MonuMAI: Dataset, deep learning pipeline and citizen science based app for monumental heritage taxonomy and classification

Alberto Lamas[1], Siham Tabik[1], Policarpo Cruz[2], Rosana Montes[1], Álvaro Martínez-Sevilla[1], Teresa Cruz[3], and Francisco Herrera[1]

[1]Andalusian Research Institute in Data Science and Computational Intelligence, University of Granada, 18071 Granada, Spain
[2]Art History Department, University of Granada, 18071 Granada, Spain
[3]Descubre Foundation, 18016 Granada, Spain.
email: albertocl@decsai.ugr.es, siham@ugr.es, jcruz@ugr.es, asevilla@ugr.es, rosana@ugr.es, teresa.cruz@fundaciondescubre.es, herrera@decsai.ugr.es

September 1, 2020

Abstract

An important part of art history can be discovered through the visual information in monument facades. However, the analysis of this visual information, i.e, morphology and architectural elements, requires high expert knowledge. An automatic system for identifying the architectural style or detecting the architectural elements of a monument based on one image will certainly help improving our knowledge in art and history. Building such tool is challenging as some styles share architectural elements, the bad conservation state of some monuments and the noise included in the image itself. The aim of this paper is to introduce MonuMAI (Monument with Mathematics and Artificial Intelligence) framework. In particular, (i) we designed MonuMAI dataset rich with expert knowledge considering the proposed architectural styles taxonomy and key elements relationship, which allows addressing several tasks, e.g., monument style classification and architectural elements detection, (ii) we developed MonuMAI deep learning pipeline based on lightweight MonuNet architecture for monument style classification and MonuMAI Key Elements Detection (MonuMAI-KED) model, and (iii) we built citizen science based MonuMAI mobile app that uses the proposed MonuMAI deep learning pipeline trained on MonuMAI dataset for performing in real life conditions. Our experiments show that both MonuNet architecture and the detection model achieve very good results under real life conditions.

Index terms— Convolutional Neural Networks, Architectural information extraction, architectural style classification, MonuMAI, monumental heritage.

1 Introduction

The knowledge of the architectural style and elements of monumental heritage is of paramount importance in several fields, in art, art history, tourism [1], architecture, geometry, academia and culture [2]. However, recognizing the style or facade elements of historical monuments is reachable only to very few experts.

Machine learning based automatic systems [18, 24] have been barely used to address such task due to the difficulty in obtaining expert knowledge. Indeed, there is neither an established monumental heritage taxonomy for architectural styles and elements, nor automatic tools to help identifying the style or architectural elements of a monument based on the image of its facade.

Most previous works on architectural style identification combine traditional machine learning with computer vision techniques [19, 28, 3], which implies a high level of human supervision and produces models with a reduced generalization capacity [18, 24, 27]. For example, the authors in [24, 27] first use Deformable Part-based Model (DPM) with histogram of oriented gradients (HOG) for features extraction then apply support vector machine (SVM) for classification. Similarly, the authors in [18] addressed the architectural style classification of three styles, Romanesque, Gothic and Baroque by first extracting features with Scale-Invariant Feature Transform (SIFT) then using SVM for classification.

The authors in [14] used state-of-the-art deep learning (DL) models for the classification of architectural elements such as columns, bell tower, apse, or altar among others. The proposed approach does not identify the architectural style in the input image and can not be used under real life conditions as it requires manual intervention.

The above approaches provide good accuracy but suffer from strong constraints: i) unstructured approaches that are difficult to scale to more styles or elements [3, 27], ii) lack of robustness to different perspectives in pictures [19], and/or iii) unable to perform in real-life conditions [14, 28, 24], i.e., presence of other elements such as trees, people, traffic signals in the input image..

The recent progress in DL networks [8, 15] in general and Convolutional Neural Networks (CNNs) [7] in particular has made several breakthrough in many computer vision tasks, in image classification [22], object detection in images [2, 11, 12] and semantic segmentation [9, 13, 26]. CNNs present a high potential for architectural feature extraction in images [25]. Their ability to learn textures, patterns and colors make them suitable for analytical vision tasks, such as the architectural style classification and the detection of architectural elements in a monument facade image.

Nevertheless, identifying the style or detecting architectural elements of a given monumental heritage facade based on an image of its facade is challenging due to the next reasons:

- Some styles may share the same key characteristic architectural elements.
- The facade may contain external elements to the main style because they were

[1]UNESCO heritage tourism program: whc.unesco.org/uploads/activities/documents/activity-669-7
[2]UNESCO heritage education program: whc.unesco.org/en/wheducation

appended during restoration process or during their very long construction process.

- The conservation state of some monuments can make the identification process more difficult.
- Noise, darkness and shadow in the image itself may hide some of the characteristic elements of the style.
- The image perspective can deform some similar geometric shapes, e.g. arches.

To build an automatic architectural style classification model and a key architectural element detection model, it is essential to create a dataset that takes into account all the aforementioned challenges. Accordingly, this work addresses the problem by means of MonuMAI (Monument with Mathematics and Artificial Intelligence) framework considering three main axis. Firstly, to design MonuMAI dataset, an annotated monument image dataset based on a new monumental heritage taxonomy of architectural key elements and styles. Secondly, to develop MonuMAI deep learning pipeline for identifying architectural styles or detecting architectural elements in monument images. Thirdly, to provide citizen science based MonuMAI app which integrates both models for performing in real life conditions.

In particular, the contributions of this work can be summarized as follows:

- To design an monumental heritage image dataset taking into account that:
  - We define the first tree taxonomy that captures the relationship between four of the most distinguishable architectural styles in Europe, Hispanic-Muslim, Gothic, Renaissance, and Baroque, and the key elements that characterize each style. The monumental heritage taxonomy is defined under a close collaboration with three art experts.
  - We design a multi-task MonuMAI dataset, a quality multi-task monument dataset with two types of annotation which make it suitable not only for classification and detection tasks but also for analyzing explanation methods.
- To develop MonuMAI deep learning pipeline considering that:
  - We develop a lightweight deep CNN called MonuNet architecture, for monument style classification. MonuNet architecture achieves a good balance between the number of layers and trainable parameters along with a low computational cost that is fundamental on low performance devices such as smartphones.
  - We propose a key architectural element detection model, called MonuMAI-KED model, based on Faster R-CNN [17] object detection mode and ResNet-101 [5] CNN-based model. The detected elements are horseshoe arch, lobed arch, flat arch, pointed arch, ogee arch, trefoil arch, gothic pinnacle, porthole, lintelled doorway, rounded arch, triangular pediment (or pointed pediment), segmental pediment, serliana, broken pediment, and solomonic column.

- To provide MonuMAI app [3], a free tool for the automatic extraction of architectural information through the architectural style classification and key elements detection. This app is available for smartphone and via web, as a citizen science based MonuMAI app it helps spreading art knowledge and collecting images to further increase MonuMAI dataset.

Our experiments show that both MonuNet architecture and the key architectural element detection models achieve very good results including in real-life conditions.

This paper is organized as follows. Section 2 provides the new taxonomy based on architectural styles and elements. Section 3 describes MonuMAI image dataset, a new two-fold dataset for classification and detection. Section 4 shows the process for finding the optimal lightweight model for architectural style classification. Section 5 presents MonuMAI app, a mobile application for the general public. Section 6 gives the experimental setup and results that show the suitability of DL models on the new MonuMAI dataset. Lastly, Section 7 gives the final conclusions and future works.

2 Monumental heritage taxonomy

In each period of history, man has adopted a different way of understanding architecture depending on socio-cultural and technical considerations. Each architectural style uses a series of recognizable elements from the appearance and variations of basic elements, either of structural nature (supports, covering system) or ornamental nature (decorative types).

