BAŞKENT UNIVERSITY
INSTITUTE OF SCIENCE AND ENGINEERING
DEPARTMENT OF COMPUTER ENGINEERING
MASTER OF SCIENCE IN COMPUTER ENGINEERING

RECOGNIZING VISUAL PLACES FROM LANDSCAPES WITH ZERO SHOT LEARNING

BY

ERDEM SAVAŞCI

MASTER OF SCIENCE THESIS

ANKARA – 2021

BAŞKENT UNIVERSITY
INSTITUTE OF SCIENCE AND ENGINEERING
DEPARTMENT OF COMPUTER ENGINEERING
MASTER OF SCIENCE IN COMPUTER ENGINEERING

RECOGNIZING VISUAL PLACES FROM LANDSCAPES WITH ZERO SHOT LEARNING

BY

ERDEM SAVAŞCI

MASTER OF SCIENCE THESIS

ADVISOR

DR. EMRE SÜMER

ANKARA – 2021

BAŞKENT UNIVERSITY

INSTITUTE OF SCIENCE AND ENGINEERING

This study, which was prepared by ERDEM SAVAŞCI, for the program of MASTER IN COMPUTER ENGINEERING WITH THESIS, has been approved in partial fulfillment of the requirements for the degree of MASTER OF SCIENCE in COMPUTER ENGINEERING Department by the following committee.

Date of Thesis Defense: 20 / 08 / 2021

Thesis Title: RECOGNIZING VISUAL PLACES FROM LANDMARKS WITH ZERO-SHOT LEARNING

Examining Committee Members	**Signature**
Asst. Prof. Dr. Emre Sümer (Supervisor)	……
Assoc. Prof. Dr. Mehmet Serdar Güzel	……
Asst. Prof. Dr. Tunç Aşuroğlu	……

APPROVAL

…/…/2021

Prof. Dr. Ömer Faruk ELALDI

Director, Institute of Science and Engineering

Date: …/…/2021

BAŞKENT ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

YÜKSEK LİSANS TEZ ÇALIŞMASI ORİJİNALLİK RAPORU

Tarih: …/…/2021

Öğrencinin Adı, Soyadı: ERDEM SAVAŞCI

Öğrencinin Numarası: 21820281

Anabilim Dalı: BİLGİSAYAR MÜHENDİSLİĞİ

Programı: BİLGİSAYAR MÜHENDİSLİĞİ TEZLİ YÜKSEK LİSANS

Danışmanın Unvanı/Adı, Soyadı: DR. ÖĞR. ÜYESİ EMRE SÜMER

Tez Başlığı: Örneksiz Öğrenme Yöntemi ile Görsel Yerler Üzerinden Tanıma Gerçekleştirme

Yukarıda başlığı belirtilen Yüksek Lisans tez çalışmamın; Giriş, Ana Bölümler ve Sonuç Bölümünden oluşan, toplam 69 sayfalık kısmına ilişkin, 27 / 08 / 2021 tarihinde tez danışmanım tarafından Turnitin adlı intihal tespit programından aşağıda belirtilen filtrelemeler uygulanarak alınmış olan orijinallik raporuna göre, tezimin benzerlik oranı % 6'dır. Uygulanan filtrelemeler:

1. Kaynakça hariç

2. Alıntılar hariç

3. Beş (5) kelimeden daha az örtüşme içeren metin kısımları hariç

"Başkent Üniversitesi Enstitüleri Tez Çalışması Orijinallik Raporu Alınması ve Kullanılması Usul ve Esaslarını" inceledim ve bu uygulama esaslarında belirtilen azami benzerlik oranlarına tez çalışmamın herhangi bir intihal içermediğini; aksinin tespit edileceği muhtemel durumda doğabilecek her türlü hukuki sorumluluğu kabul ettiğimi ve yukarıda vermiş olduğum bilgilerin doğru olduğunu beyan ederim.

Öğrenci İmzası: …….

ONAY

Tarih: …/…/2021

Dr. Öğr. Üyesi Emre Sümer

……

ACKNOWLEDGEMENTS

I would first like to thank my Master of Science Thesis Advisor Asst. Prof. Dr. Emre Sümer for his valuable comments and guidance throughout the thesis. I will always be thankful.

