Architecture recognition by means of convolutional neural networks

ARCHITECTURE RECOGNITION BY MEANS OF CONVOLUTIONAL NEURAL NETWORKS

Louis N. Andrianaivo[1,2], Roberto D'Autilia[2], Valerio Palma[1]

[1] F*UL*L | the Future *Urban Legacy* Lab,
Politecnico di Torino,
Via Agostino da Montefeltro 2, 10134 Torino, Italy
valerio.palma@polito.it
[2] Dipartimento di Matematica e Fisica,
Università degli Studi Roma Tre,
Largo San Leonardo Murialdo 1, 00146 Roma, Italy
landrianaivo@mat.uniroma3.it, roberto.dautilia@uniroma3.it

Commission II,WG II/8

KEY WORDS: Artificial Intelligence, Machine Learning, Deep Learning, Convolutional Neural Networks, Image Classification, Architectural Heritage, Mobile Computing

ABSTRACT:

The use of mobile computing technologies can change the experience of visiting cultural sites by making vast digital heritage collections accessible on site. The spread of machine learning technologies on mobile devices is encouraging the interaction of artificial intelligence with the shape of the built environment. However, while some research already applies deep learning image recognition in an urban context, the literature on how to develop effective neural networks to detect architectural features is still limited, as well as the availability of architecture-related datasets. This work presents the steps and results of the prototype development of a mobile app to perform monument recognition using convolutional neural networks. The tool allows users to interact with the physical space and access a digital archive of texts, models, images and other data.

1. INTRODUCTION

1.1 Introduction

It is almost commonplace to say that the machine learning technology is changing the way we interact with the real world. For many years the theoretical studies of the brain structure (Amit, 1992), the artificial neural networks (Hopfield, 1982) and the physics of disordered systems (Mezard et al., 1987) have built the theoretical foundation for effective machine learning systems, but only in recent years the technology has been available to engineer on these intelligent devices. The machine learning technology puts together models of neural networks based on Hebb ideas (Hebb, 1949), the modeling revisitation of these concepts (Hopfield, 1982, D'Autilia and Guerra, 1991) and the new hardware and software tools (Abadi et al., 2016).

In recent years, the availability of parallel computers with fast GPUs has made it possible to simulate networks made of many neurons with complex connection topologies. The resulting machine learning systems are often considered only black boxes that can be used by those who know little or nothing of the theory. However many problems are still open. For example it is not clear how to find the optimal topology of the network, for a given problem, although the enormous number of the published attempts realizes a zoology of models that can be compared and selected as in a sort of evolutionary environment.

From an epistemological point of view one of the most relevant aspects of the diffusion of the machine learning is given by its interdisciplinary. The Artificial intelligence spans across different subjects, including disciplines such as linguistics, theoretical physics, computer science, art, biology or logic. After all, machine learning has been designed to parody the functions of the brain, by definition an interdisciplinary device.

In this cultural framework we suggest to use deep learning methods to build queries to access a database starting from the real world data. Our goal is to consider a real object such as a monument or an architectural artifact image as the input of a machine learning system to query a database and to extract all the information related to that object.

1.2 Motivation

The objects that make up a city can be viewed as the links to a set of stories. Most of these information is often inaccessible. An ancient, modern or even an archaeological site is a place where many events happened in different times. The corpus of this information forms a virtual network of mutually connected information. This information can be stored in a database, but the queries to access them can depend on many parameters. In presence of an ancient statue one may be interested in the sculptural technique, in the history of the person represented, in its archeology or in the biography of the author. At the same time we may be interested in learning about all the similar specimens found in other areas of the world.

Solutions such as audio tours, informative panels or QR codes are not suitable to perform all these queries. It is worth to note that widespread portable music recognition systems already interact with features of the environment to get access to data and services. Similarly, a mobile device could connect locations, artworks, architectural objects and their spatial characteristics to carefully selected digital contents. Starting from the recognition of object itself, a machine learning system can produce complex queries, and by learning the interests of the users it can reorganize the data in the most appropriate way.

Another key challenge in cultural heritage management is to optimize the resources allocated. Most of the informative infrastructure that can be found in cultural sites requires maintenance and site-specific design. Machine learning technologies can be suitable for both large sites that cannot provide appropriate services and small sites that cannot afford surveillance and maintenance.

Mobile computing technologies can overcome many limitations of previously adopted methods to enhance the experience of visiting cultural sites and make digital heritage collections increasingly accessible.