In general, humans determine the class of an object by first identifying the key elements in that object. Based on this intuition, our approach is based on the way an expert identifies the style of a monument. We propose a new tree taxonomy that defines the relationships of monumental heritage between the different styles and their characteristic elements.

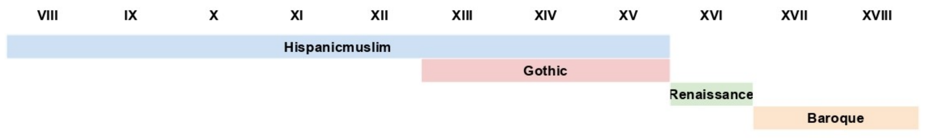


Figure 1: The evolution of the four considered styles over the last eleven centuries

We consider four of the most distinctive architectural styles in Europe, Hispanic-Muslim, Gothic, Renaissance, and Baroque, from the period between VIII and XVIII centuries (Figure 1). These styles cover most of the middle Ages (V-XV century) and Modern Ages (XV-XVIII century).

The architectural elements of monuments cover three types: brackets, ornaments and archs. Hispanic-Muslim style is characterized by horseshoe, lobed arches and flat arch ornament. Gothic style is characterized by pointed, ogee, trefoil arches and Gothic pinnacle. Renaissance style is characterized by porthole, lintelled doorway, rounded arches, triangular pediment, segmental pediment ornaments, and serliana bracket. Baroque

[3]monumai.ugr.es

style is characterized by broken pediment ornament, solomonic column bracket, porthole, lintelled doorway, and rounded arches. Figure 2 illustrates the characteristic elements of each style.

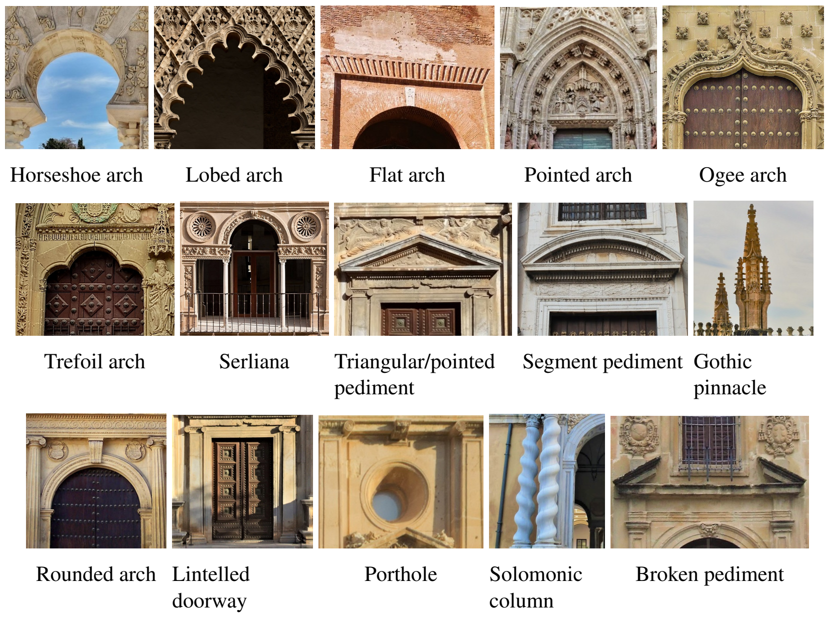


Figure 2: Fifteen architectural key elements considered in this work to identify the Hispanic-Muslim, Gothic, Renaissance, and Baroque styles.

Based on this information, we built a tree-structured taxonomy, as shown in Figure 3, in which the first level nodes are the styles and the second level nodes are the characteristic elements of each style. Lower levels (tree leaves) represent variations of some key element.

The main elements of each style have been selected, but there are inevitable limitations and overlaps. On the one hand, architectural styles are not independent, they have a beginning, an evolution and end, interacting with the preceding and following style periods. On the other hand, some elements are used in more than one style and hence for more accurate recognition it is necessary to detect and display them together.

3 MonuMAI: a multi-task monument image dataset

We built MonuMAI dataset, a new public dataset [4], rich in expert knowledge. It was annotated by three experts using two types of annotations which makes it useful for several tasks, for monument style classification (Section 3.1), for key elements detection (Section 3.2), and other potential applications.

[4]dasci.es/transferencia/dascii-hub/open-data/monumai-open-data

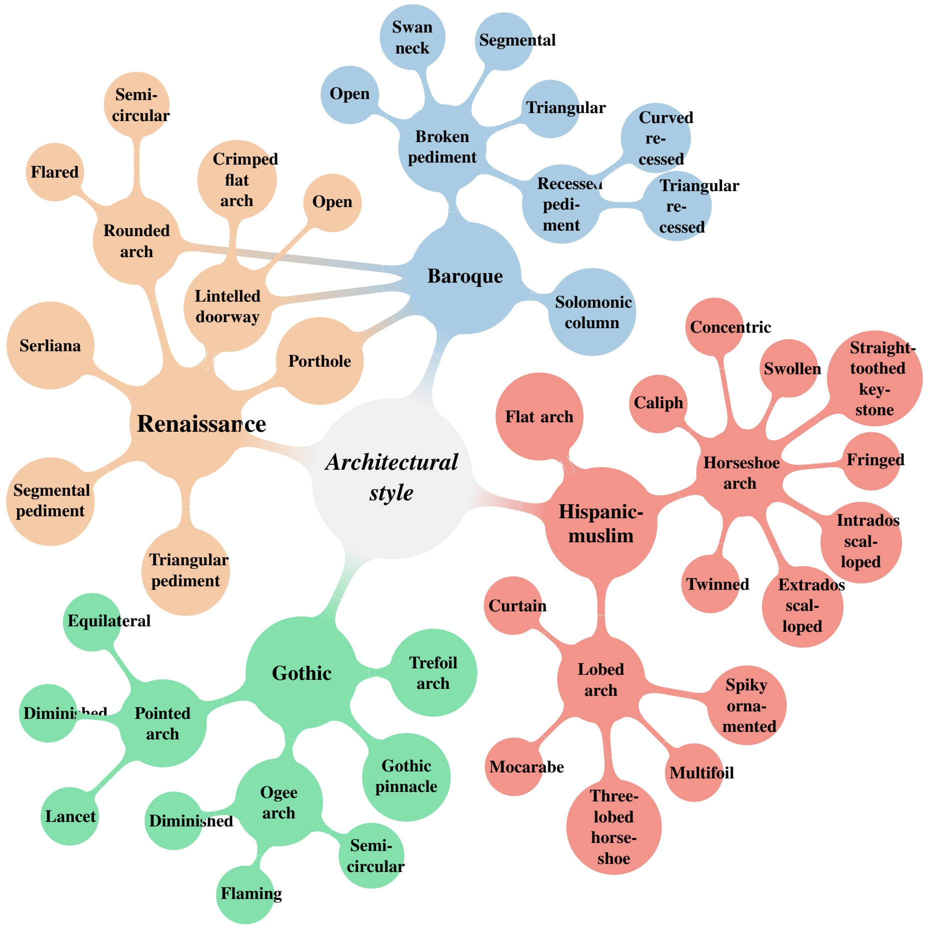


Figure 3: A tree-structured taxonomy of the four considered styles (second level nodes), their characteristic elements (third level nodes), and subtypes of every element (tree leaves).

