HERITAGE IDENTIFICATION OF MONUMENTS USING DEEP LEARNING TECHNIQUES

Submitted by

Nisha Devi S(21MDB005)

in partial fulfillment for the award of the degree of

MASTER OF SCIENCE
Data Science & Business Analysis

DEPARTMENT OF COMPUTER SCIENCE

RATHINAM COLLEGE OF ARTS AND SCIENCE

(AUTONOMOUS)

COIMBATORE - 641021 (INDIA)

May - 2023

RATHINAM COLLEGE OF ARTS AND SCIENCE

(AUTONOMOUS)

COIMBATORE - 641021

BONAFIDE CERTIFICATE

This is to certify that the Phase-I report entitled **Heritage Identification Of Monuments Using Deep Learning Techniques** submitted by **Nisha Devi S**, for the award of the Degree of Master of Computer Science specialization in **“Data Science & Business Analysis”** is a bonafide record of the work carried out by her under my guidance and supervision at Rathinam College of Arts and Science, Coimbatore

Mrs.V.Kanimozhi M.E.,(Ph.D)	**Mr.Siva Prakash M.Tech., (Ph.D)**
Supervisor	Mentor

Submitted for the University Examination held on 20.04.2023

INTERNAL EXAMINER **EXTERNAL EXAMINER**

RATHINAM COLLEGE OF ARTS AND SCIENCE

(AUTONOMOUS)

COIMBATORE - 641021

DECLARATION

I am **Nisha Devi S**, hereby declare that this thesis entitled **"Heritage Identification Of Monuments Using Deep Learning Techniques",** is the record of the original work done by me under the guidance of **Mrs.V.Kanimozhi M.E.,(Ph.D)**, Faculty Rathinam college of arts and science, Coimbatore. To the best of my knowledge this work has not formed the basis for the award of any degree similar award to any candidate in any University.

Signature of the Student:

Nisha Devi S

Place: Coimbatore

Date: 09.05.2023

COUNTERSIGNED

Mrs.V.Kanimozhi M.E.,(Ph.D)

Supervisor

Contents

Acknowledgement

On successful completion for project look back to thank who made in possible. First and foremost, thank **"THE ALMIGHTY"** for this blessing on us without which we could have not successfully our project.I are extremely grateful to **Dr.Madan.A. Sendhil, M.S., Ph.D.,** Chairman, Rathinam Group of Institutions, Coimbatore and **Dr. R.Manickam MCA., M.Phil., Ph.D.,** Secretary, Rathinam Group of Institutions, Coimbatore for giving me opportunity to study in this college.I am extremely grateful to **Dr.R.Muralidharan, M.Sc., M.Phil., M.C.A., Ph.D.,** Principal Rathinam College of Arts and Science(Autonomous), Coimbatore.Extend deep sense of valuation to **Mr.A.Uthiramoorthy, M.C.A., M.Phil., (Ph.D),** Rathinam College of Arts and Science (Autonomous) who has permitted to undergo the project. Unequally I thank **Mr.P.Sivaprakash, M.Tech., (Ph.D).,** Mentor and **Dr.Mohamed Mallick, M.E., Ph.D.,** Project Coordinator, and all the Faculty members of the Department - iNurture Education Solution pvt ltd for their constructive suggestions, advice during the course of study.I convey special thanks, to the supervisor **Mrs.V.Kanimozhi M.E.,(Ph.D)** who offered their inestimable support, guidance, valuable suggestion, motivations, helps given for the completion of the project. I dedicated sincere respect to my parents for their moral motivation in completing the project.

List of Figures

List of Tables

Abstract

The categorization of photographs captured during architectural asset assessment is a critical job in the digital recording of cultural resources. Because many photos are often processed, categorization is a time-consuming (and hence error-prone) job that takes a long time. The availability of automated solutions to aid these sorting chores would greatly enhance a key aspect of digital documentation. Furthermore, precise picture categorization allows for better administration and more efficient searches via particular phrases, assisting in researching and understanding the historical object in the issue. The major goal of this paper is to use deep learning (RESNet18) methods for categorizing photographs of architectural history, especially using convolutional neural networks. The usefulness of training these networks from scratch vs. just fine tweaking pre-trained networks is assessed for this purpose. This has been used to categorize items of interest in pictures of architectural heritage structures. Because no datasets appropriate for network training have been found, a new dataset has been constructed and made public. In terms of accuracy, promising results have been produced, and it is believed that implementing these approaches might considerably contribute to the digital recording of architectural history.

Chapter 1

Introduction

In today's fast-paced world, it is essential to preserve the rich and diverse cultural and historical heritage of the world. Archaeologists and historians have spent a significant deal of time and effort researching the various monuments and architectural styles by traveling to the locations and making first-hand observations. The classification of monuments, the segmentation of particular architectural styles, and many other applications of computer vision techniques are now being utilized to examine the monuments. These methods expand on their work and streamline and scale up a portion of the process. Monument categorization is the process of identifying and classifying photographs of monuments into sub-categories based on their architectural design.

The more general topic of landmark identification includes the categorization and acknowledgment of monuments. Even though landmark recognition is a well-researched topic of computer vision, recognizing monuments is challenging. This is brought on by a variety of problems, such as the scarcity of annotated datasets of monuments in non-English speaking regions, subtle changes in the architectural styles of monuments, and image samples with different perspectives, resolutions, lighting, scales, and viewpoints.

These important barriers significantly increase the difficulty of monument recognition in a diverse nation like India. Automatic monument designation has advantages in many areas, including but not limited to educational, historical, conservation, and tourist attractions.

1.1 Problem statement

Recent years have seen a rapid development of deep learning algorithms due to the widespread use of large image datasets and previously unheard-of computing power (DL). Convolutional neural networks (CNNs) have become one of the most used DL methods in computer vision, with applications in many different industries. This study describes an ongoing investigation into CNN techniques in the field of architectural heritage, a yet under-researched subject. The development of a smartphone app to identify monuments' first procedures and results is explained. While AI is only beginning to interface with the built world through mobile devices, heritage technologies have long produced and examined digital models and spatial archives.

1.2 Existing system

The classification of photos taken while measuring an architectural asset is a crucial task in the digital documentation of cultural assets. Classifying images is a tedious process that frequently requires a lot of time due to the regular handling of many photos (and is consequently prone to mistakes) The availability of automatic methods to make these sporting activities simpler would improve a crucial step in the digital documenting process. Also, correct categorization of the accessible pictures provides better management and more efficient searches using specific terms, aiding in the duties of investigating and interpreting the disputed heritage item.

The major objective of this research is to categorize photos of a historically significant building using deep learning techniques, notably convolutional neural networks. the effectiveness of creating these networks from scratch vs simply optimizing ones that have already been created is evaluated. All of this has been utilized to group intriguing details in images of buildings with significant architectural heritage. No datasets of this kind that are suitable for network training exist, hence a new dataset has been created and made available. It is believed that the application of these methodologies can significantly contribute to the digital documentation of architectural history. In terms of accuracy, promising findings have been generated.