1.3 Related works

Unprecedented computational resources and the availability of ever-growing datasets have recently boosted the development of deep learning (DL) techniques. DL algorithms are a subset of machine learning (ML) models which allows a machine to represent complex concepts on the basis of a hierarchy of simpler concepts (Goodfellow et al., 2016). Exploiting the capacity to learn from experience, these models can effectively interpret an input object to assign it to a category. Convolutional neural networks (CNNs) are a class of DL models which is largely applied to deal with images (Rawat and Wang, 2017). CNNs are currently used for computer vision tasks such as face recognition, handwriting recognition or image analysis for medicine and biology (Hosny et al., 2018, Webb, 2018). The spread of these technologies on mobile devices is encouraging the interaction of the artificial intelligence with the shape of the built environment.

Given a strong interest for self-driving cars, CNNs are extensively used for object detection and segmentation on street level imagery (Cordts et al., 2016). Recently, CNNs have been applied in building façade segmentation (Stathopoulou and Remondino, 2019) and architectural landmark classification (Gada et al., 2017, Amato et al., 2016). Interest in landmark recognition has also been demonstrated by Google, who issued two large datasets since 2018 and related challenges for the data scientist community Kaggle (Araujo and Weyand, 2018, Cao and Weyand, 2019). However, the literature on how to develop effective neural networks to detect architectural features is still limited, as well as the availability of architecture-related datasets.

2. DESCRIPTION

2.1 Dataset

The present application was developed for the Central Archaeological Area in Rome, which includes the imperial Fora. Compared to other datasets for object recognition, the collection of monument images from such a small area poses some challenges. First, unlike other objects, landmarks are fixed in their position, hence pictures of a same monument are likely to be similar to each other and not suitable for learning to distinguish a monument from the surroundings. Second, unlike other landmark datasets, this one gathers objects which are quite close together, both spatially and historically, so that pictures may often show the same background and similar architectural features.

Starting the project from scratch and having to document not only famous buildings, we took pictures specifically for the project. Our dataset labels the images of 46 monuments of the Imperial Fora, with pictures spanning a complete overview of the architectural characteristics. The viewpoints have been chosen with particular care: the monuments were framed from different positions and we included the most common places for the visitors, details and panoramic views. We also chose different object lighting and camera exposure conditions to make the DL model deal with more difficult circumstances. We both used pictures taken from mobile phones and professional cameras.

Monuments not only show recurring architectural structures and decorations. In fact, they feature heterogeneous dimensions and conservation status: some are quite incomplete and poorly preserved, some are part of more recent buildings, some are large and composed of many different parts.

2.2 Artificial neural networks (ANNs)

Artificial neural networks (ANNs) are the bulk of machine learning, a technology that can be used to approximate general functions. Inspired by the biological structure of the nervous system (McCulloch and Pitts, 1943), ANNs can be viewed as a graph where the nodes represent neurons (*perceptrons*, *processing units*) and the edges are the connections between couples of neurons as in Fig.1. The transfer of information

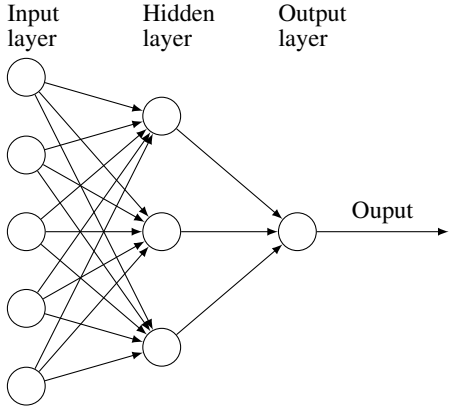

Figure 1. An artificial neural network example.

between neurons is given by the combination of non-linear operations

$$\text{out} = \sum_{i=1}^{N} v_i \times \varphi(w_i \times x + b_i) \tag{1}$$

where the operators $\times$ and $+$ are defined between tensors, and w_i and v_i are respectively the weight of the edges connecting the input layer to a hidden layer and the hidden layer to the final layer (output). The b_i are the bias assigned to each unit in the hidden layer. The function φ is the (in general non-linear) transfer or activation function.

This type of neural network is often referred to as *feedforward* or fully connected. An example of activation function, often used in fully connected neural networks, is the *ReLU*, Rectified Linear Unit, $f(x) = \max(0, x)$. By the Universal Approximation Theorem (Cybenko, 1989) any continuous function can be approximated by means of a feed forward neural network.

The criteria for the choice of the activation function are given by the hypothesis of the theorem. The neural network approximates the function by minimizing the error between the real values and the values predicted by means of the weights and biases of the graph. The main idea is to find the minimum of a value function by applying the *gradient descent* method on the error function. In general, datasets are very large in size and dimension so it is more efficient to use the *stochastic gradient descent* (SGD) to save memory, applying the method on a randomly chosen batch of the dataset.