I would also like to thank all my instructors at Başkent University for their priceless efforts to pass their own knowledge on to us persistently.

Last but not least, I would like to thank my family for their infinite moral support and help throughout my life.

Furthermore, I would also like to thank my thesis committee and all the staff of Başkent University.

This accomplishment would not have been possible without any of these people.

ABSTRACT

Erdem SAVAŞCI

RECOGNIZING VISUAL PLACES FROM LANDSCAPES WITH ZERO-SHOT LEARNING

Başkent University Institute of Science and Engineering

Department of Computer Engineering

2021

Image processing and deep learning methods are being developed day by day and the need of recognizing objects and extracting information from visual-based digital multimedia data like pictures and videos is increasing. Therefore, the algorithms are also changing rapidly to meet the requirements of the tasks in everyday life. Zero-shot learning is a new topic and it encourages having small datasets. It still accomplishes the recognition task efficiently. As the name implies, Zero-shot learning is the process of making predictions for the unseen or untrained categories of data based on some amount of data that is learned by the training process earlier. In this study, visual recognition which is based on detecting the city, country, and continent is investigated. There is no or enough similar work on this problem up to now. The unseen places are recognized by training on similar places by categorizing the cities, countries, and continents. Labels and some auxiliary features like a 3D color histogram and GIST features are used to increase the detection accuracy. A new dataset is created for this study and with this work, the importance of the amount of data in the dataset is discussed and various metrics like performance and duration are demonstrated. Also, the comparison with the regular training problem setup is discussed. After all the results are examined, it is seen that ZSL performs better when search space is constrained with only unseen classes at test time. Also, the original generalized ZSL method performs better than the other generalized ZSL method and also inductive ZSL. Compared to traditional learning, various ZSL methods can give sufficient results too.

KEYWORDS: Geolocation, Image processing, Deep learning, ZSL, Visual place recognition

Advisor: Asst. Prof. Dr. Emre Sümer, Başkent University, Computer Engineering Department.

ÖZ

Erdem SAVAŞCI

ÖRNEKSİZ ÖĞRENME YÖNTEMİ İLE GÖRSEL YERLER ÜZERİNDEN TANIMA GERÇEKLEŞTİRME

Başkent Üniversitesi Fen Bilimleri Enstitüsü

Bilgisayar Mühendisliği Anabilim Dalı

2021

Resimler üzerinden tanıma gerçekleştirme halen üzerinde sıkça çalışılan bir konu olmakla beraber gün geçtikçe ihtiyacın da arttığı bir alandır. Örneksiz Öğrenme yöntemi ile daha az veri üzerinde çalışılmakta olduğu için ve gerçek hayat senaryolarına yakın olması itibariyle önceden görülmemiş ve lokasyonu bilinmeyen görsel yerler üzerinden tanıma işleminde kullanılmasının yararlı olacağı düşünülmüştür. Literatürde Örneksiz Öğrenme yöntemi yaklaşımı ile şehir, ülke ve kıta bazında bir tanıma işleminin gerçekleştirilmesi konusunda örnek olabilecek bir çalışmaya rastlanmamıştır. Tanıma işleminin güçlendirilmesi için uzaklık metotları, üç boyutlu histogram ve GIST (resim içerisindeki karakteristik özelliklerin kodlanmış hali) veri seti üzerinden çıkartılarak yer tahmini işleminde kullanılmıştır. Bu çalışmada farklı şehirlerin yer aldığı yeni bir veri seti oluşturulmuş olup, görülmemiş bir yer resmi üzerinden tahminin ancak önceden benzer yerler üzerinden gerçekleşen öğrenme işlemi sayesinde yapılabileceği için çalışmada alınmış olan bazı kararlar da anlatılmıştır. Çalışma kapsamında normal öğrenme tekniği ile kıyaslama yapılıp Örneksiz Öğrenme yönteminin avantajları anlatılmıştır. Tüm sonuçlar incelendikten sonra standart Örneksiz Öğrenme metodunun genelleştirilmiş ve test zamanında sadece test kümesi üzerinde çalışan versiyonundan daha iyi çalıştığı görülmüştür. Ek olarak genelleştirilmiş ve test zamanında eğitim ve test kümesi üzerinde çalışan versiyonun en iyi sonuçlar verdiği ve normal öğrenme tekniği ile kıyaslanabilir olduğu görülmüştür.