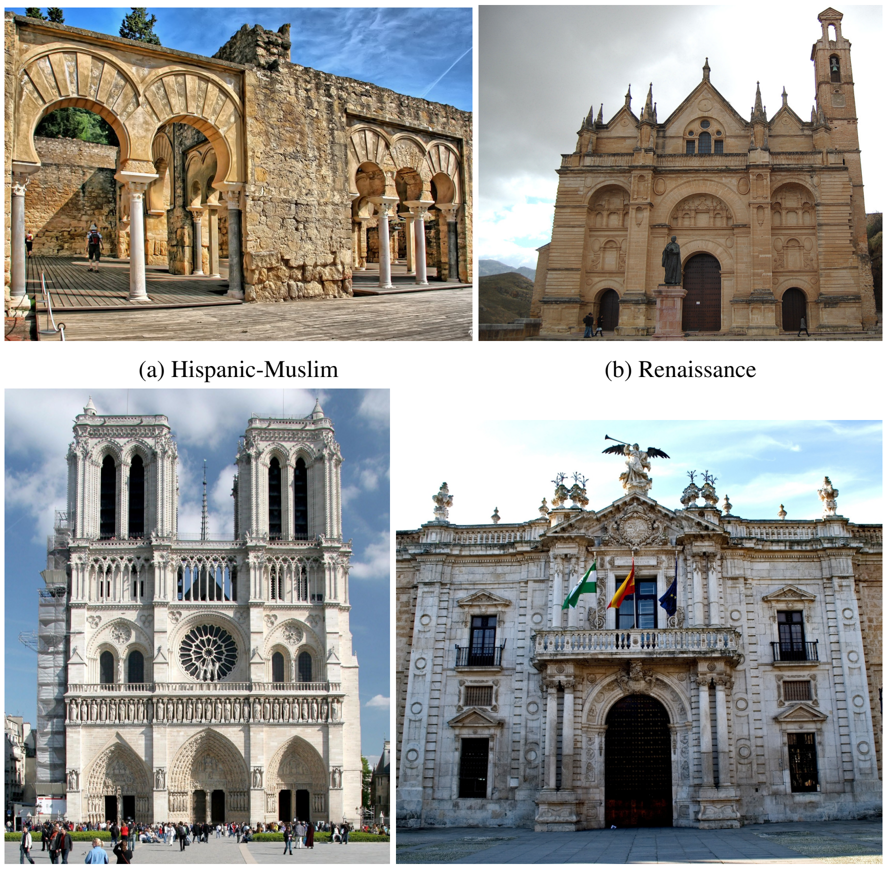
3.1 MonuMAI dataset for architectural style classification

In practice, architectural styles cannot be found in their pure state for two reasons:

- Some styles such as renaissance and baroque share common characteristic elements.
- Over time monuments may suffer alterations by including elements from other styles.

Hence, in a recent image of a monument that was built in the ancient world period, we cannot predict a pure style but rather the dominant style.

MonuMAI dataset contains 1514 RGB images of monument facades of four architectural styles Hispanic-Muslim, Gothic, Renaissance, and Baroque, illustrated in Figure 4. A summary of the characteristics of MonuMAI dataset is shown in Table 1.



(a) Hispanic-Muslim (b) Renaissance

(c) Gothic (d) Baroque

Figure 4: Illustration of the four considered architectural styles. (a) Medina Azahara (Córdoba, Spain 936) represents the Hispanic-Muslim style, (b) the Royal collegiate church of Santa María la Mayor (Antequera, Spain 1514) represents Renaissance style, (c) the Cathedral of Notre Dame (París, France 1163) represents the Gothic style and the Royal Factory of tobacco (Sevilla, Spain 1728) represents the Baroque style.

Architectural style	**#images**	**ratio** (%)
Hispanic-Muslim	327	21.6
Gothic	359	23.7
Renaissance	312	20.6
Baroque	516	34.1
	1514	

Table 1: Characteristics of MonuMAI architectural style classification dataset (#images represents the number of images.)

The used images for building MonuMAI dataset were taken so that the monument is centered and fills most of the image. These high quality images were selected from Internet and taken by smartphone cameras thanks to the educational MonuMAI app developed by the same authors of this paper under a citizen science project.

The first annotation is actually a label of the dominant architectural style in the image, Hispanic-Muslim, Gothic, Renaissance, and Baroque. This annotation makes MonuMAI dataset useful for the task of architectural style classification.

3.2 MonuMAI dataset for key architectural elements detection

Besides the label that indicates the monument style, additional expert knowledge about the existent key elements in each image is provided. The area where each characteristic element is located in the image is delimited using a bounding box and labeled using the name of one of the fifteen key elements, horseshoe arch, lobed arch, flat arch, pointed arch, ogee arch, trefoil arch, Gothic pinnacle, triangular pediment, segmental pediment, serliana, porthole, lintelled doorway, rounded arch, broken pediment, and solomonic column. The elements that have more than 60% of their pixels occluded are not included. The architectural elements annotation is illustrated in Figure 5. This type of annotation makes MonuMAI dataset suitable for the architectural elements detection task.

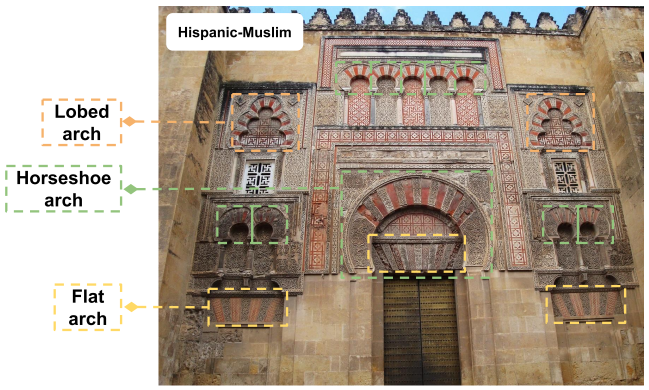


Figure 5: Labeling framework. Architectural elements annotation (lobed arch, horseshoe arch and flat arch) provides element name and location.

MonuMAI dataset includes 6650 annotated key elements distributed into the fifteen architectural element classes. A statistical analysis of these elements is shown in Table 2.

Architectural element	**Count**	**Element rate**	**Repetition ratio**	**Architectural style**
Horseshoe arch	640	9.62%	2.09	Hispanic-muslim
Lobed arch	148	2.23%	3.44	Hispanic-muslim
Flat arch	161	2.42%	1.42	Hispanic-muslim
Pointed arch	495	7.44%	2.03	Gothic
Ogee arch	238	3.58%	1.58	Gothic
Trefoil arch	57	0.87%	1.36	Gothic
Gothic pinnacle	346	5.20%	4.81	Gothic
Triangular pediment	743	11.17%	2.07	Renaissance
Segmental pediment	258	3.88%	1.91	Renaissance
Serliana	77	1.16%	1.13	Renaissance
Porthole	392	5.89%	3.27	Renaissance/Baroque
Lintelled doorway	1150	17.29%	2.73	Renaissance/Baroque
Rounded arch	1044	15.70%	1.87	Renaissance/Baroque
Broken pediment	604	9.08%	1.41	Baroque
Solomonic column	297	4.47%	3.67	Baroque

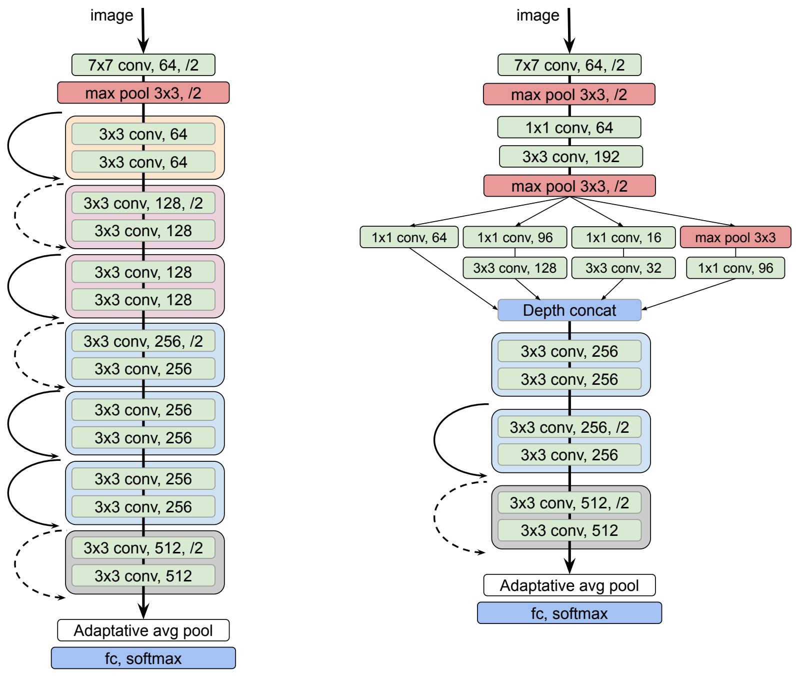
Table 2: Characteristics of the architectural styles dataset, where count is the number of occurrences of an element in the dataset, element rate is the ratio between the number of occurrences of an element and the total number of all elements, repetition rate is the average number of occurrences of an element per image, only in the images where it occurs at least once.