In ANNs, the SGD is applied to each layer. As we only have the real value of the output layer, we start from the final layer and go backward. This process is called *backpropagation*.
The choice of the neural network parameters (N number of the units, activation function, etc) is usually matter of experience, for the right parameters are those which let the network perform the best.

2.3 Deep learning and convolutional neural networks (CNNs)

The design of the architecture of a neural network does not have a clear setup. However, it can be observed that the more units we have the better the neural network approximates the function we want to model. In (Hornik, 1991) an estimation of this number is given, which is exponential over the dimension of the input in the worst case, and in (Sutskever and Hinton, 2008) it is suggested to break the feedforward network into several hidden layers to improve the performance.

This kind of network is referred to as the *deep feedforward neural network* and its architecture is a composition of feedforward neural networks. The number of hidden layers is the *depth* of the network, and this family of networks forms a class of deep neural networks. *Deep learning* techniques study these families of networks.
Deep learning is basically an optimization and regulation of the tensor operations so that the network could perform well.

2.3.1 CNNs description: The *Convolutional neural networks* (CNNs) (LeCun, 1989) belong to the class of deep neural networks, based on the mathematical convolution operation, a method used in signal processing to minimize the noise. Indeed, let f and g be two functions well defined in $(-\infty, +\infty)$, the convolution of f and g is defined by

$$(f * g)(t) = \int_{-\infty}^{+\infty} f(x)g(t-x)dx \tag{2}$$

Here f is referred to as the output signal where g is the weight, a distribution probability. For CNNs, these are the input dataset and the *kernel* respectively. The equivalence of this formula in a 2D ($2^{\text{-nd}}$ rank tensor) discrete time is given by

$$(I * K)[i,j] = \sum_{r=0}^{k-1}\sum_{c=0}^{k-1} I[r+i, c+j]K[r,c] \tag{3}$$

In Fig.2 an example of this operation is shown as a tensor product. Notices that the dimension of the weighted output is reduced. To keep the same dimension as the input we could use *zero padded* input, this makes sense for instance for 1D discrete signal processing application by assuming that at time 0 there is no signal.

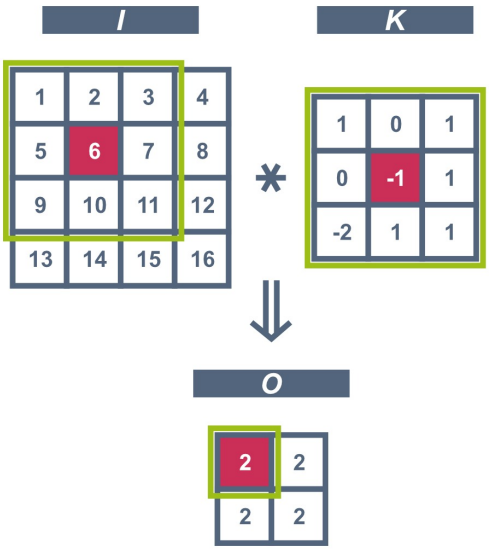

Figure 2. A 2D convolution example.

The convolution is well defined for any dimension of the inputs, for example 1D for time series and 3D for color video. A traditional application of the convolution operation is image filtering. Image data can be considered as a 2D layer for RGB channels which is compatible with convolution operations.

The CNN architecture for image classification is illustrated in Fig.3. There are several layers in which operations on the coordinates are performed by using convolution to extract