1.3 Main objectives and contributions

In this article, we apply deep learning techniques to the classification of images for the documentation of the architectural cultural heritage. We specifically study the use of the most representative convolutional neural networks to extract useful information from images. The main objective is to verify the real usefulness of some of these networks that set the current state of the art for their application in the classification of images of heritage buildings. There is much literature of different applications of deep learning in the classification of images, both generic [12–18], and specific, such as aerial images [19,20], medical images [21], license plate and vehicle recognition [22], gait recognition [23], classification of microorganisms [24], recognition of the urban environment [25], fruit recognition [26] and many more. There is also literature concerning the classification of images of architectural heritage, but using other techniques such as pattern detection [27], instance retrieval [28], Gabor filters and support vector machine [29], computer vision algorithms [11], clustering and learning of local features [30], hierarchical sparse coding of blocklets [31], or Multinomial Latent Logistic Regression [32]; but no references concerning the classification of images of architectural heritage are known using deep learning, except a prior publication of the authors [33]. Another of the topics studied is the evaluation of whether, for these tasks, it is better to train a network from scratch (full training) or to use fine tuning of a pre-trained network. There is a reference that analyzes this specific aspect in medical images [21] and something similar in images of food [34,35], but no bibliography in the field of architectural heritage is known. There are few datasets of images in the realm of historical heritage (to date only one of architectural styles has been found [32]), which has motivated the creation of one of our own focused on architectural elements. This dataset aims to be the origin of a system which could be useful for the training of neural convolutional networks or other techniques of classification in this type of task. The absence of datasets prevents the comparison between different techniques and works, which is why, by way of comparison, these techniques have also been evaluated using the indicated dataset of architectural styles in order to verify that the results obtained are superior to other classical classification methodologies. In the labeling of images considered in this article, the goal is to automatically deduce the main element to be reflected in each image. In this way, it is possible to organize the available images in different categories and facilitate the work of the specialists or enthusiasts who search through the collection by consulting very precise keywords. The applications of these techniques in documentation tasks are many: web and mobile applications to access and consult details of a heritage building, study of the different elements of a building in different geographical areas and different times, study of their evolution, development of systems that could be trained to detect historical periods or architectural styles, reveal pathologies, find alterations or previous interventions, search for similar images in other sites, etc.

1.4 Literature Review

Researchers have been studying landmark classification, a subset of monument classification, over the last few decades. They have used a range of approaches that can either be global feature-based or local feature-based. Edges, textures, and colours are among the most fundamental and resource-intensive global properties. Linde et al. [2] demonstrated the superiority of higher-order composite field histograms by performing efficient computation on sparse matrices. Ge et al. [3] demonstrated a covariance descriptor-based approach using a Support Vector Machine (SVM) for the combined classifier and voting technique. Contextual priming can be utilized to use scene information for object detection, as demonstrated by Torralba et al.'s[4] use of visual context for location recognition and categorization. In 2020, a novel approach that made use of an ensemble of subcentre ArcFace models [5] with dynamic margins and just global features was successful in winning the Google Landmark Recognition challenge on the GLDv2 dataset [6]. Due to their lack of granularity and inability to focus on Regions of Interest, global features are often used in conjunction with local features to solve object detection problems in general, such as monument detection. (ROIs) Local features are focused on Points of Interest (POI) or Regions of Interest and are robust to partial occlusion, illumination fluctuation, and changes in viewpoint. (ROI). Common approaches include affine-invariant features [8] and scale-invariant Feature Transform (SIFT) [7]. These techniques commonly model the visual words that are grouped and represent local qualities using a Bag-of-Words (BoW) model [9,10]. Numerous such methods have been put forth, such as the use of a probability density response map to assess the likelihood of local patches [11], the estimation of patch saliency using contextual data [12], the estimation of patch importance using non-parametric density estimation [13], spatial pyramid kernel-based BoW methods (SPK-BoW) [14,15], and scalable vocabulary trees [16].

Chapter 2

API building Flask and Streamlit

2.1 Flask

Flask was created by Armin Ronacher of Pocoo, an international group of Python enthusiasts formed in 2004[7]. In April 2016, the Pocoo team was disbanded, and development of Flask and related libraries passed to the newly formed Pallets project [8],[9]. Flask is a python-based net utility framework. Use flask to enhance small websites. In easy language it enables quit users to interact along with your python code (in this situation our ML fashions) immediately from their internet browser without needing any library code documents, etc. Flask is very easy to carry out Restful API's using python. Flask permits you to create net applications very effortlessly, that is why it allows you to consciousness more on different crucial factors of the ML lifestyles cycle such as EDA, engineering issues, etc. It has many modules that make it smooth for an internet developer to write down programs without having to fear approximately details like protocol control, cable control, and so forth. Flask offers a spread of options for growing net programs and provides us with the essential equipment and libraries that permit us to build an internet utility.

Ease of Flask Framework: In the current scenario machine learning is a widely used domain in the web application for various purposes. Flask is the framework that is python based micro-framework and It is used for developing small scale website building. As it is a micro-framework means little to no dependencies to external libraries. It is very easy to use and to make the Rest full API's using python. Using a flask framework, we can host local, cloud etc. In the machine learning Prediction model, mostly Flask Framework is used for its deployment.

Features: Flask provides integrated aid for unit testing.

Deep learning is a field of machine learning that uses neural networks with multiple layers to learn and make predictions on complex datasets. Deep learning has been widely used in various applications, such as image recognition, natural language processing, and speech recognition.

To build a deep learning-based application using Flask, you need to have a good understanding of both Flask and deep learning concepts. Here are the general steps involved:

1) Create a Flask web application: You can create a new Flask application using the Flask CLI or manually by creating a new Python file and importing the Flask module. Define the application routes, which will handle incoming requests and return responses.

2) Install deep learning libraries: To use deep learning in your Flask application, you need to install relevant libraries such as TensorFlow, Keras, PyTorch, or scikit-learn.

You can use pip to install these libraries.

3) Build the deep learning model: Define the architecture of the neural network, choose an appropriate loss function, and optimizer. Train the model on a dataset or use a pre-trained model.

4) Integrate the model into the Flask application: Once the model is trained, you can integrate it into your Flask application. Define a new route that will handle the input data, preprocess it, and pass it through the model to make predictions.

5)Test and deploy: Test your application locally to ensure that it's working correctly. Once you're satisfied with the application's performance, deploy it to a web server or cloud platform.

Some tips for building a successful deep learning-based Flask application:

* Use appropriate libraries for your specific task and data.

* Optimize the model's architecture and parameters for better performance.

* Make sure to preprocess and normalize input data before feeding it into the model.

* Handle errors and exceptions gracefully to provide better user experience.

* Secure your application by using secure communication protocols and protecting sensitive data.