By analyzing the key architectural elements in images of MonuMAI dataset, we found an unbalanced number of elements classes. For example, elements such as Trefoil arch or Serliana represents only 0.87% and 1.16% of total elements respectively. In contrast, elements such as Triangular pediment, Rounded arch, or Lintelled doorway has higher rates, represent 11.17%, 15.7%, and 17.29% of total elements respectively. This unbalance makes the detection of some characteristic elements more challenging.

It is noteworthy the high repetition rate of several elements. Some are used as a building component which represents part of the structure such as Porthole or Lintelled doorway with 3.27 and 2.73 element repetition rates respectively. However, other frequent elements are decorative but important for an architectural style such as lobed arch to Hispanic-muslim, or Solomonic column to Baroque, with 3.44 and 3.67 elements per image respectively. Accordingly, a high element repetition rate do not always set the predominant architectural style of a monument.

4 MonuNet: optimal lightweight architectural style classification network

State-of-the-art architectures such as, the latest Inception, DenseNet and ResNet, provide good performance with less efforts. However, such models contain a large volume of weights and require high computational load which make them prohibitive for small devices, e.g. smartphones and tablets. Our design of the optimal network will be guided by finding a good balance between accuracy, robustness and computation requirements. Our objective is to design a lightweight network that is as good as heavy state-of-the-art models but more suitable for small devices.



(a) Preliminary architecture (b) MonuNet

Figure 6: MonuNet architecture construction process, (a) a preliminary net built by fully stacking basic blocks and residual connections with boosted middle level of abstraction, and (b) the best lightweight net, a combination of Inception-style for the initial level with residual network.

We built a deep CNN, called MonuNet, appropriate for architectural style classification and devices with limited computational resources. We selected ResNet-18 as starting point, which is built by stacking the same basic building block [5]. First, we evaluated the relevance of the different abstraction levels of the network, low, medium and high level. Each abstraction level is depicted with a different color in Figure 6(a). Then, we determine the right size and depth of each level by replicating the proper number of building blocks. Indeed, Residual networks are built following this simple concept.

We created different models based on the basic building blocks by finding the best combination of blocks at each abstraction level, low, medium, and last levels. Higher performance and more stable learning resulted by enhancing the intermediate level. This architectural tricks along with the reduction of low and last levels provide notably lighter model in terms of trainable weights and computation load with a marginal loss of accuracy. The resulting preliminary network is depicted Figure 6(a).

Examples of more powerful building blocks are Inception blocks [20]. They are used in many state-of-the-art networks [21] and are proven to extract more diverse

Layer name	Output size	preliminary	Layer name	Output size	MonuNet
conv2d	$112 \times 112 \times 64$	$7 \times 7, 64, /2$	conv2d	$112 \times 112 \times 64$	$7 \times 7, 64, /2$
max pool	$56 \times 56 \times 64$	$3 \times 3, /2$	max pool	$56 \times 56 \times 64$	$3 \times 3, /2$
conv2d	$56 \times 56 \times 64$	$3 \times 3, 64$	conv2d	$56 \times 56 \times 64$	$1 \times 1, 64$
conv2d	$56 \times 56 \times 64$	$3 \times 3, 64$	conv2d	$56 \times 56 \times 192$	$3 \times 3, 192$
conv2d	$28 \times 28 \times 128$	$3 \times 3, 128, /2$	max pool	$28 \times 28 \times 192$	$3 \times 3, /2$
conv2d	$28 \times 28 \times 128$	$3 \times 3, 128$	conv2d	$28 \times 28 \times 64$	$1 \times 1, 64$
conv2d	$28 \times 28 \times 128$	$3 \times 3, 128$	conv2d	$28 \times 28 \times 96$	$1 \times 1, 96$
conv2d	$28 \times 28 \times 128$	$3 \times 3, 128$	conv2d	$28 \times 28 \times 128$	$3 \times 3, 128$
conv2d	$14 \times 14 \times 256$	$3 \times 3, 256, /2$	conv2d	$28 \times 28 \times 16$	$1 \times 1, 16$
conv2d	$14 \times 14 \times 256$	$3 \times 3, 256$	conv2d	$28 \times 28 \times 32$	$3 \times 3, 32$
conv2d	$14 \times 14 \times 256$	$3 \times 3, 256$	max pool	$28 \times 28 \times 192$	$3 \times 3, /1$
conv2d	$14 \times 14 \times 256$	$3 \times 3, 256$	conv2d	$28 \times 28 \times 32$	$1 \times 1, 32$
conv2d	$14 \times 14 \times 256$	$3 \times 3, 256$	conv2d	$28 \times 28 \times 256$	$3 \times 3, 256$
conv2d	$14 \times 14 \times 256$	$3 \times 3, 256$	conv2d	$28 \times 28 \times 256$	$3 \times 3, 256$
conv2d	$7 \times 7 \times 512$	$3 \times 3, 512, /2$	conv2d	$14 \times 14 \times 256$	$3 \times 3, 256, /2$
conv2d	$7 \times 7 \times 512$	$3 \times 3, 512$	conv2d	$14 \times 14 \times 256$	$3 \times 3, 256$
adapt. avg pool	$1 \times 1 \times 512$		conv2d	$7 \times 7 \times 512$	$3 \times 3, 512, /2$
fc	4	softmax	conv2d	$7 \times 7 \times 512$	$3 \times 3, 512$
			adapt. avg pool	$1 \times 1 \times 512$	
			fc	4	softmax

Table 3: MonuNet architecture schemes in detail, where Output size is specified in terms of image size and channel depth, MonuNet version column specifies kernel size, number of channels, and stride (default /1).

features through the various branches of the block. Accordingly, an initial feature extraction similar to Inception architecture network can be beneficial.

We slightly modified Inception block, i.e., the basic block used in GoogleNet, by reducing the kernel size from the 5×5 convolutional layer to 3×3. Unlike Inception-ResNet architecture [21] the residual connections and inception block are detached.

The resulting MonuNet architecture is shown in Figure 6(b). In this network, we replaced the first half of the preliminary architecture by inception-style as early feature extraction stage. MonuNet increases its width and reduces its length resulting in a more balanced architecture which requires slightly more computation, but with almost half of the trainable weights of ResNet-18.

The architecture of the preliminary and MonuNet models are detailed in Table 3.

5 MonuMAI: deep learning pipeline and app

The automatic architectural analysis of a given monument image needs for an accurate process of visual information extraction. The definition of MonuMAI deep learning pipeline allows to perform different information extraction strategies such as the style classification and key architectural elements detection in an unified process. Its automation makes it suitable to be performed in MonuMAI app through smartphones.

MonuMAI deep learning pipeline and its different stages are described in Section 5.1. MonuMAI-KED model for key architectural element detection is described in Section 5.2. The aims of MonuMAI app are detailed in Section 5.3.