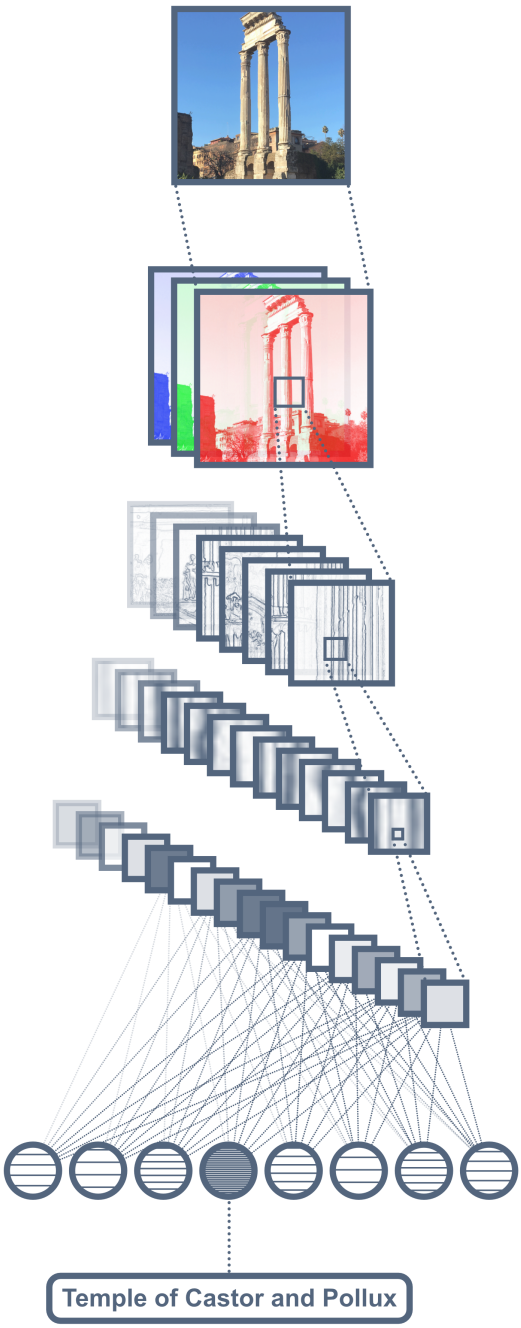

Figure 3. Illustration of a CNN on image data.

important features in the pictures. At the final layer we have the fully connected one which predicts the output.

A convolutional layer is composed by three operations: *convolution*, *non-linear function and pooling* (sub sampling). As shown in Fig.3 the first hidden layer (edge detection) for instance outputs different layers of features which correspond to several kernels. Next to the convolution we introduce the non-linear function. Its purpose is similar to the feed forward neural network. In most cases, we use ReLU to replace the negative coordinates (the pixels of the picture) by zero. The last operation is pooling, to reduce the dimension of each output by keeping the important information. There are many pooling techniques available: *max-pooling*, *average-pooling*, *sum-pooling*. An example of max-pooling is shown in Fig.4, parameterized by the *size* and the *stride* with a filter of size

2×2 with stride 2.

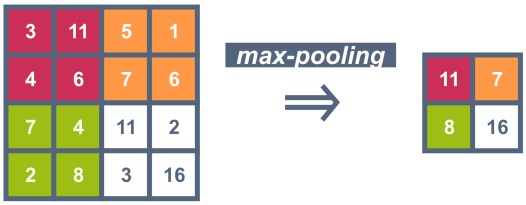

Figure 4. A Max-pooling example.

2.3.2 CNNs training: The training of the CNNs is the same as in the case of the feed forward neural networks. In Fig.3 the last layer is a fully connected layer and the convolution layer is the result of a tensor product. The backpropagation procedure has been used to train the model. As mentioned before, the gradient descent method can be exploited to find the minimum of the error.

This is done in three main steps: inference (forward), computation of the gradient and updating the weights (backward). More precisely, we start from a random weight and bias, run an inference on batch of the dataset and finally compute the gradient of the error function (using a chain rule) to update the weights of the network with respect the backward ordered layer.

The last step is the most important for the training and has to be done carefully. Due to memory and computational resources, the statistical gradient descent performs better together with optimization and regularization techniques.

3. METHODS AND EXPERIMENTS

This section presents the methods and strategies adopted to construct and optimize the ML model. We describe the steps of data preparation (including augmentation techniques and data cleaning), network training and testing and model building for different platforms. For each step, we discuss methodology and the operations we carried out in our experiments.

3.1 Methodology

3.1.1 Data preparation. The preparation of the dataset is one of the fundamental steps in ML. Good results require a careful treatment of data. We started by inspecting the available images for each class, then we applied data augmentation, formatting and cleaning.

methods:

1. *Data selection:* we collected some samples of pictures for each monument. Since pictures may present different classes (more than one monument), we manually verified each picture and highlighted (*e.g.* by cropping the picture) the features to make the network “learn” properly the corresponding monument.

2. *Data augmentation:* The collected picture dataset does not contain enough images to train the network. Some platforms can artificially generate more data during the training, but we chose to extend the dataset in advance. In fact, the relatively small number of images available for each class makes it difficult to handle duplicate data if we cannot access the random image generation algorithms.
For our purposes, we needed at least 600 pictures for each class (500 for training, 100 for validation). To improve the dataset, we applied the following transformations:
   - Rotation (clockwise and counter-clockwise for a small range of angles)
   - Crop (given an estimation of the area occupied by the class object)
   - Flip (top-bottom and left-right)
   - Distortion (simple, Gaussian, ...)
   - Zoom
   - Histogram equalization
   - Invert
   - Resize (related to what needed by the network architecture)

3. *Data cleaning:* each operation applied during the data augmentation is random. Since several duplicate images are expected, we adopted a classical technique to cleaned up the data, by creating a hash table for the images. For this, we used a *JSON*[1] format database.

operations:
Data augmentation resulted in more than 70% duplicate images for each classes, so we needed to generate more than the targeted 600 images. In our experiment, we produced around 2000 images per class using Augmentor (Bloice et al., 2019) (image augmentation library in Python). The data cleaning was done using a MongoDB[2] database to hash the images.