2.2 Streamlit Framework

Another Framework That we can use for the Machine learning model deployment is the Streamlit framework. It is also an open-source framework for building web apps for Data science and Ml. It will allow writing a code as good as writing a python code. Adrien Treuille, Thiago Teixeira, and Amanda Kelly created “Streamlit” [2]. Streamlit is user-friendly. It is the faster way to build data apps and share them according to the streamlit founders. If you are enthusiasts about machine learning and you do not want to waste your time building a web app for the Ml model deployment, then streamlit is important.it is built to interact with the data in the model.

Ease of Streamlit Framework: A streamlit framework is also a popular tool for creating user interfaces. Powerful custom web pages can develop using this open-source python library for machine learning and Data Science. It is compatible with major Python libraries such as Scikit-learn, Keras, PyTorch, SymPy(latex), NumPy, pandas, Matplotlib [3]. Streamlit allows you to use the HTML code directly within the Python file. For the front-end UI development, streamlit does not essentially require CSS Formatting and different templates. It is the best lightweight technology, purely based on python. Creating a complex application on streamlit suggest creating a separate folder for the templates and styles guides. It can do everything from loading a model to creating a model to front-end UI development. **Features:** Fully based on python

Chapter 3

Dataset Description

Architectural Heritage Elements Dataset (AHE) is an image dataset for developing deep learning algorithms and specific techniques in the classification of architectural heritage images. This dataset consists of 10235 images classified in 10 categories: Altar: 829 images; Apse: 514 images; Bell tower: 1059 images; Column: 1919 images; Dome (inner): 616 images; Dome (outer): 1177 images; Flying buttress: 407 images; Gargoyle (and Chimera): 1571 images; Stained glass: 1033 images; Vault: 1110 images. It is inspired by the CIFAR-10 dataset but with the objective in mind of developing tools that facilitate the tasks of classifying images in the field of cultural heritage documentation. Most of the images have been obtained from Flickr and Wikimedia Commons (all of them under creative commons license).

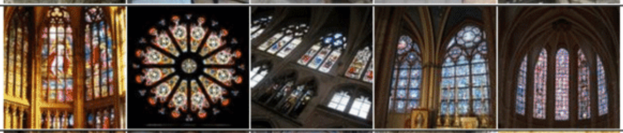
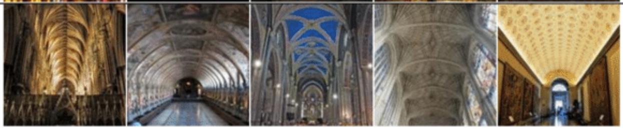
Category	Examples
Altar (829 images)	
Apse (514 images)	
Bell tower (1059 images)	
Column (1919 images)	
Dome (inner) (616 images)	
Dome (outer) (1177 images)	
Flying buttress (407 images)	
Gargoyle (and Chimera) (1571 images)	
Stained glass (1033 images)	
Vault (1110 images)	

Figure 3.1: Overview of the dataset

Chapter 4

Methodology

Deep learning is actually a subset of machine learning. It technically is machine learning and functions in the same way but it has different capabilities. The main difference between deep and machine learning is, machine learning models become well progressively but the model still needs some guidance.A branch of machine learning called deep learning appeared. The popularity of machine learning and the development of the computing capacity of computers enabled this new technology. Deep learning as a concept is very similar to machine learning but uses different algorithms. While machine learning works with regression algorithms or decision trees, deep learning uses neural networks that function very similarly to the biological neural connections of our brain.

If a machine learning model returns an inaccurate prediction then the programmer needs to fix that problem explicitly but in the case of deep learning, the model does it by him.The first advantage of deep learning over machine learning is the needlessness of the so-called feature extraction. Automatic car driving system is a good example of deep learning.Deep learning that enable the network to learn from unsupervised data and solve complex problems. Deep Learning approaches such as Convolutional Neural Network, Auto Encoder, Deep Belief Network, Recurrent Neural Network, Generative Adversal Network and Deep Reinforcement Learning are the algorithms used in Deep Learning.In our project we are using Convolutional Neural Networks.

4.1 Convolution Neural Network

One of the most popular deep neural networks is Convolutional Neural Networks. It is a class of deep neural networks, most commonly applied to analyze visual imagery and specializes in processing data that has a grid-like topology, such as an image. A digital image is a binary representation of visual data.Convolutional neural networks are composed of multiple layers of artificial neurons.The main advantage of CNN compared to its predecessors is that it automatically detects the important features without any human supervision.It can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other.The main advantage of CNN compared to its predecessors is that it automatically detects the important features without any human supervision. CNN utilizes spatial correlations which exist with the input data. Each concurrent layer of the neural network connects some input neurons.

4.1.1 Convolution Neural Network in image processing

A convolutional neural network is a type of artificial neural network used in image recognition and processing that is specifically designed to process pixel data.CNNs are fully connected feed forward neural networks. CNNs are very effective in reducing the number of parameters without losing on the quality of models. Images have high dimensionality (as each pixel is considered as a feature) which suits the above described abilities of CNNs.It as a machine learning algorithm that can take in an input image, assign importance weights and biases to various objects in the image, and then can differentiate one from the another.It works by extraction features from the image. Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

We use Convolutional Neural Network and Deep Learning based yolo for Real Time Detection and Recognition of Human Faces, which is simple face detection and recognition system is proposed in this paper which has the capability to recognize human faces in single as well as multiple face images in a database in real time with masks on or off the face. Pre-processing of the proposed frame work includes noise removal and hole filling in colour images. After pre-processing, face detection is performed by using CNNs architecture. Architecture layers of CNN are created using Keras Library in Python. Detected faces are augmented to make computation fast. By using Principal Analysis Component (PCA) features are extracted from the augmented image. For feature selection, we use Sobel Edge Detector.

4.1.2 The Input Image

Real-time input images are used in this proposed system. Face of person in input images must be fully or partially covered as they have masks on it. The system requires a reasonable number of pixels and an acceptable amount of brightness for processing. Based on experimental evidence, it is supposed to perform well indoors as well as outdoors i.e. passport offices, hospitals, hotels, police stations and schools etc.

4.1.3 The Pre-processing Stage

Input image dataset must be loaded as Python data structures for pre-processing to overturn the noise disturbances, enhance some relevant features, and for further analysis of the trained model. Input image needs to be pre-processed before face detection and matching techniques are applied. Thus pre-processing comprises noise removal, eye and mask detection, and hole filling techniques. Noise removal and hole filling help eliminate false detection of face/ faces. After the pre-processing, the face image is cropped and re-localised. Histogram Normalisation is done to improve the quality of the pre- processed image.

4.1.4 The Face Detection Stage

We perform face detection usingHAAR Cascade algorithm.This system consists of the value of all black pixels in greyscale images was accumulated. They then deducted from the total number of white boxes. Finally, the outcome is compared to the given threshold, and if the criterion is met, the function considers it a hit.In general, for each computation in Haar-feature, each single pixel in the feature areas can need to be obtained, and this step can be avoided by using integral images in which the value of each pixel is equal to the number of grey values above and left in the image.

