Big data, civic hackers, and the quest for a new utopia. WW Norton & Company, 2013.

[19] P. Espinoza-Arias, M. Poveda-Villalón, R. García-Castro, and O. Corcho, “Ontological Representation of Smart City Data: From Devices to Cities,” Appl. Sci., vol. 9, no. 1, p. 32, 2019.

[20] V. Albino, U. Berardi, and R. M. Dangelico, “Smart cities: Definitions, dimensions, performance, and initiatives,” J. urban Technol., vol. 22, no. 1, pp. 3–21, 2015.

[21] M. Katsumi and M. Fox, “Ontologies for transportation research: A survey,” Transp. Res. Part C Emerg. Technol., vol. 89, pp. 53–82, 2018.

[22] B. Lorenz, H. J. Ohlbach, and L. Yang, “Ontology of transportation networks,” 2005.

[23] Z. Xiao, H. B. Lim, and L. Ponnambalam, “Participatory sensing for smart cities: A case study on transport trip quality measurement,” IEEE Trans. Ind. Informatics, vol. 13, no. 2, pp. 759–770, 2017.

[24] “GTFS Transit Data,” 2019. [Online]. Available: https://www.sfmta.com/reports/gtfs-transit-data. [Accessed: 19-Apr-2019].

[25] K. M. De Oliveira, F. Bacha, H. Mnasser, and M. Abed, “Transportation ontology definition and application for the content personalization of user interfaces,” Expert Syst. Appl., vol. 40, no. 8, pp. 3145–3159, 2013.

[26] A. J. Bermejo, J. Villadangos, J. J. Astrain, and A. Cordoba, “Ontology based road traffic management,” in Intelligent Distributed Computing VI, Springer, 2013, pp. 103–108.

[27] M. Codescu, G. Horsinka, O. Kutz, T. Mossakowski, and R. Rau, “Osmonto-an ontology of openstreetmap tags,” State map Eur., vol. 2011, 2011.

[28] N. Anand, M. Yang, J. H. R. Van Duin, and L. Tavasszy, “GenCLOn: An ontology for city logistics,” Expert Syst. Appl., vol. 39, no. 15, pp. 11944–11960, 2012.

[29] N. Hill, G. Gibson, E. Guidorzi, S. Amaral, A. Parlikad, and Y. Jin, “Scoping Study into Deriving Transport Benefits from Big Data and the Internet of Things in Smart Cities,” 2017.

[30] P. Rode et al., “Accessibility in cities: transport and urban form,” in Disrupting mobility, Springer, 2017, pp. 239–273.

[31] K. Farkas, G. Feher, A. Benczur, and C. Sidlo, “Crowdsensing based public transport information service in smart cities,” IEEE Commun. Mag., vol. 53, no. 8, pp. 158–165, 2015.

[32] C. Navarro, M. Roca-Riu, S. Furió, and M. Estrada, “Designing new models for energy efficiency in urban freight transport for smart cities and its application to the Spanish case,” Transp. Res. Procedia, vol. 12, pp. 314–324, 2016.

[33] C. F. Calvillo, A. Sánchez-Miralles, and J. Villar, “Energy management and planning in smart cities,” Renew. Sustain. Energy Rev., vol. 55, pp. 273–287, 2016.

[34] C. Lim, K. Kim, and P. P. Maglio, “Smart cities with big data : Reference models , challenges , and considerations,” Cities, no. February, pp. 1–14, 2018.

[35] R. Khatoun and S. Zeadally, “Smart cities: concepts, architectures, research opportunities.,” Commun. Acm, vol. 59, no. 8, pp. 46–57, 2016.

[36] F. Behrendt, “Why cycling matters for smart cities. Internet of bicycles for intelligent transport,” J. Transp. Geogr., vol. 56, pp. 157–164, 2016.

[37] F. Bifulco, M. Tregua, C. C. Amitrano, and A. D’Auria, “ICT and sustainability in smart cities management,” Int. J. Public Sect. Manag., vol. 29, no. 2, pp. 132–147, 2016.

[38] P. Cardullo and R. Kitchin, “Being a ‘citizen’in the smart city: up and down the scaffold of smart citizen participation in Dublin, Ireland,” GeoJournal, vol. 84, no. 1, pp. 1–13, 2019.

[39] C. J. Martin, J. Evans, and A. Karvonen, “Smart and sustainable? Five tensions in the visions and practices of the smart-sustainable city in Europe and North America,” Technol. Forecast. Soc. Change, vol. 133, pp. 269–278, 2018.

[40] “W3C Semantic Web Interest Group: Basic Geo (WGS84 lat/long) Vocabulary.” [Online]. Available: https://www.w3.org/2003/01/geo/. [Accessed: 20-Apr-2019].