3.1.2 Network training. The convolutional network works well for the classification task and is widely used for computer vision problems. The idea is to extract the features of the input image by using a sufficient number of convolutional layers and use a fully connected network at the final layer to perform the classification (*deep convolutional neural network*). The main purpose of training is to tune the kernel parameters, together with the weights and the biases for the fully connected layer. Moreover, we need to be able to use the inference model into a mobile device.

methods:

1. *The model:* after several trials, we chose to use a class of CNNs known as the *MobileNets* (Howard et al., 2017). It was developed by Google researchers to be an efficient candidate for mobile DL models. The model fits our needs because it is fast and light and it shows better accuracy when tested on the *ImageNet* database. The main feature of this architecture is the use of *depthwise separable convolutions*, composed by depthwise 3×3 and pointwise 1×1 convolution layers. This significantly reduces the number of parameters to be trained in the network. The MobileNet-224 architecture can be seen as follows:
In Table 1, the architecture of the mobileNet with the input image shape $224 \times 224 \times 3$ is shown. Each convolution layer is followed by the batch normalization (Ioffe and Szegedy, 2015) and the activation function ReLU. The width multipliers of the MobileNet model are $\alpha = 0.25, 0.5, 0.75, 1.0$, to reduce the size of the model and the width of the depthwise separable convolution.

2. *Training:* the MobileNet model is easy to train for a right choice of parameters. To this purpose we used some known techniques for debugging neural networks such as:
   - Initialization of the weight
   - Regularization of the tuning parameter
   - Choice of the back-propagation function
   - Fine tuning
   - Dropout

3. *Test :* we tested the accuracy of our model for a set of 100 images not included in the training dataset.

[1] JavaScript Object Notation.
[2] Cross-platform document-oriented NoSQL database program.

convolution type	kernel shape	input shape
normal	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
depthwise	$3 \times 3 \times 32$	$112 \times 112 \times 32$
pointwise	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
depthwise	$3 \times 3 \times 64$	$112 \times 112 \times 64$
pointwise	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
depthwise	$3 \times 3 \times 128$	$56 \times 56 \times 128$
pointwise	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
depthwise	$3 \times 3 \times 128$	$56 \times 56 \times 128$
pointwise	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
depthwise	$3 \times 3 \times 256$	$28 \times 28 \times 256$
pointwise	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
depthwise	$3 \times 3 \times 256$	$28 \times 28 \times 256$
pointwise	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
$5 \times$ depthwise pointwise	$3 \times 3 \times 512$ $1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$ $14 \times 14 \times 512$
depthwise	$3 \times 3 \times 512$	$14 \times 14 \times 512$
pointwise	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 1024$
depthwise	$3 \times 3 \times 1024$	$7 \times 7 \times 1024$
pointwise	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
averagePool2D	7×7	7×1024
fully connected	$1024 \times N$	$1 \times 1 \times 1024$

Table 1. MobileNet-224

operations:
We used the implementation of MobiliNet version 1 from *Keras* (Chollet et al., 2015), a Python library for ML, and the technique known as *transfer learning*, which is time saving but results in a good performance of the model. Instead of using random initialization, the network was pre-trained on the ImageNet database because it has the same features (color pixels).

For the training, we needed to match the model and our dataset. We used the 224 version of the MobileNet model with the width multiplier $\alpha = 1.0$. The Keras implementation offers a choice on the pooling layer. For our purpose we used the average pooling 2D as in Table 1 followed by a dropout layer for the training.

In the fully connected layer, we initialized the weight by random uniform value with L_2 regularization. We used SGD for the back-propagation optimizer with the categorical *cross-entropy loss* function (very suitable for *softmax* activation functions). For fine-tuning, we retrained the same network with different numbers of classes, so that we could adjust the training parameters to obtain the right choice.

3.1.3 Building the model. The final step is to use the inference model on a mobile device with iOS or Android operative system.

methods:

1. *Converting the inference model for iOS:* there are several ML libraries that be used on iOS devices. For this project, we took advantage of the recent framework *Core ML*[3] developed by Apple: an API which enables an easy interaction with the device performances, especially for ML application.

2. *Converting the inference model for Android:* here we used *tensorFlow lite*[4], an open source DL framework for mobile included in TensorFlow (Abadi et al., 2016).

operations:
By means of Core ML tools we converted the Keras model into a Core ML model and exploited TensorFlow to obtain the lite version of the inference model to be used on Android systems.