4.1.5 The Feature-Extraction Stage

Feature Extraction improves model accuracy by extracting features from pre-processed face images and translating them to a lower dimension without sacrificing image characteristics. This stage allows for the classification of human faces.

4.1.6 The Classification Stage

Principal Component Analysis(PCA) is used to classify faces after an image recognition model has been trained to identify face images. Identifying variations in human faces is not always apparent, but PCA comes into the picture and proves to be the ideal procedure for dealing with the problem of face recognition. PCA does not operate classifying face images based on geometrical attributes, but rather checks which all factors would influence the faces in an image. PCA was widely used in the field of pattern recognition for classification problems.PCA demonstrates its strength in terms of data reduction and perception.

4.1.7 Training Stage

The method is based on the notion that it learns from pre- processed face images and utilizes CNN model to construct a framework to classify images based on which group it belongs to. This qualified model is saved and used in the prediction section later. In CNN model, the stages of feature extraction are done by PCA and feature selection done by Sobel Edge Detector and thus it improves classification efficiency andaccuracy of the training model.

4.1.8 Prediction Stage

In this stage, the saved model automatically detects theoftheface maskimagecaptured by the webcam or camera. The saved model and the pre-processed images are loaded for predicting the person behind the mask. CNN offers high accuracy over face detection, classification and recognition produces precise and exactresults.CNN model follows a sequential model along with Keras Library in Python for prediction of human faces.

4.2 Full Training of AlexNet Network

For the first of the tests carried out, the AlexNet network was chosen, which is a well-known network and widely used in this type of task. It is recognized as the one that led to the resurgence of these techniques. This network was developed by Krizhevsky et al. [16], is a deep CNN architecture and was the winning model in the ImageNet Large Scale

Visual Recognition Challenge (ILSVRC-2012) [50]. In this challenge, the models try to classify the images into 1000 different categories (generic such as “volcano”, “obelisk” or “lemur”). In contrast to earlier CNN models, AlexNet consists of five convolutional layers, of which the first, second, and fifth are followed by pooling layers, and three fully connected layers (a total of approximately 60 million parameters).

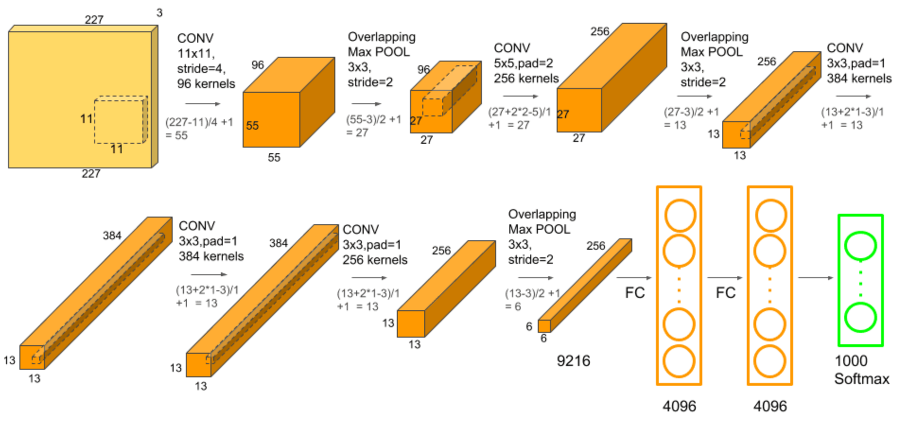


Figure 4.1: Alexnet Architecture

The success of AlexNet is attributed to certain practical solutions, such as Rectified Linear Units (ReLU), data augmentation and dropout. The ReLU, which is simply a half-wave rectifier function such that f(x) = max (x; 0), can significantly accelerate the training phase; Data augmentation is an effective way to reduce over-fitting when training a large CNN, generating more training images by trimming small patches and horizontally flipping those patches; while the dropout technique, which reduces the co-adaptations of neurons by randomly establishing the zero value at the exit of some hidden neurons, is used in fully connected layers to reduce overfitting. In short, the success of AlexNet popularized the application of large CNNs in the tasks of visual recognition, so it has become a classic architecture within the CNNs.

4.3 Full Training of a Residual Network (ResNet)

We also decided to use the original residual network developed by He et al., of Microsoft [15], which has led to a growing adoption of this specific type of network due to its good results. The depth of the networks has a decisive influence on their learning, but adjusting this parameter optimally is a very difficult task. In theory, when the number of layers in a network increases, its performance should also improve. However, in practice, this is not true for two main reasons: the vanishing gradient (many neurons become ineffective/useless during the training of such deep networks); and the optimization of parameters is highly complex (by increasing the number of layers, it increases the number of parameters to adjust, which makes training these networks very difficult, leading to higher errors than in the case of shallower networks).

The residual networks seek to increase the network's depth without such problems affecting the results. The central idea of residual networks is based on the introduction of an identity function between layers. In conventional networks, there is a nonlinear function y = H (x) between layers (underlying mapping), as shown on the left of Figure 5. In residual networks, we have a new nonlinear function y = F (x) + id (x) = F (x) + x, here F (x) is the residual (on the right of Figure 5). This modification (called shortcut connections) allows important information to be carried from the previous layer to the next layers. Doing this avoids the problem of the vanishing gradient.