5.1 MonuMAI deep learning pipeline: Style classification and key architectural elements detection

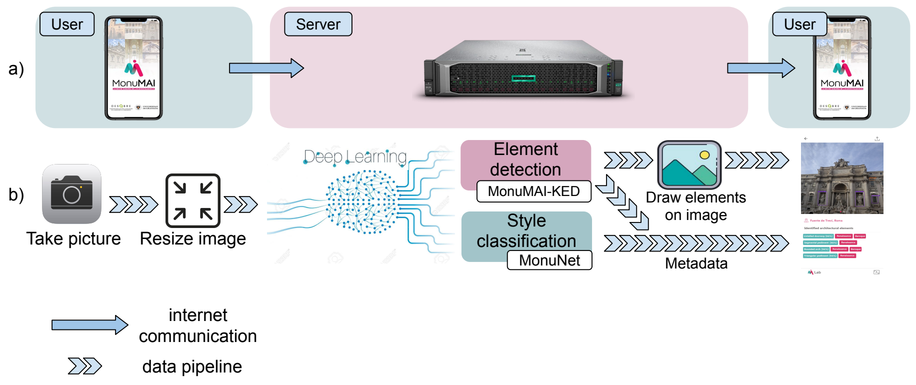


Figure 7: MonuMAI deep learning pipeline illustrate (a) the communication connections and (b) image processing stages, from take a monument picture to respond to the user.

The quality of smartphone cameras and internet connection allows broadcasting the image between user and a central server for heavy processing. For an accurate detection in real conditions, the architectural element detection requires an object detection model with a heavy CNN that need to be performed on a high performance computing server.

MonuMAI app currently runs both elements detection and style classification thought MonuMAI deep learning pipeline that it is shown in Figure 7. MonuMAI deep learning pipeline includes several stages that are listed below:

- The camera application returns a high resolution image.
- The size of the image is lowered to fit the networks and then compressed maintaining its quality.
- The compressed image is sent from user to server.
- MonuMAI deep learning pipeline for classification and detection.

1. Architectural style classification: MonuNet model.
2. Architectural Key elements detection (MonuMAI-KED model).
   - Key element localization: Faster R-CNN (bounding boxes).
   - Key elements classification: ResNet-101.

- The key elements are indicated in the compressed image.
- The image and architectural information metadata are sent back from server to user.
- MonuMAI app shows monument information in a visual interface.

The MonuNet lightweight model can also be performed in the smartphones to further minimizing data communication time and hence improving response time.

5.2 MonuMAI-KED model

The architectural information extraction performed through MonuMAI deep learning pipeline considers different levels of information. The key architectural element detection, called MonuMAI-KED model provides fine-grained information giving the location and the name of the key elements present in the picture of a given monument. Each detected key element contributes to the architectural information of the monument with its name, location, and relationship between the elements and styles.

MonuMAI-KED model is based on Faster R-CNN object detection model and ResNet-101 CNN-based model. It is trained using the key architectural elements annotations provided by MonuMAI dataset.

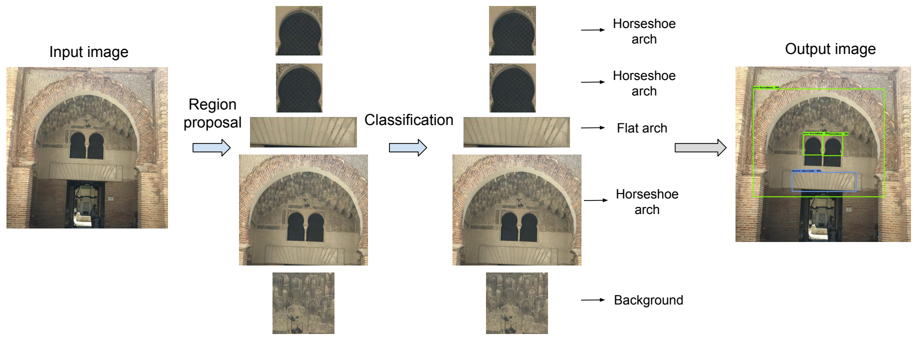


Figure 8: Illustration of the key architectural element detection process of MonuMAI-KED model.

MonuMAI-KED model performs the key architectural element detection through the object detection task based on a DL model that reformulate the problem of key architectural element detection (Figure 8) into a two-step approach of a selective search technique and a CNN classification model:

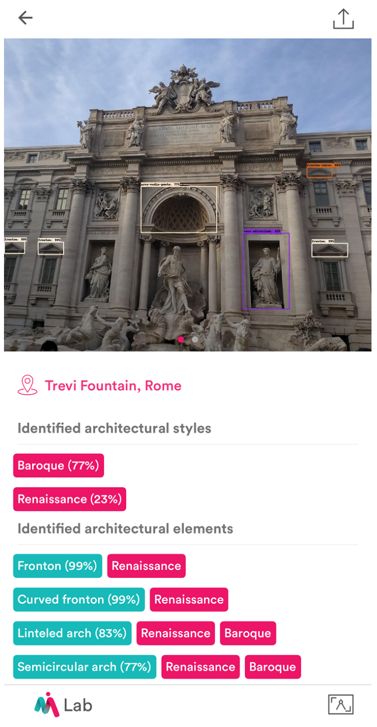
- The selective search method is implemented in Faster R-CNN by a region proposal network that generates candidate regions from the input image. The candidate regions are selected based on a regression learning stage considering the visual features and region shape of the selected elements.

- Then, each region is classified by ResNet-101 CNN as an architectural element or background. The classified regions as key elements are shown as the architectural information from the monument image.

5.3 MonuMAI app

MonuMAI is an interdisciplinary project that aims at emulating the knowledge of the art expert by combining monument images, Artificial Intelligence, and smartphones by means of a citizen science approach.

The obtained automatic system identifies the architectural style and architectural elements of monuments through images and artificial intelligence algorithms. MonuMAI web as well as Android and iOS mobile app are available [5] for free download as tools to spread knowledge among the general public. This tool is also used to collect more monument images in order to increase the available MonuMAI dataset resource. These images are also revised and labeled by art experts.



Trevi Fountain, Rome

Identified architectural styles

Baroque (77%)

Renaissance (23%)

Identified architectural elements

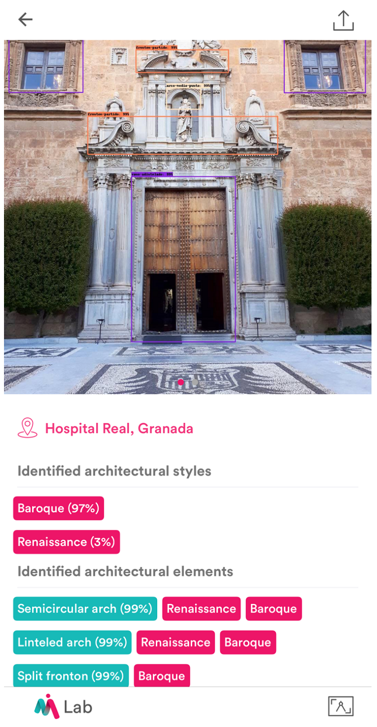
Fronton (99%) Renaissance

Curved fronton (99%) Renaissance

Linteled arch (83%) Renaissance Baroque

Semicircular arch (77%) Renaissance Baroque

(a)



Hospital Real, Granada

Identified architectural styles

Baroque (97%)

Renaissance (3%)

Identified architectural elements

Semicircular arch (99%) Renaissance Baroque

Linteled arch (99%) Renaissance Baroque

Split fronton (99%) Baroque

(b)

Figure 9: MonuMAI app examples of analysed monuments shared by users. Best shared monuments are visible to all users. Figure (a) shows the Fontana di Trevi, Roma, and (b) the Hospital Real, Granada.