[41] S. Roche, “Geographic information science II: Less space, more places in smart cities,” Prog. Hum. Geogr., vol. 40, no. 4, pp. 565–573, 2016.

[42] E. Sirin and B. Parsia, “SPARQL-DL: SPARQL Query for OWL-DL.,” in OWLED, 2007, vol. 258.

[43] “Calculate Distance by Latitude and Longitude using Java | GeoDataSource.” [Online]. Available: http://www.geodatasource.com/developers/java. [Accessed: 19-Apr-2019].

# INSTRUCTIONS TO CONTRIBUTORS

The main aim of International Journal of Artificial Intelligence and Expert Systems (IJAE) is to provide a platform to AI & Expert Systems (ES) scientists and professionals to share their research and report new advances in the field of AI and ES. IJAE is a refereed journal producing well-written original research articles and studies, high quality papers as well as state-of-the-art surveys related to AI and ES. By establishing an effective channel of communication between theoretical researchers and practitioners, IJAE provides necessary support to practitioners in the design and development of intelligent and expert systems, and the difficulties faced by the practitioners in using the theoretical results provide feedback to the theoreticians to revalidate their models. IJAE thus meets the demand of both theoretical and applied researchers in artificial intelligence, soft computing and expert systems.

IJAE is a broad journal covering all branches of Artificial Intelligence and Expert Systems and its application in the topics including but not limited to technology & computing, fuzzy logic, expert systems, neural networks, reasoning and evolution, automatic control, mechatronics, robotics, web intelligence applications, heuristic and AI planning strategies and tools, computational theories of learning, intelligent system architectures.

To build its International reputation, we are disseminating the publication information through Google Books, Google Scholar, Open J Gate, ScientificCommons, Docstoc and many more. Our International Editors are working on establishing ISI listing and a good impact factor for IJAE.

The initial efforts helped to shape the editorial policy and to sharpen the focus of the journal. Starting with Volume 9, 2020, IJAE will be appearing with more focused issues related to artificial intelligence and expert systems studies. Besides normal publications, IJAE intend to organized special issues on more focused topics. Each special issue will have a designated editor (editors) – either member of the editorial board or another recognized specialist in the respective field.

We are open to contributions, proposals for any topic as well as for editors and reviewers. We understand that it is through the effort of volunteers that CSC Journals continues to grow and flourish.

**LIST OF TOPICS**

The realm of International Journal of Artificial Intelligence and Expert Systems (IJAE) extends, but not limited, to the following:

- AI for Web Intelligence Applications
- AI Parallel Processing Tools
- AI Tools for Computer Vision and Speech Understand
- Application in VLSI Algorithms and Mobile Communication
- Case-based reasoning
- Derivative-free Optimization Algorithms
- Evolutionary and Swarm Algorithms
- Expert Systems Components
- Fuzzy Sets and logic
- Hybridization of Intelligent Models/algorithms
- Inference
- AI in Bioinformatics
- AI Tools for CAD and VLSI Analysis/Design/Testing
- AI Tools for Multimedia
- Automated Reasoning
- Data and Web Mining
- Emotional Intelligence
- Expert System Development Stages
- Expert-System Development Lifecycle
- Heuristic and AI Planning Strategies and Tools
- Image Understanding
- Integrated/Hybrid AI Approaches

- Intelligent Planning
- Intelligent System Architectures
- Knowledge-Based Systems
- Logic Programming
- Multi-agent Systems
- Neural Networks for AI
- Parallel and Distributed Realization of Intelligence
- Reasoning and Evolution of Knowledge Bases
- Rule-Based Systems
- Uncertainty
- Intelligent Search
- Knowledge Acquisition
- Knowledge-Based/Expert Systems
- Machine learning
- Neural Computing
- Object-Oriented Programming for AI
- Problem solving Methods
- Rough Sets
- Self-Healing and Autonomous Systems
- Visual/linguistic Perception

**CALL FOR PAPERS**

**Volume: 9** - **Issue: 1**

**i. Submission Deadline :** December 31, 2019

**ii. Author Notification:** January 31, 2020

**iii. Issue Publication:** February 2020

## CONTACT INFORMATION

**Computer Science Journals Sdn BhD**

B-5-8 Plaza Mont Kiara, Mont Kiara

50480, Kuala Lumpur, MALAYSIA

Phone: 006 03 6204 5627

Fax: 006 03 6204 5628

Email: cscpress@cscjournals.org

CSC PUBLISHERS © 2019
COMPUTER SCIENCE JOURNALS SDN BHD
B-5-8 PLAZA MONT KIARA
MONT KIARA
50480, KUALA LUMPUR
MALAYSIA

PHONE: 006 03 6204 5627
FAX: 006 03 6204 5628
EMAIL: cscpress@cscjournals.org