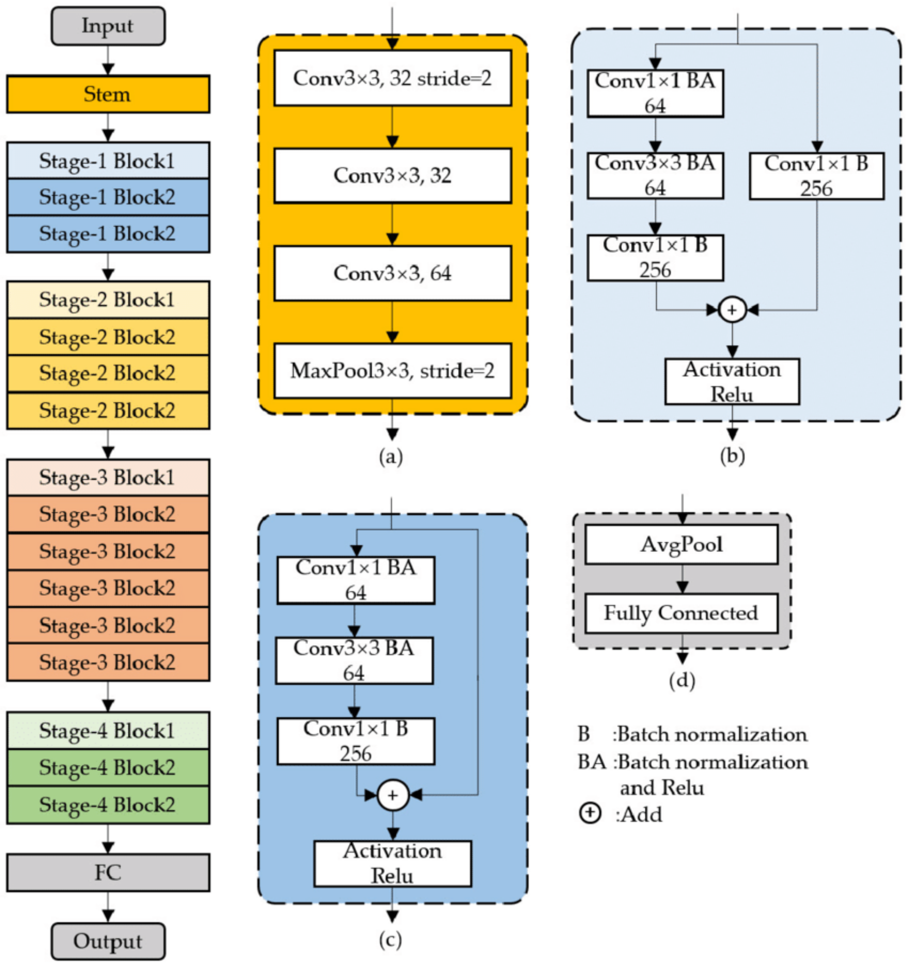


Figure 4.2: Resnet Architecture

Chapter 5

Result and Discussion

In this section the experimental setting is introduced first, to establish the basic idea of the work

Architectural Heritage Elements Dataset (AHEDataset), which generated in three versions: Originally the dataset was published in two versions, first one contains images of different sizes and the second was scaled into 128×128 pixels, as well as the small dataset was created with a small size of 64×64 and 32×32. The third version was selected as a subset of each class which consists of 500 images that scaled into 224 × 224 pixels to be compatible with the pre-trained CNN.The dataset was partitioned randomly into 70% for training and 30% for validation. 2- The pre-trained Convolutional Neural Network (GoogLNet, ResNet 18 and Alexnet) were used during the course of this research to classify the Architectural Heritage Elements Dataset. All the CNN that used were modified to be suitable for those kinds of images. The modification starts from the input layer to be able for multi-channel images, convolutional layer to select size of filters proper the size of the entire images, fully connected layer was edited to be suitable the number of categories in the used dataset, finally the output classification layer was edit based on number of classes of fully connected layer. Our experimental study using AHEDataset original version was trained. Dataset are trained fully by three pre-trained Convolutional Neural Networks such as alexnet ,resnet18 and google net. Table shows a summary of the results based on a different test performed. Consider the dataset with 64×64 image size, the resnet-18 achieved the highest accuracy of 93.4 at the same learning rate and iteration in comparison to other techniques. In another case, the highest accuracy achieved is 91 based on alexnet under the same conditions for images with a size of 128×128.

Pretrained CNN algorithm	Image size	Epoch	Validation iteration	Training iteration	Learning rate	Accuracy
Resnet 18	64*64	20	98	588	0.0003	93.4
Alexnet	64*64	20	98	588	0.0003	91.0
Googlenet	64*64	20	98	588	0.0003	87.4

Table 5.1: Accuracy table

Figure show samples of a testing image with Predicted accuracy for each sample of an original dataset based on ResNet-18 respectively.