ANAHTAR SÖZCÜKLER: Coğrafi konum, Görüntü işleme, Derin öğrenme, Örneksiz öğrenme, Görsel yer tanıma

Danışman: Dr. Öğr. Üy. Emre Sümer, Başkent Üniversitesi, Bilgisayar Mühendisliği Bölümü.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS ..... **i**

ABSTRACT ..... **ii**

ÖZ ..... **iii**

LIST OF FIGURES ..... **vi**

LIST OF TABLES ..... **viii**

LIST OF SYMBOLS AND ABBREVIATIONS ..... **ix**

1. INTRODUCTION **1**

1.1. Problem Statement and Motivation **1**

1.2. Image Classification Using Convolutional Neural Networks **3**

1.3. Introduction to Zero-Shot Learning **7**

1.4. Zero-Shot Learning vs Traditional Methods **10**

1.5. Auxiliary Features and Inference Methods **13**

1.6. Contribution of the Thesis **17**

1.7. Thesis Outline **20**

2. LITERATURE REVIEW **21**

2.1. Similar Works on Recognizing Cities or Landmarks **21**

2.2. Problems in Similar Works **27**

2.3. Discussion and Comparison **29**

3. MATERIALS AND METHODS **35**

3.1. Gathering of Images and Creation of Dataset **35**

3.2. Dataset Splits **37**

3.3. City, Country, Continent Classes and Grouping 39

3.4. Some Considerations 41

3.5. Used Convolutional Neural Network Architectures 42

3.6. Creating Features for Recognition 45

3.7. Training Methods 48

3.8. Validation Methods 51

3.8.1. Train-Validation-Test Split 51

3.8.2. Leave-one-group-out Cross-Validation Split 52

3.9. Testing Methods 54

3.10. Final Enhancements 56

4. RESULTS AND DISCUSSION 57

4.1. Results of Zero-Shot Learning Approach 57

4.2. Comparison of Different Problem Setups and Approaches 64

4.3. Limits and Shortcomings of the Thesis 66

4.4. Discussion and Possible Improvements for Future 67

5. CONCLUSION 68

REFERENCES 70

LIST OF FIGURES

Page

Figure 1.1. An overview of a CNN architecture as a block diagram..... 4

Figure 1.2. Overview of VGG16 architecture 6

Figure 1.3. Phases of Zero-shot learning 8

Figure 1.4. Zero-shot learning approach and relation of true connection and bias 10

Figure 1.5. Summarized view of different methods 11

Figure 1.6. Applying Hamming distance on two attribute or label vectors.....14

Figure 1.7. Illustration of global and local receptive fields.....17

Figure 2.1. Partitioning earth using S2 cells.....24

Figure 2.2. t-SNE visualization of features 25

Figure 2.3. Different viewpoints of the same place and firing of the same convolution filters.....25

Figure 2.4. Different stages with corresponded attributes.....26

Figure 2.5. Schematic illustration of ZSL and GZSL.....26

Figure 2.6. Visualization of occlusion heatmaps.....28

Figure 2.7. Visualization of inclusion heatmap 28

Figure 2.8. Sample images from the original World Cities dataset.....30

Figure 2.9. Some street-view images from the created street-level dataset.....32

Figure 2.10. Example of similarities of different cities 32

Figure 2.11. Example of similarities of different cities 33

Figure 2.12. Example of similarities of different cities 33

Figure 2.13. Separate RGB color components 33

Figure 2.14. Separate CIELAB color components 34

Figure 2.15. Separate CIELAB color components after adjusting contrast 34

Figure 3.1. Illustration of images in the created dataset 36

Figure 3.2. Cities included in the created dataset 40

Figure 3.3. Countries included in the created dataset 40

Figure 3.4. Continents included in the created dataset 41

Figure 3.5. Main layers of the model architecture used for the ZSL configuration 42

Figure 3.6. Main layers of the model architecture used for traditional training configuration 43

Figure 3.7. Grad-CAM results for an image belonging to Hong Kong as a seen class 44

Figure 3.8. Grad-CAM results for an image belonging to Venice as an unseen class 44