[3]https://developer.apple.com/documentation/coreml
[4]https://www.tensorflow.org/lite/

4. RESULTS AND PERFORMANCE

4.1 Convergence and accuracy

We processed all our experiment on the NVIDIA DGX-1[5] deep learning server from the Department of Mathematics of Roma Tre University occupied by 2×Intel(R) Xeon(R) Processor E5-2698 v4 and 8× NVIDIA Tesla P100 GPU 16GB HBM2 and 512GB of RAM.
As mentioned before, the *transfer learning* technique saved us some time for the training and debugging. Starting the training from scratch might need more data than we possessed to obtain a better accuracy in a short time. In Fig.5, we give a comparison of the loss and accuracy between the same model with a different initialization of the weights. The models are trained with the same parameters on the same data; we notice that the accuracy is good but the zero-knowledge model needs more tuning as the loss seems to be trapped in a local minimum.

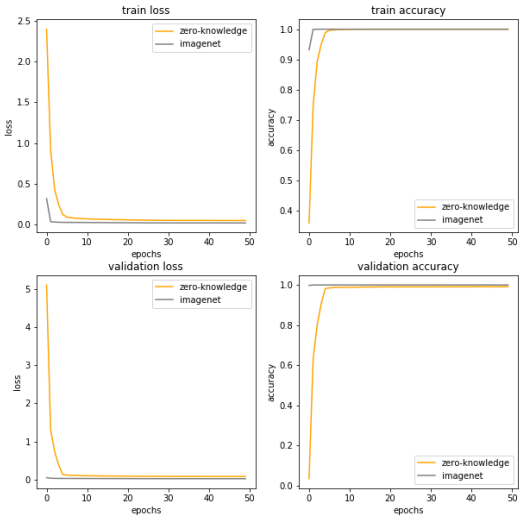

Figure 5. Comparison between a warm-up 50 epochs of the model initialized by the ImageNet and random weight.

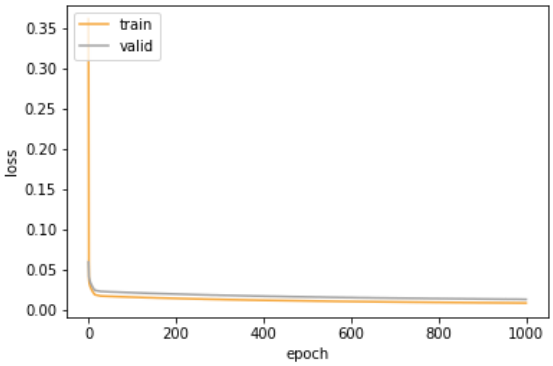

Figure 6. Convergence of the loss in 1000 epochs (ImageNet weight).

In Fig.6, we present the convergence of the cross-entropy

[5]https://www.nvidia.com/en-us/data-center/dgx-1/

during the training measured in 1000 epochs, on 46 classes. The model was initialized by the trained weight from ImageNet (this weight was already trained on less classes). It took around 32 hours to obtain a score of 0.9988086 with loss 0.013294 from a separate test data.

4.2 Profile

Having the said powerful resources, we provide a time and memory profiling of the overall processes we followed.

Pre-processing. The augmentation of the data required much computational power and time. We generated 2000 images per each classes. It took around 155 seconds for 80 CPU cores and required 40 GB of RAM. For a total of 46 monuments, the augmentation of the data took around 2 hours. The cleaning process for the duplicate pictures was done quickly. When the remaining data were enough for the training, we passed to the next steps.

Training. For the training we used one graphic card Tesla P100 of 16GB DRAM memory in which by default TensorFlow allocates all the 16GB. The input parameters used during the training depends on the data and the performance of the model, for instance the batch-size must be more than or equal to the number of the output labels while the number of epoch is arbitrary. For example, for 39 classes we fed around 390 batches per epoch and we ran two models for a given number of epochs. The weights from the most accurate were used in the fine-tuning.

4.3 App structure

As described in a previous work (Palma, 2019) the prototype app was developed using the Xcode development environment and the Swift programming language, and it runs on iOS platforms[6]. The app’s interface was designed aiming at ease of use, and it features two main views (Fig.7). The first view presents the camera monitor and allows the user to frame a monument inside a squared target. Pictures from the camera are continuously processed in the background. When the DL algorithm recognizes a monument, if accuracy exceeds a predefined threshold, the name of the monument is shown. The user can then touch inside the square and enter the second view. It displays information on the monuments, including name, time coverage, pictures and texts. The detail view can also show 3D models that the user can interactively explore.

When internet connection is available, each time the app is launched the information is downloaded or updated from web. The app connects to the web repository Cult, developed and maintained at the University of Padova[7]. The Cult database is available through a web interface which allowed us to upload monument entries and related documents and metadata. The app can access the server through a web application programming interface. After a first connection, the app can work offline exploiting the lightweight DL model which is stored on the device.