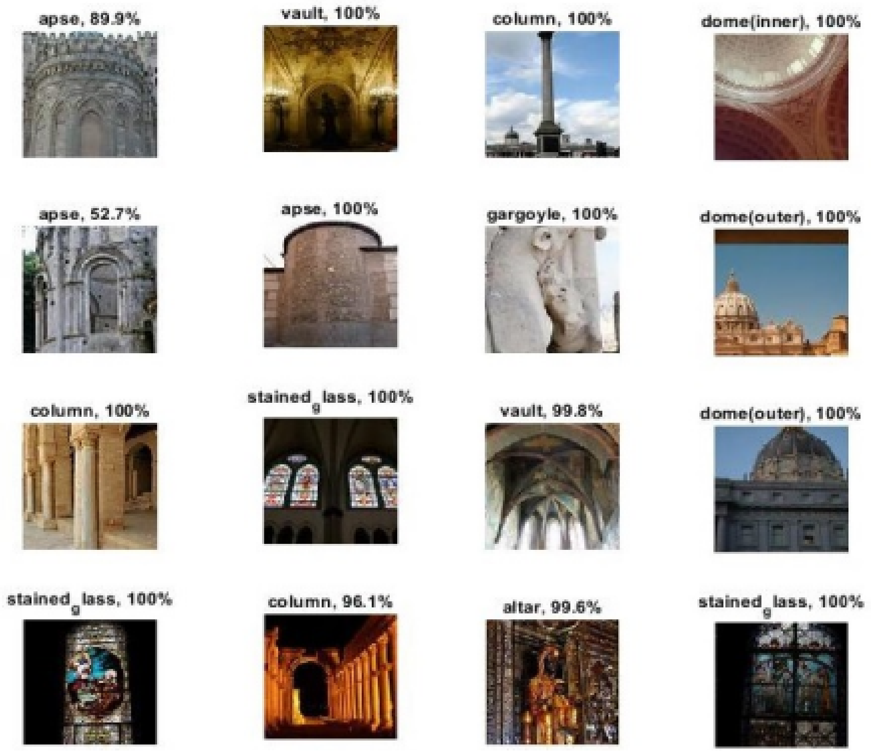


Figure 5.1: Sample of Predicted accuracy based on ResNet-18

Chapter 6

API Building

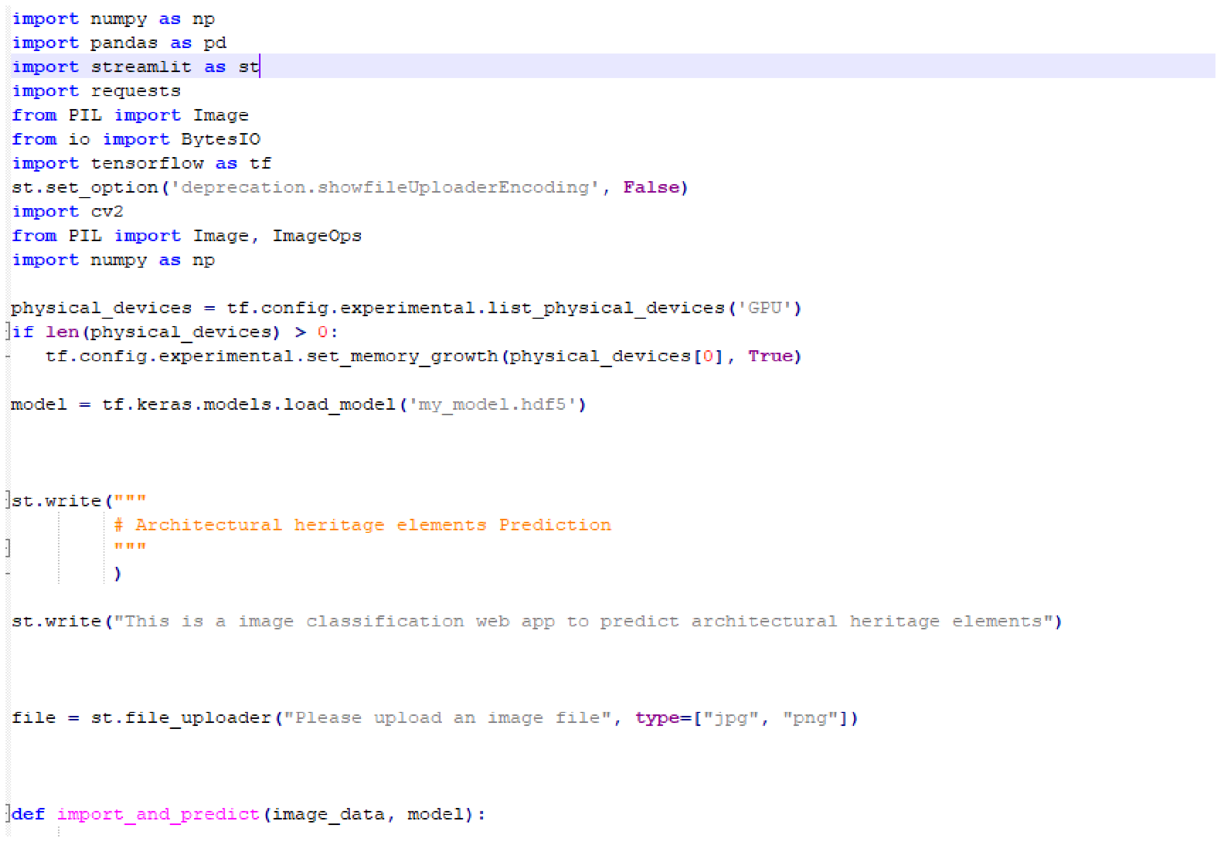
6.0.1 Backend

In this section, file name Python file model.py is used to classify the depression result.The model is dump into machine learning and converted into a pickle filr. In Python, pickling refers to storing a variable into a .pkl file.

The above figure syntax belongs to streamlit, the web framework we will be using. Running this HTML with Flask later on, will yield different outputs.

Clearly, this design is unsuitable for deployment seeing how vanilla it is. You can use your own design for this tutorial, preferably one that suits your model purposes and your organisation. Here are some things to watch if you want to use your own design

Make sure you have a form with method = "POST" . Remember the names of the input bars. For conformity between us, I suggest using name = "Enter input" and name = "View Output" for the inputs.



```
import numpy as np
import pandas as pd
import streamlit as st
import requests
from PIL import Image
from io import BytesIO
import tensorflow as tf
st.set_option('deprecation.showfileUploaderEncoding', False)
import cv2
from PIL import Image, ImageOps
import numpy as np

physical_devices = tf.config.experimental.list_physical_devices('GPU')
if len(physical_devices) > 0:
   tf.config.experimental.set_memory_growth(physical_devices[0], True)

model = tf.keras.models.load_model('my_model.hdf5')



st.write("""
         # Architectural heritage elements Prediction
         """
         )

st.write("This is a image classification web app to predict architectural heritage elements")



file = st.file_uploader("Please upload an image file", type=["jpg", "png"])



def import_and_predict(image_data, model):
```

Figure 6.1: Index File

6.0.2 Frontend

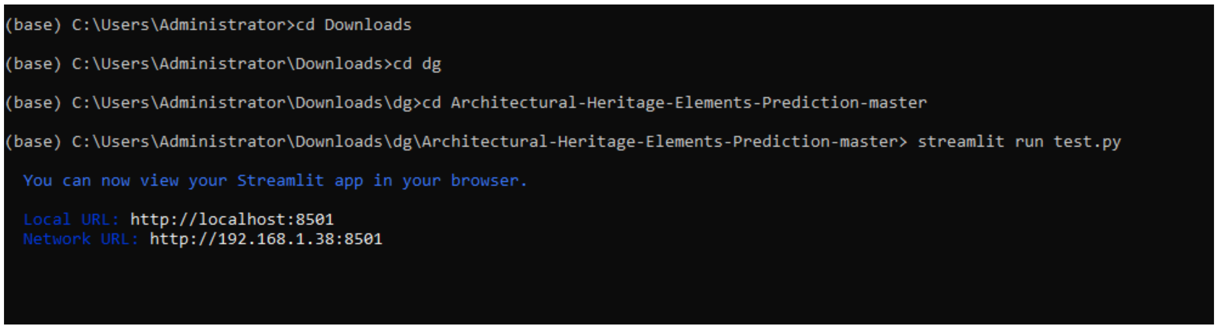
The main function should only run if we receive a POST request from the HTML. We can impose this condition by looking at the condition request.method == ”POST”.

If we receive a POST request, we want to get the values from the inputs. We can achieve this through the code, request.form.get(”name”). Where ”name” is the name of the input bars in the HTML

To call on the model we pickled earlier, we use the function joblib.load(”clf.pkl”).

To render an HTML file with Flask syntax, we use the function rendertemplate(”file.HTML”).

Running script



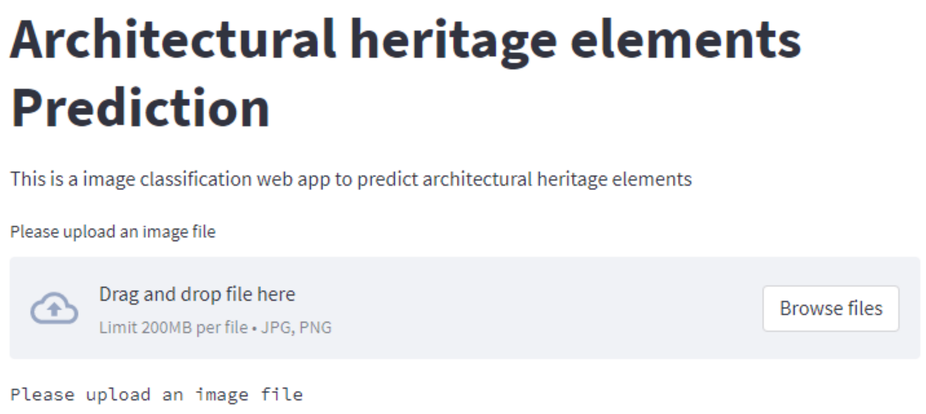
```
(base) C:\Users\Administrator>cd Downloads

(base) C:\Users\Administrator\Downloads>cd dg

(base) C:\Users\Administrator\Downloads\dg>cd Architectural-Heritage-Elements-Prediction-master

(base) C:\Users\Administrator\Downloads\dg\Architectural-Heritage-Elements-Prediction-master> streamlit run test.py

  You can now view your Streamlit app in your browser.

  Local URL: http://localhost:8501
  Network URL: http://192.168.1.38:8501
```

Figure 6.2: Runnig streamlit application

6.0.3 Result

To run the app we just created, go to your terminal or command prompt. You have to set the directory to your project directory. In my case, the folder name is Machine-Learning-Web-App.