Figure 3.9. Word cloud of the labels used to represent cities 45

Figure 3.10. 3D color histogram visualizations of a Havana city image 46

Figure 3.11. Visualizations of GIST descriptors 47

Figure 3.12. Pseudocode of traditional training process 48

Figure 3.13. Pseudocode of ZSL training process 49

Figure 3.14. Pseudocode of TZSL training process 50

Figure 3.15. Summary of the experimented different techniques throughout the study 51

Figure 3.16. Pseudocode of validation process using train-validation-test split 52

Figure 3.17. Pseudocode of validation process using cross-validation split 53

Figure 3.18. Pseudocode of traditional testing algorithm 54

Figure 3.19. Pseudocode of ZSL testing algorithm 55

LIST OF TABLES

Page

Table 1.1. Comparison of different classification techniques....12

Table 3.1. Total city images in each split 38

Table 4.1. Zero-shot learning results of city classification having only test classes at test time 57

Table 4.2. Zero-shot learning results of city classification having both seen and unseen classes at test time....58

Table 4.3. Zero-shot learning results of country classification having only unseen classes at test time 59

Table 4.4. Zero-shot learning results of country classification having both seen and unseen classes at test time....59

Table 4.5. Zero-shot learning results of continent classification having only unseen classes at test time 60

Table 4.6. Zero-shot learning results of country classification having both seen and unseen classes at test time....60

Table 4.7. Zero-shot learning results of city classification with or without enabling auxiliary features at test time 61

Table 4.8. Zero-shot learning results of city classification having only European classes....62

Table 4.9. Zero-shot learning results of city classification having all 42 classes 62

Table 4.10. Transductive Zero-shot learning results of city classification having 6 classes 63

Table 4.11. Traditional training results of city classification having only European classes....64

Table 4.12. Traditional training results of city classification having all the classes 65

Table 4.13. Traditional training results of city classification having different sizes of classes 65

LIST OF SYMBOLS AND ABBREVIATIONS

ANN	Artificial Neural Network
BCE	Binary Cross-Entropy
CNN	Convolutional Neural Network
FC	Fully-Connected
GIST	Global Image Descriptor
GTZSL	Generalized Transductive Zero-Shot Learning
GTZSL-1	Transductive Zero-shot learning using the alternative generalized approach
GTZSL-2	Transductive Zero-shot learning using the original generalized approach
GZSL	Generalized Zero-Shot Learning
GZSL-1	Alternative approach to the original generalized Zero-shot learning
GZSL-2	Original generalized Zero-shot learning
Grad-CAM	Gradient-weighted Class Activation Mapping
LSTM	Long-Short Term Memory
NLL	Negative Log-Likelihood
RNN	Recurrent Neural Network
SIFT	Scale-Invariant Feature Transform
T-SNE	T-Distributed Stochastic Neighbor Embedding
TZSL	Transductive Zero-Shot Learning
ZSL	Zero-Shot Learning

1. INTRODUCTION

1.1. Problem Statement and Motivation

Image processing is a very important topic to work on and the need of detecting and recognizing objects in digital multimedia formats like pictures or videos is increasing day by day. Applicable fields are so many that nearly every technological device uses image processing nowadays. With the advance of the hardware, the processing capabilities are increased. Detection, recognition, and identification can be done by processing images. The recognition process of an object is always hard because we need many samples of the object that we want to recognize. Therefore, datasets play a very important role in this topic. With the increasing works on machine learning and deep learning, the image processing field gained many capabilities in terms of recognizing nearly anything at any time.

Recognizing visual places is also a challenging task because of the complexity of the images. This complexity can lead to bad results since extracting good and generic information from the dataset having many objects in each image can be hard. Good generalization is needed to ensure that the trained model can give accurate and good results. The good thing about recognizing visual places is its nature of being applicable to various fields such as GPS and map systems, guessing games, or even social media applications. Training and predicting visual places can solve many problems in human lives. An unknown visual place can be recognized by only capturing a single photo and making predictions on this single image. There is a video game based on location discovery called GeoGuessr [1] and this study is similar to it in terms of discovering location-based on features but the goal is predicting a location without human interaction.