The use of mobile applications is one of the most influential achievements for the

[5]monumai.ugr.es

society in the last decade. The educational applications represent a rising market demand from users [6]. In combination with machine learning techniques, the integration of artificial intelligence algorithm in our daily apps have become usual [23]. Besides, these algorithms provides functionalities from intelligent behavior that adapts to users or performs inherently human tasks. For instance, DL techniques based on CNNs can perform visual tasks. It is noteworthy several functions of the image processing field such as the scene recognition [29] in Google Photos for image topic grouping or image quality improvements for camera applications. Facial recognition [4] also is an important feature of apps such as Facebook and Instagram.

We develop MonuMAI app using DL models to extract the architectural information of monument images, illustrated in Figure 9. The architectural element detection and style classification in combination with the monumental heritage taxonomy provides meaningful information about some elements on the morphology of the monument.

6 Experimental Analysis

This section provides a first analysis of MonuNet architectural style classification and key element detection models using MonuMAI dataset. The experimental setup is given in Section 6.1. The results of the MonuNet model for architectural style classification using the first type of annotations in MonuMAI dataset is given in Section 6.2, and the results of the object detection model for key architectural element detection using the second type of annotations of MonuMAI dataset is given in Section 6.3.

6.1 Experimental setup

All the results shown in this section are obtained using 5 fold cross validation technique. This partition scheme is also used to train and validate the key-elements detection model and the architectural style classification model.

The performance metrics used to evaluate the architectural style classification of the four architectural styles are precision, recall, and F1 score (equation 1).

$$\begin{aligned} precision &= \frac{TP}{TP+FP} \\ recall &= \frac{TP}{TP+FN} \\ F1 &= 2 \times \frac{precision \times recall}{precision + recall} \end{aligned} \tag{1}$$

where the number of true positives (TP), false positives (FP), and false negatives (FN) is computed for each class.

The corresponding macro-precision, macro-recall, and macro-F1 are the average precision, recall, and F1 respectively of the four style categories for the multi-class classification.

[6]US-education-technology-education-apps-market-us-2016-2020

The detection performance of Faster R-CNN on the 15 element classes is evaluated in terms of mAP (equation 2) and mAR (equation 3) standard metrics for object detection tasks given 100 output regions.

$$mAP = \frac{\sum_{i=1}^{K} AP_i}{K} \qquad AP_i = \frac{1}{10} \sum_{r\in[0.5,\ldots,0.95]} \int_0^1 p(r)dr \tag{2}$$

$$mAR = \frac{\sum_{i=1}^{K} AR_i}{K} \qquad AR_i = 2\int_{0.5}^1 recall(o)do \tag{3}$$

where given K categories of elements, p represents the precision and r $recall$ define the area under the interpolated precision-recall curve for each class i. whereas o is IoU (intersection over union) in recall(o) is the corresponding recall under the recall-IoU curve for each class i.

The classification models were built and evaluated using PyTorch api [16]. For the classification task, we trained models on MonuMAI dataset for the architectural style classification. As optimization algorithm, we used Stochastic Gradient Descent(SGD) and cross-entropy loss training from scratch. The data augmentation performs equal normalization at train and test, resize to 224 square images from random crop at train and center crop at test.

The detection models were built and evaluated using TensorFlow [1] as back-end and its object detection api [6] as front-end. The architectural element detection we used Faster R-CNN architecture based on ResNet-101 pre-trained on COCO [10]. We fine-tuned the weights and retrain model on our dataset. As optimization method, we used SGD: momentum of 0.9, learning rate of 0.0003, and batch size of 1.

All the experiments were carried out on a GPU cluster with Nvidia Titan RTX. MonuMAI dataset, pre-trained MonuNet model and all the hyper-parameters for classification and detection are available at [7].

6.2 MonuNet model analysis

This section provides an exhaustive analysis of diverse well known CNNs in the task of architectural style classification, in Section 6.2.1. All these models are trained on the new built MonuMAI classification dataset. The evaluation of the new MonuNet model is given in Section 6.2.2.

6.2.1 Results of lightweight architectures

We evaluated eight diverse nets in the architectural style classification task using MonuMAI dataset. The weight, number of layers, trainable layers, FLOPs, macro-precision, macro-recall, and macro-F1 are summarized in Table 4.

As we can observe from Table 4, in general, higher performance are achieved by residual connection based models. In particular, ResNet-18 provides the highest performance with 89.3% macro-precision, 88.54% macro-recall, and 88.87% macro-F1.

[7]github.com/ari-dasci/OD-MonuMAI

	#parm($\times 10^6$)	**#layer**	**#lrn layer**	**FLOPs**($\times 10^9$)	**m-prec**(%)	**m-recall**(%)	**m-F1**(%)
ResNet-18	11.69	69	41	1.82	**89.30**	**88.54**	**88.87**
ResNet-34	21.80	125	73	3.67	88.66	88.13	88.37
DenseNet-121	7.98	371	242	2.87	87.53	87.53	87.51
GoogleNet	7.01	197	115	1.59	87.21	87.01	87.07
ShuffleNetV2 x2.0	7.39	168	113	0.59	85.63	84.39	84.85
MobileNetV2	3.50	159	105	0.31	83.95	83.99	83.91
ShuffleNetV2 x1.0	2.28	168	113	**0.15**	81.60	80.34	80.79
SqueezeNet 1.0	**1.24**	66	26	0.35	71.06	66.85	68.37

Table 4: Characteristics and performance of the considered classification models on MonuMAI architectural style classification dataset. Performance metrics are present in terms of macro-precision (m-prec), macro-recall (m-recall), and macro-F1 score (m-F1). Characteristics are presents in terms of number of parameters (#parm), number of layers (#layers), number of learnable layers (#lrn layer), and floating point operations per second (FLOPs).

Deeper and heavier models such as ResNet-34 or DenseNet-121 provide respectively 0.5% and 1.36% lower macro-F1. Notice that GoogleNet model based on stacked Inception modules is considerably lighter but provides lower performance than ReseNet-18, 1.8% lower macro-F1.

6.2.2 MonuNet model performance and optimization

MonuNet and preliminary models pursue an optimal balance between size, computation, and performance. In this section we analyse and evaluate both networks.

	#parm($\times 10^6$)	**#layer**	**#lrn layer**	**FLOPs**($\times 10^9$)	**m-prec**(%)	**m-recall**(%)	**m-F1**(%)
Preliminary model	7.56	62	37	1.59	88.95	88.15	88.50
MonuNet	6.38	59	35	1.95	90.09	89.59	89.81

Table 5: Characteristics and performance of MonuNet model and preliminary version on MonuMAI architectural style classification dataset.

MonuNet model seeks a good balance between accuracy and computation requirements as shown in Table 5. Both preliminary and MonuNet models have high performance and lower computational requirements. The preliminary model has a comparable number of layers to ResNet-18, however, it has 12.6% and 35.3% less FLOPs and number of parameters respectively with only 0.37% lower macro-F1.

The best performance on MonuMAI architectural style classification is obtained by MonuNet model. The integration of the modified Inception module in the low level layers boosted the model performance up to 90.09% macro-precision, 89.59% macro-recall, and 89.81% macro-F1.

MonuNet model outperforms ResNet-18 by reducing the number of parameters by 45.4%, the number of general layers by 14.5%, the number of trainable layers by 14.6%, but increasing the number of FLOPs by 7.1%. Thus, MonuNet model keeps a low computational cost (FLOPs) but minimizing the model size, which is suitable to be performed along with numerous processes and apps on smartphones.

The performance of MonuNet model per each architectural style is shown in Table

6.

	#img	**Baroque**	**Gothic**	**Hisp-Muslim**	**Renaissance**	**Precision**(%)	**Recall**(%)	**F1**(%)
Baroque	516	469	10	3	34	89.50	90.89	90.19
Gothic	359	4	345	6	4	91.03	**96.10**	93.50
Hisp.-Muslim	327	3	12	307	5	**95.05**	93.88	**94.46**
Renaissance	312	48	12	7	245	85.07	78.53	81.67
					macro	90.09	89.58	89.81

Table 6: Results of MonuNet classification model on MonuMAI architectural style classification dataset.