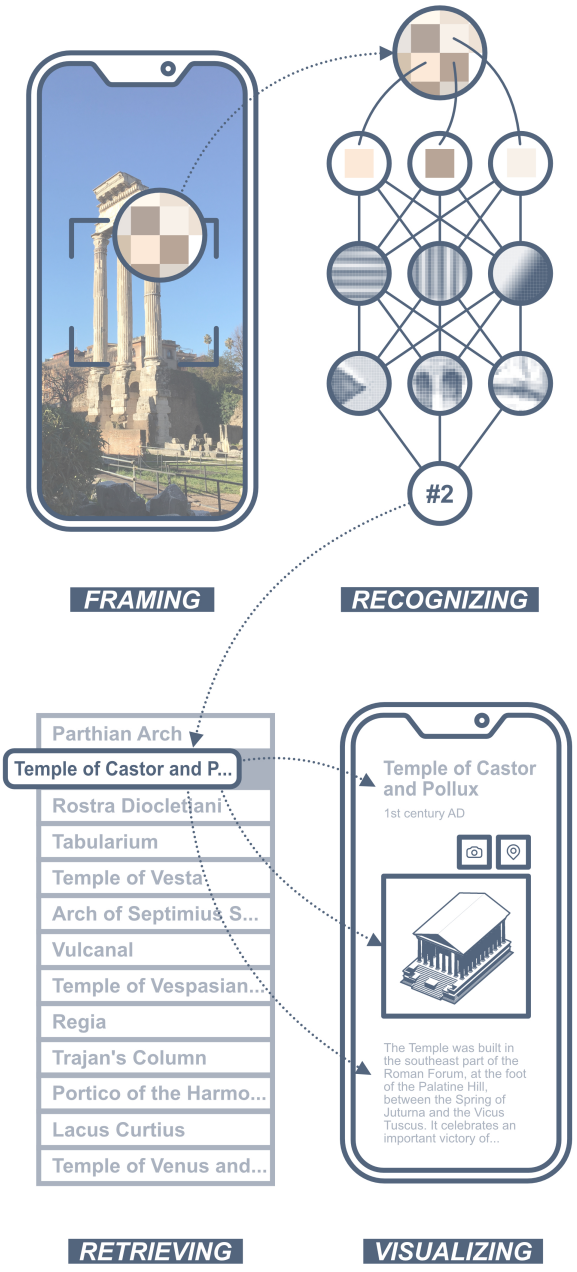

Figure 7. Scheme of the connection between the main view of the app interface and the monument detail view.

5. CONCLUSIONS

As we highlighted in the introduction, the AI is changing the way we interact with things, in particular with the city. Search engines for information are still based mainly on texts, partially on images and sounds, and operate on general and unorganized datasets. Libraries, archives, academic studies and researches, constitute a framework with a logical structure that is not generally accessible to non-experts. In the presence of an archaeological or architectural art object, however, the users search for cultured or academic information that is general not easily available.

The Italian architectural and archaeological heritage represents an immense mine of information, studies, research or simply stories that deserve to be accessed in a simple way starting from real objects. The creation of information sets indexed by real objects seems to be one of the possible natural evolution of a paradigm that has remained stable in the last thirty years.

This approach has the additional advantage of bringing people away from virtual environments by putting closer to the real monuments that become a set of entry points for a labyrinth of knowledge on the history and culture.

The traditional archaeological guides, even when they are

[6] A simpler prototype was developed for Android platforms using the Android Studio development environment, for the main purpose of testing the coverted inference model.

[7] The website and database were developed at the Department of Civil, Architectural and Environmental Engineering at the University of Padova as part of the Tu-CULT project, and it is maintained by the ReLOAD lab.

reorganized according to new technologies, remain prescriptive and rather intrusive objects. On the contrary, the model proposed in this paper is a sort of cultured automaton, a wise friend who, consulted in the right place, can lead the user on a cultural path that can radically change her the level of knowledge. From this point of view the tool described in these pages can become a useful educational tool.

The possibility of changing our awareness and knowledge of ancient and modern architecture can be realized by a smart and pragmatic use of artificial intelligence technologies. At the same time, the mixing of the disciplinary fields is also the starting point of a cultural transformation that no longer contemplates any difference between the humanistic, scientific and artistic disciplines.

ACKNOWLEDGMENTS

The team working on the project at F*U*L*L* | the Future *Urban Legacy* Lab of Politecnico di Torino is also composed by Matteo Robiglio (full professor at the Department of Architecture and Design and project manager at F*U*L*L*), Claudio Casetti (associate professor at the Department of Control and Computer Engineering and project coordinator) and Francesca Frassoldati (associate professor at the Department of Architecture and Design). Désirée Adiutori from the Department of Computational Sciences of the University of Roma Tre contributed to develop the Android prototype app.