In this web portal upload image and give run



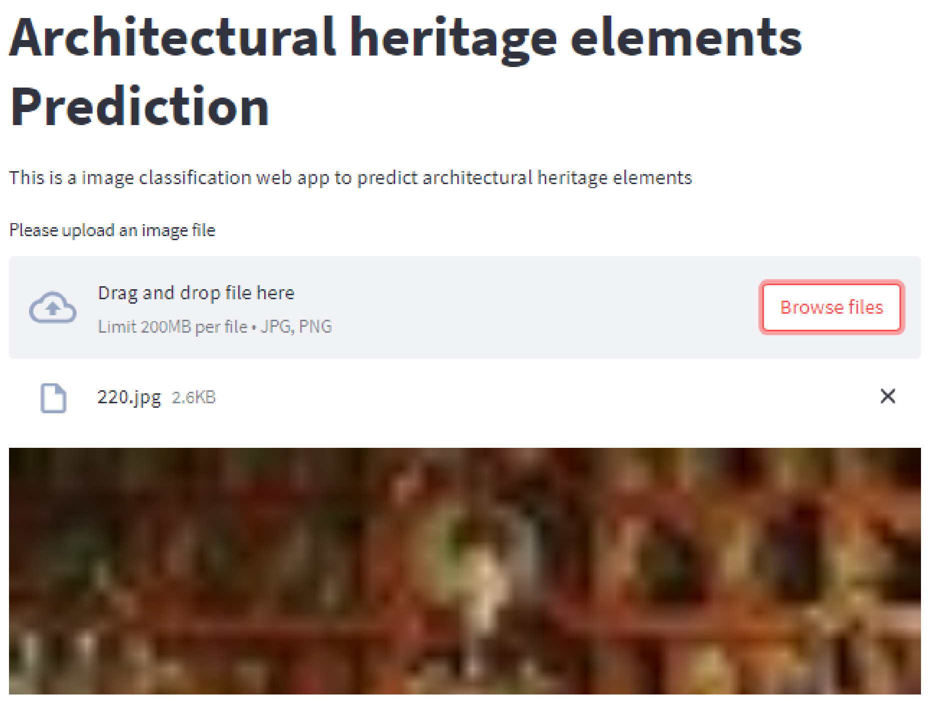
Architectural heritage elements Prediction

This is a image classification web app to predict architectural heritage elements

Please upload an image file

```
Please upload an image file
```

Figure 6.3: Streamlit webpage

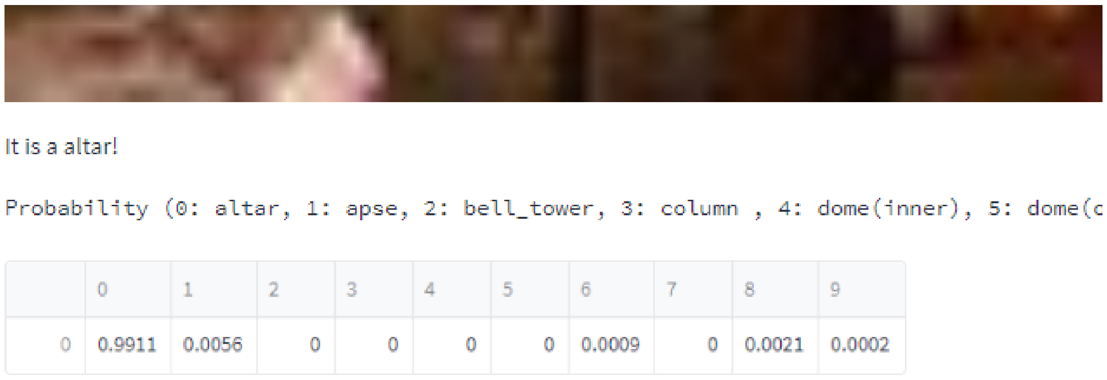


Architectural heritage elements Prediction

This is a image classification web app to predict architectural heritage elements

Please upload an image file

Figure 6.4: Loading Image for testing



It is a altar!

```
Probability (0: altar, 1: apse, 2: bell_tower, 3: column , 4: dome(inner), 5: dome(c
```

	0	1	2	3	4	5	6	7	8	9
0	0.9911	0.0056	0	0	0	0	0.0009	0	0.0021	0.0002

Figure 6.5: Predicting Image

Chapter 7

Conclusion

This work presents a model for architecture heritage image classification using deep learning with a convolutional neural network. In this model, CNN uses a pre-trained structure (GoogLNet , ResNet-18 and Alexnet). The classification results which are achieved by Resnet 18 overperformed the other techniques in comparison with other CNN on dataset. However, ResNet-18 produced better classification results compared with other pre-trained CNN with the highest accuracy 93.4.

Streamlit provides a simple and intuitive platform to build interactive web applications that can visualize and analyze data in real-time. With Streamlit, it is easy to develop customized dashboards and user interfaces that can help experts and stakeholders to make informed decisions about heritage preservation.

The prediction of architectural heritage elements using Streamlit can also help to optimize the use of resources and prioritize conservation efforts. By identifying the most critical elements and areas of risk, it is possible to allocate resources more effectively and reduce the likelihood of irreversible damage.

Overall, the use of Streamlit for predicting architectural heritage elements is a promising area of research that can lead to significant improvements in heritage conservation and restoration. As more data becomes available and machine learning techniques continue to evolve, we can expect to see more sophisticated and accurate predictions that will enable us to better protect our cultural heritage for future generations.

Chapter 8

References

1. Letellier, R.; Schmid, W.; LeBlanc, F. Recording, Documentation, and Information Management for the Conservation of Heritage Places: Guiding Principles; Routledge: London, UK; New York, NY, USA, 2007.

2. Remondino, F. Heritage Recording and 3D Modeling with Photogrammetry. Remote Sens. 2011, 3, 1104– 1138.

3. CIPA Heritage Documentation. Available online: http://cipa.icomos.org/ (accessed on 25 September 2017).

4. ICOMOS, International Council on Monuments Sites. Available online: http://www.icomos.org/ (accessed on 25 September 2017).

5. ISPRS, International Society of Photogrammetry and Remote Sensing. Available online: http://www.isprs.org/ (accessed on 25 September 2017).

6. Beck, L. Digital Documentation in the Conservation of Cultural Heritage: Finding the Practical in best Practice. Int. Arch. Photogramm. Remote Sens. Spat. Inf.