In general, MonuNet model provides good results in all four styles. In particular, the Hispanic-Muslim style obtains the highest performance with 95.05% precision, 93.88% recall, and 94.46% F1. Unlike the Renaissance style which obtains the lowest performance, with 85.07% precision, 78.53% recall, and 81.67% F1. These differences can be explained by the fact that:

- The Hispanic-Muslim monument appearance is quite distinct in architectural forms or decorative painting.

- The overlapping between Renaissance and Baroque features such as shared architectural elements, or similar building characteristics make the classification difficult, as it can be seen in the confusion matrix of Table 6.

The overall high performance can be explained by the quality of the dataset that includes images with high resolution, the monuments facade is centered filling most of the image area, and very few noise. More monument classification results are shown in Figure 10. As it can be seen, the style classification is accurate in many cases, but some uncertain classifications generated in adverse conditions by contextual elements, shadows, or light conditions.

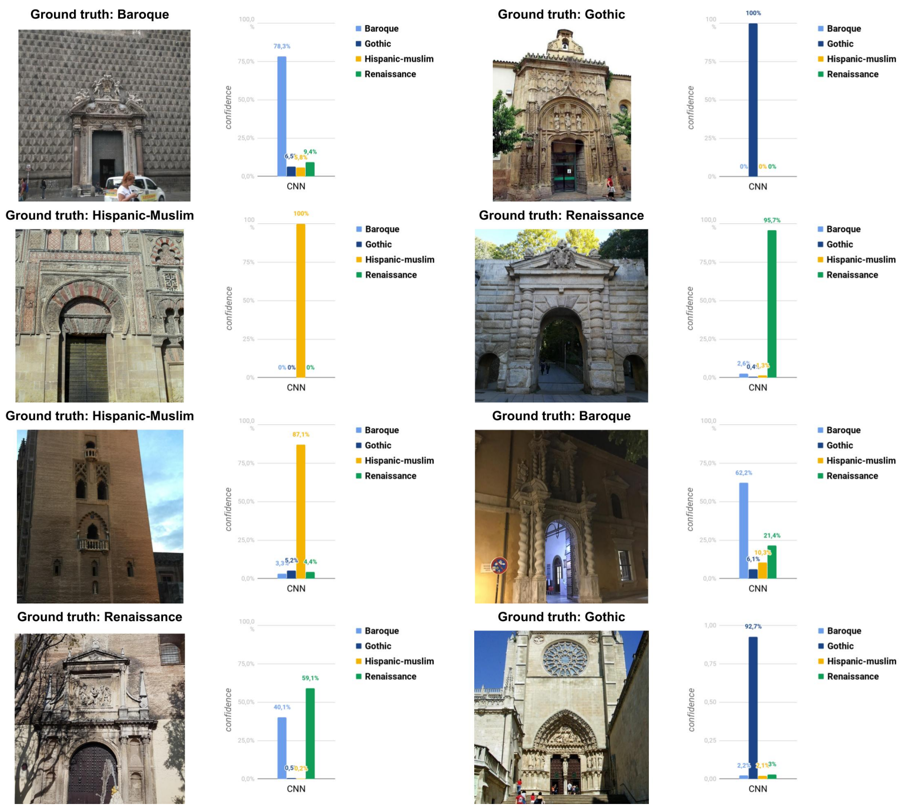


Figure 10: Examples of architectural style classification in monument facades. MonuNet model is trained on MonuMAI classification dataset.

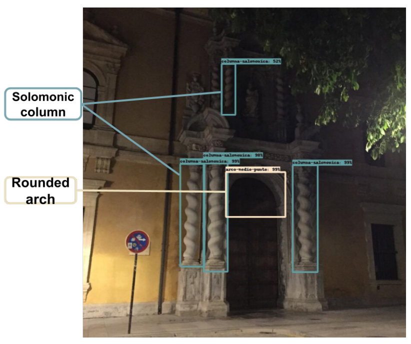
6.3 MonuMAI-KED model analysis

This section analyses the performance and results of the architectural elements detection on the MonuMAI detection dataset, depicted in Section 3.2.

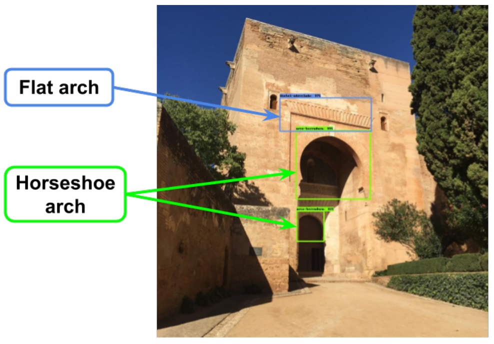
mAP	mAP[0.5]	mAP[0.75]	mAP S	mAP M	mAP L	mAR	mAR S	mAR M	mAR L
0.602	0.911	0.712	0.569	0.527	0.64	0.683	0.58	0.602	0.716

Table 7: Detection performance metrics of the architectural element detection model trained on MonuMAI key architectural elements dataset.

The evaluation of the object detection model on the fifteen classes are shown in Table 7. As it can be seen, the detection model achieves an mAP of 0.6 and mAR of 0.68. The performance mAP can be considered as good since the state-of-the-art detectors on COCO detection challenge with 80 classes is 0.52 taking into consideration the difference in the amount of data. Higher mAP is better.



(a)



(b)

Figure 11: Examples of architectural elements detection in complex scenes. Figure (a), the College of San Pablo (facade built in 1717, Granada), and figure (b), Gate of justice in Alhambra (1348, Granada).

Two examples of monument images in complex scenes are shown in Figure 11.

- The monument (a) shows the Faculty of Law in the College of San Pablo, which is a Baroque monument. As it can be seen, this image includes multiple types of noise, part of a tree, a road sign and low light with shadows. However the detection model is able to detect architectural elements as the characteristics Solomonic columns in low light conditions.
- The image (b) shows the Royal Monastery of Cartuja, a renaissance monument. The image shows smaller elements that are correctly detected, for instance, the Segmental pediment which is distinctive of Renaissance.

Additional detection results giving architectural element information are presented in Figure 12. The figure shows diverse monuments and contexts of the four architectural styles with different perspective and natural light conditions. Hence, the potential of architectural element detection based on CNNs provide outstanding results given contextual background, distance, or monument features. Despite of common elements such as pedestrians, plants in general and cars can occlude some parts of the image, or sometimes urban layout forces the perspective from which the pictures are taken.

The potential of CNN based models have proved their ability to differentiate similar architectural elements even when their shapes and building features are similar. The architectural element detection performs in realistic and complex scenarios providing architectural information.

In conclusion, the detection of architectural elements shows considerable performance despite external object in the background or parts of urban infrastructure.