REFERENCES

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D. G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., Wicke, M., Yu, Y., Zheng, X., 2016. Tensorflow: A system for large-scale machine learning. *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, OSDI’16, USENIX Association, Berkeley, CA, USA, pp. 265-283.

Amato, G., Falchi, F., Vadicamo, L., 2016. Visual Recognition of Ancient Inscriptions Using Convolutional Neural Network and Fisher Vector. *Journal on Computing and Cultural Heritage*, 9(4), pp. 1-24.

Amit, D. J., 1992. *Modelling Brain Function: The World of Attractor Neural Networks*. 1st edn, Cambridge University Press, New York, NY, USA.

Araujo, A., Weyand, T., 2018. Google-Landmarks: A New Dataset and Challenge for Landmark Recognition. http://ai.googleblog.com/2018/03/google-landmarks-new-dataset-and.html. (23 June 2019).

Bloice, M. D., Roth, P. M., Holzinger, A., 2019. Biomedical image augmentation using Augmentor. *Bioinformatics*. https://doi.org/10.1093/bioinformatics/btz259.

Cao, B., Weyand, T., 2019. Announcing Google-Landmarks-v2: An Improved Dataset for Landmark Recognition & Retrieval. http://ai.googleblog.com/2019/05/announcing-google-landmarks-v2-improved.html. (23 June 2019).

Chollet, F. et al., 2015. Keras. https://github.com/fchollet/keras. (23 June 2019).

Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., Schiele, B., 2016. The Cityscapes Dataset for Semantic Urban Scene Understanding. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Las Vegas, NV, USA, pp. 3213-3223.

Cybenko, G., 1989. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4), pp. 303-314. https://doi.org/10.1007/BF02551274.

D’Autilia, R., Guerra, F., 1991. Qualitative aspects of signal processing through dynamic neural networks. pp. 447-462.

Gada, S., Mehta, V., Kanchan, K., Jain, C., Raut, P., 2017. Monument Recognition Using Deep Neural Networks. *2017 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC)*, IEEE, Coimbatore, pp. 1-6.

Goodfellow, I., Bengio, Y., Courville, A., 2016. *Deep Learning*. MIT Press.

Hebb, D. O., 1949. *The organization of behavior: A neuropsychological theory*. Wiley, New York.

Hopfield, J. J., 1982. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79(8), pp. 2554-2558. https://www.pnas.org/content/79/8/2554.

Hornik, K., 1991. Approximation Capabilities of Multilayer Feedforward Networks. *Neural Netw.*, 4(2), pp. 251-257. http://dx.doi.org/10.1016/0893-6080(91)90009-T.

Hosny, A., Parmar, C., Quackenbush, J., Schwartz, L. H., Aerts, H. J. W. L., 2018. Artificial intelligence in radiology. *Nature Reviews Cancer*, 18(8), pp. 500-510. http://www.nature.com/articles/s41568-018-0016-5.

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H., 2017. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *CoRR*, abs/1704.04861. http://arxiv.org/abs/1704.04861.

Ioffe, S., Szegedy, C., 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. F. Bach, D. Blei (eds), *Proceedings of the 32nd International Conference on Machine Learning*, Proceedings of Machine Learning Research, 37, PMLR, Lille, France, pp. 448-456.

LeCun, Y., 1989. *Generalization and network design strategies*. Elsevier.

McCulloch, W. S., Pitts, W., 1943. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4), pp. 115-133. https://doi.org/10.1007/BF02478259.

Mezard, M., Parisi, G., Virasoro, M. A., 1987. *Spin Glass Theory and Beyond*. Lecture Notes in Physics Series, World Scientific.

Palma, V., 2019. Towards deep learning for architecture: a monument recognition mobile app. *ISPRS - International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLII-2/W9, pp. 551-556. https://doi.org/10.5194/isprs-archives-XLII-2-W9-551-2019.

Rawat, W., Wang, Z., 2017. Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review. *Neural Computation*, 29, pp. 1-98.

Stathopoulou, E. K., Remondino, F., 2019. Semantic photogrammetry: boosting image-based 3D reconstruction with semantic labeling. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLII-2/W9, pp. 685-690.

Sutskever, I., Hinton, G. E., 2008. Deep, Narrow Sigmoid Belief Networks Are Universal Approximators. *Neural Computation*, 20(11), pp. 2629-2636. https://doi.org/10.1162/neco.2008.12-07-661.

Webb, S., 2018. Deep learning for biology. *Nature*, 554(7693), pp. 555-557. http://www.nature.com/doifinder/10.1038/d41586-018-02174-z.