Sci. 2013, XL-5/W2, 85–90.

7. Hassani, F.; Moser, M.; Rampold, R.; Wu, C. Documentation of cultural heritage; techniques, potentials, and constraints. Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci. 2015, XL-5/W7, 207–214.

8. López, F.J.; Lerones, P.M.; Llamas, J.; Gómez-García-Bermejo, J.; Zalama, E. A framework for using point cloud data of heritage buildings towards geometry modeling in a BIM context: A case study on Santa Maria la Real de Mave Church. Int. J. Archit. Heritage 2017, 11, doi:10.1080/15583058.2017.1325541.

9. Apollonio, F.I.; Giovannini, E.C. A paradata documentation methodology for the Uncertainty Visualization in digital reconstruction of CH artifacts. SCIRES-IT 2015, 5, 1–24.

10. Di Giulio, R.; Maietti, F.; Piaia, E.; Medici, M.; Ferrari, F.; Turillazzi, B. Integrated Data Capturing Requirements for 3d Semantic Modelling of Cultural Heritage: The INCEPTION Protocol. Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci. 2017, XLII-2/W3, 251–257.

11. Oses, N.; Dornaika, F.; Moujahid, A. Image-based delineation and classification of built heritage masonry. Remote Sens. 2014, 6, 1863–1889.

12. Girshick, R.; Donahue, J.; Darrell, T.; Malik, J. Region-Based Convolutional Networks for Accurate Object Detection and Segmentation. IEEE Trans. Pattern Anal. Mach. Intell. 2016, 38, 142–158.

13. Long, J.; Shelhamer, E.; Darrell, T. Fully Convolutional Networks for Semantic Segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015.

14. Cireşan, D.; Meier, U.; Schmidhuber, J. Multi-column deep neural networks for image classification. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, 16–21 June 2012. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. arXiv 2015, arXiv:1512.03385.

15. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. In Proceedings of the 25th International Conference on Neural Information Processing Systems, Lake Tahoe, NV, USA, 3–6 December 2012; pp. 1097–1105.

16. Simonyan, K.; Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv 2014, arXiv:1409.1556.

17. Szegedy, C.; Toshev, A.; Erhan, D. Deep neural networks for object detection. In Proceedings of the 26th International Conference on Neural Information Processing Systems, Lake Tahoe, NV, USA, 5–10 December 2013; pp. 2553–2561.

18. Mnih, V.; Hinton, G. Learning to Label Aerial Images from Noisy Data. In Proceedings of the 29th International Conference on Machine Learning (ICML-12), Edinburgh, UK, 27 June–3 July 2012; pp. 567– 574. 20. Gao, F.; Huang, T.;

Wang, J.; Sun, J.; Hussain, A.; Yang, E. Dual-Branch Deep Convolution Neural Network for Polarimetric SAR Image Classification. Appl. Sci. 2017, 7, 447, doi:10.3390/app7050447.

19. . Tajbakhsh, N.; Shin, J.; Gurudu, S.; Hurst, R.; Kendall, C.; Gotway, M.; Liang, J. Convolutional Neural Networks for Medical Image Analysis: Full Training or Fine Tuning? IEEE Trans. Med. Imaging 2016, 35, 1299–1312.

20. Gao, Y.; Lee, H.J. Local Tiled Deep Networks for Recognition of Vehicle Make and Model. Sensors 2016, 16, 226, doi:10.3390/s16020226.

21. Li, C.; Min, X.; Sun, S.; Lin, W.; Tang, Z. DeepGait: A Learning Deep Convolutional Representation for View-Invariant Gait Recognition Using Joint Bayesian. Appl. Sci. 2017, 7, 210, doi:10.3390/app7030210.

22. Pedraza, A.; Bueno, G.; Deniz, O.; Cristóbal, G.; Blanco, S.; Borrego-Ramos, M. Automated Diatom Classification (Part B): A Deep Learning Approach. Appl. Sci. 2017, 7, 460, doi:10.3390/app7050460.

23. Liu, L.; Wang, H.; Wu, C. A machine learning method for the large-scale evaluation of urban visual environment. arXiv 2016, arXiv:1608.03396.

24. Sa, I.; Ge, Z.; Dayoub, F.; Upcroft, B.; Perez, T.; McCool, C. DeepFruits: A Fruit Detection System Using Deep Neural Networks. Sensors 2016, 16, 1222, doi:10.3390/s16081222.

25. Chu, W.-T.; Tsai, M.-H. Visual pattern discovery for architecture image classification and product image search. In Proceedings of the 2nd ACM International Conference on Multimedia Retrieval, Hong Kong, China, 5–8 June 2012.

26. Goel, A.; Juneja, M.; Jawahar, C.V. Are buildings only instances?: Exploration in architectural style categories. In Proceedings of the Eighth Indian Conference on Computer Vision, Graphics and Image Processing, Mumbai, India, 16–19 December 2012.

27. Mathias, M.; Martinovic, A.; Weissenberg, J.; Haegler, S.; Van Gool, L. Automatic Architectural Style Recognition. In Proceedings of the 4th ISPRS International Workshop 3D-ARCH 2011, Trento, Italy, 2–4 March 2011; Volume XXXVIII-5/W16, pp. 171–176. 30. Shalunts, G.; Haxhimusa, Y.; Sablatni, R. Architectural Style Classification of Building Facade Windows. In Advances in Visual Computing 6939; Springer: Las Vegas, NV, USA, 2011; pp. 280–289.

28. Zhang, L.; Song, M.; Liu, X.; Sun, L.; Chen, C.; Bu, J. Recognizing architecture styles by hierarchical sparse coding of blocklets. Inf. Sci. 2014, 254, 141–154.

29. Xu, Z.; Tao, D.; Zhang, Y.; Wu, J.; Tsoi, A.C. Architectural Style Classification Using Multinomial Latent Logistic Regression. In Computer Vision—ECCV 2014; Springer: Cham, Switzerland, 2014; Volume 8689, pp. 600–615.

30. Llamas, J.; Lerones, P.; Zalama, E.; Gómez García -Bermejo, J. Applying Deep Learning Techniques to Cultural Heritage Images within the INCEPTION Project.

In EuroMed 2016: Digital Heritage. Progress in Cultural Heritage: Documentation, Preservation, and Protection. Part I, Nicosia, Cyprus, 31 October–5 November 2016; Springer: Cham, Switzerland, 2016; Volume 10059, pp. 25–32.

31. Lu, Y. Food Image Recognition by Using Convolutional Neural Networks (CNNs). arXiv 2016, arXiv:1612.00983.

32. Yanai, K.; Kawano, Y. Food image recognition using deep convolutional network with pre-training and fine-tuning. In Proceedings of the IEEE International Conference on Multimedia Expo Workshops (ICMEW), Turin, Italy, 29 June–3 July 2015; pp. 1–6. 36. Datahub. Available online: https://datahub.io (accessed on 25 September 2017).