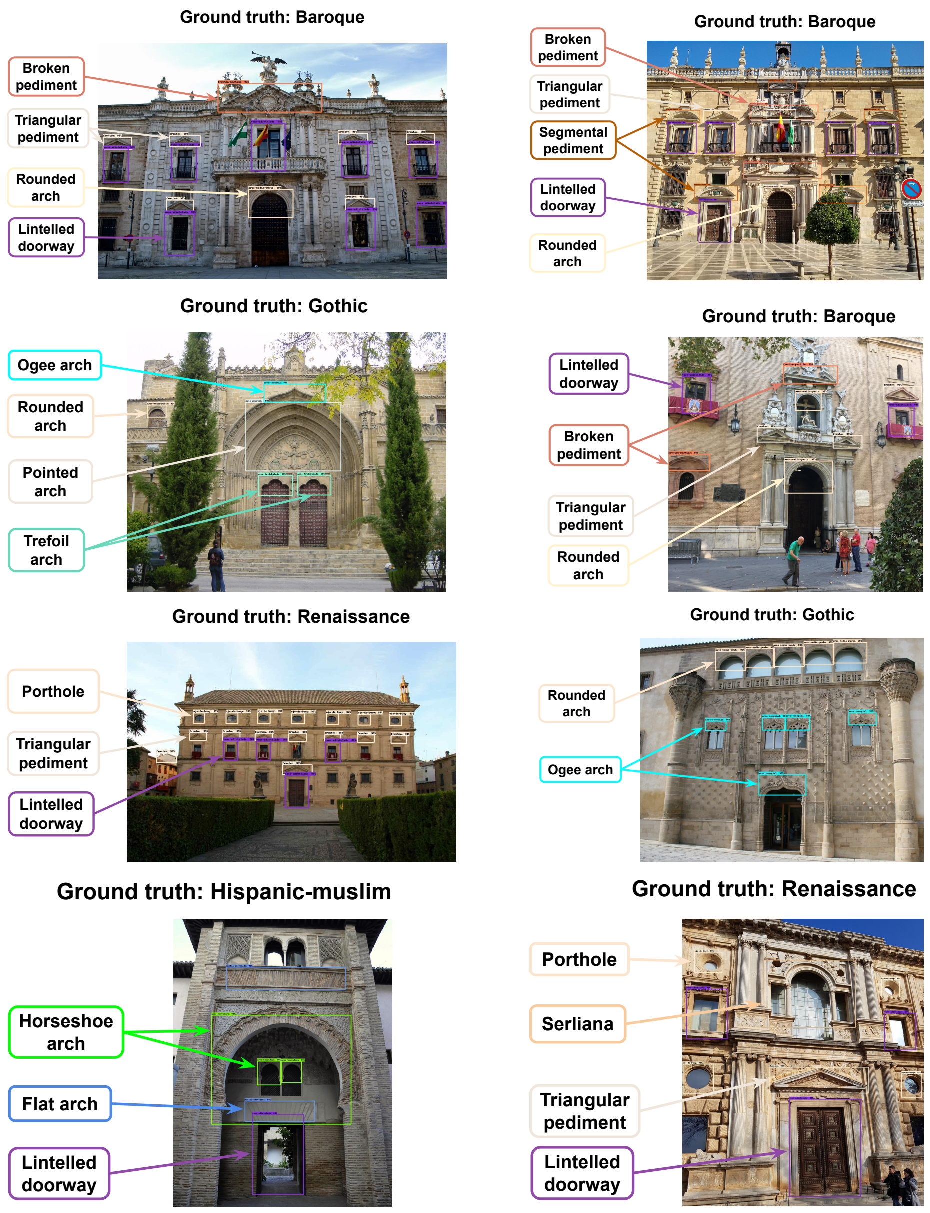


Figure 12: Examples of architectural element detection for monument facades in diverse situations, brightness, perspective, distance to monument, or urban elements.

7 Conclusion remarks

This work presents MonuMAI framework that includes the three main axis proposed in this work. MonuMAI dataset designed according to the defined monumental heritage taxonomy, for architectural styles classification and key elements detection tasks. MonuMAI deep learning pipeline based on MonuNet architecture and MonuMAI-KED model for architectural style classification and key element detection respectively. Citizen science based MonuMAI app that integrate MonuMAI deep learning pipeline using MonuMAI dataset for performing in real life conditions.

The obtained MonuNet and MonuMAI-KED models trained on MonuMAI dataset show high potential even in low quality images and provides satisfactory results in real condition scenarios as shown by MonuMAI app. Such automatic system could ease the democratization of this kind of knowledge to a much wider segment of the population by integrating it into smartphones or in combination with augmented or virtual reality applications that can be used in several fields, e.g., in tourism, teaching, or science dissemination.

In conclusion, we highlight MonuMAI dataset and the monumental heritage taxonomy as primary resources for building automatic models of architectural information extraction. We also point out the potential of MonuMAI deep learning pipeline through MonuNet and MoniMAI-KED models to perform in real life conditions as it is shown by MonuMAI app.

As future work, we will work on developing an embedded architecture for the architectural style classification and key elements detection. Both tasks can provide feedback to each other, implement explainable mechanisms, and move embedded model execution to smartphones.

Acknowledgements

This work was partially supported by the Spanish Ministry of Science and Technology under the project TIN2017-89517-P. S. Tabik was supported by the Ramon y Cajal Programme (RYC-2015-18136).

References

[1] Martín Abadi et al. Tensorflow: A system for large-scale machine learning. *Operating Systems Design and Implementation*, 16:265–283, 2016.

[2] Alberto Castillo et al. Brightness guided preprocessing for automatic cold steel weapon detection in surveillance videos with deep learning. *Neurocomputing*, 330:151–161, 2019.

[3] Wei-Ta Chu and Ming-Hung Tsai. Visual pattern discovery for architecture image classification and product image search. In *Proceedings of the 2nd ACM International Conference on Multimedia Retrieval*, page 27. ACM, 2012.

[4] Xuanyi Dong et al. Supervision-by-registration: An unsupervised approach to improve the precision of facial landmark detectors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 360–368, 2018.

[5] Kaiming He et al. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[6] Jonathan Huang et al. Speed/accuracy trade-offs for modern convolutional object detectors. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7310–7311, 2017.

[7] Phillip Isola et al. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.

[8] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015.

[9] Yi Li et al. Fully convolutional instance-aware semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2359–2367, 2017.

[10] Tsung-Yi Lin et al. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[11] Tsung-Yi Lin et al. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017.

[12] Tsung-Yi Lin et al. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[13] Hantang Liu et al. Deepfacade: a deep learning approach to facade parsing. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 2301–2307. AAAI Press, 2017.

[14] Jose Llamas et al. Classification of architectural heritage images using deep learning techniques. *Applied Sciences*, 7(10):992, 2017.

[15] Dhruv Mahajan et al. Exploring the limits of weakly supervised pretraining. In *European Conference on Computer Vision*, pages 185–201. Springer, 2018.

[16] Adam Paszke et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pages 8024–8035, 2019.

[17] Shaoqing Ren et al. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[18] Gayane Shalunts. Architectural style classification of building facade towers. In *International Symposium on Visual Computing*, pages 285–294. Springer, 2015.

[19] Gayane Shalunts, Yll Haxhimusa, and Robert Sablatnig. Classification of gothic and baroque architectural elements. In *2012 19th International Conference on Systems, Signals and Image Processing (IWSSIP)*, pages 316–319, April 2012.

[20] Christian Szegedy et al. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[21] Christian Szegedy et al. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-first AAAI conference on artificial intelligence*, pages 4278–4284, 2017.

[22] Fei Wang et al. Residual attention network for image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3156–3164, 2017.

[23] Mengwei Xu et al. A first look at deep learning apps on smartphones. *arXiv preprint arXiv:1812.05448*, 2018.

[24] Zhe Xu et al. Architectural style classification using multinomial latent logistic regression. In *European Conference on Computer Vision*, pages 600–615. Springer, 2014.

[25] Nianyin Zeng et al. Facial expression recognition via learning deep sparse autoencoders. *Neurocomputing*, 273:643–649, 2018.

[26] Nianyin Zeng et al. Deep-reinforcement-learning-based images segmentation for quantitative analysis of gold immunochromatographic strip. *Neurocomputing*, 2020.

[27] Luming Zhang et al. Recognizing architecture styles by hierarchical sparse coding of blocklets. *Information Sciences*, 254:141–154, 2014.

[28] Peipei Zhao et al. Architectural style classification based on feature extraction module. *IEEE Access*, 6:52598–52606, 2018.

[29] Bolei Zhou et al. Learning deep features for scene recognition using places database. In *Advances in neural information processing systems*, pages 487–495, 2014.