Predicting a visual place is possible by capturing some significant features at training time. These significant features may be architectural for the city, country, or continent recognition. This learning visual places process is very similar to the human brain. The human brain can remember visual places by encoding the significant features and because Artificial Neural Networks principles are very similar to the human brain, it is possible to teach machines to learn visual places by deep learning methods. This work is also very similar to real-life scenarios and is possible by extracting useful features and using them in the prediction part. The extracted features contain significant labels of dataset images. For example, GIST features [2], [3] contain

vectorized information and can represent the macro meaning of an image. Likewise, 3D color histogram features [4] can help to differentiate two different pictures with some significant colors such as comparing an ocean image with a forest image. Of course, the results can sometimes be obvious and sometimes not. With the help of distance methods, these extracted features serve their purpose as auxiliary features to increase the detection rate.

It is possible to categorize visual places like cities, countries, and continents. Similar studies show that this task can be possible by creating a dataset regarding GPS and location tags for the city images by fetching images from the web or manually capturing the images of the cities. Usually, a custom dataset is created for this task because visual places often vary based on the purpose. The city recognition task usually needs many samples on different significant landmarks from each image. For this purpose, huge datasets are usually created having many samples for each image. The gathering process of images is hard to achieve because no matter how much there exists samples, they will not be enough to cover the whole city. So, regarding this problem, the model that can make good generalizations can be used to make predictions on the images. Also, with the growing number of images on the Internet, this task can also be easily implemented by using Online Learning (learning as data comes in) methods. For example, a similar study [5] has constructed a 6 million GPS-tagged images dataset to estimate the geographic information from images. Another related study [6] has constructed a novel CNN model which is based on Inception architecture by collecting millions of images from the web and then testing by creating a test set consists of millions of images from Flickr to measure the localization accuracy. These studies are important for this task and even they cannot recognize the whole world, recognizing significant or the part of the world is enough, for now, to give advice about an unknown place. Also, these studies are not conducted on training seen places at training time while predicting unseen places based on learned features from seen places at test time which is done in this thesis.

In these days there are increasing studies on Zero-shot learning, One-Shot Learning, and Few-Shot Learning. These problem setups achieve a very important task which is reducing the data need. The data need for recognition tasks which are recognizing visual places is very high as in this study. Since visual places can contain similar features in them, these problem setups can be applied. Zero-shot learning is an effective solution for city, country, and continent recognition because the dataset needed can be really small, and still visual place recognition can

be achieved. Visual place recognition with Zero-shot learning technique mainly contains the steps which are the creation of labels set, extracting GIST and 3D color histogram features, training on seen classes, validation on unseen classes and finally testing on unseen classes. Testing on unseen classes is done by comparing labels created for unseen data with seen classes' data using Hamming Distance to calculate the distance between two label vectors and also using the extra features which are GIST and 3D color histogram features as auxiliary features to increase the recognition rate. Cities can be recognized by this technique and country and continent recognition is also done by mapping recognized cities into their countries and comparing them. In this way, when the country or continent is the same for the compared cities, then the prediction of the queried city can be counted as correct. Zero-shot learning can achieve enough results compared to regular training methods if datasets and labels are created in the right way. Labels can help the comparing process as they contain summarized information from images. For example, when comparing two visual places, one place can have ocean, boat, water labels and the other place can have forest, tree, and sky labels. These labels are encoded in a way that model outputs and learning from the images as comparing label distance and the other auxiliary features to recognize city images.

1.2. Image Classification Using Convolutional Neural Networks

Convolutional Neural Network (CNN) architectures [7], [8] are frequently used in these days for many various visual tasks. Gu et al. [9, pp. 1] define this architecture as below:

"Convolutional Neural Network (CNN) is a well-known deep learning architecture inspired by the natural visual perception mechanism of the living creatures."

CNN architecture is similar to traditional ANN architecture in that they both aim to achieve learning by processing inputs, calculating loss, and learning from loss. CNN is specialized in working with image data which the architecture can extract features and learn from them. Working on raw pixel data from images enables capturing all the details from images and making a generalization based on the features that the architecture found important. CNNs are suitable for image processing and recognition tasks because of their nature and there are many popular and successful models based on this architecture like ResNet, VGG, Inception, etc. These networks include many layers and each layer is responsible for its own task as

processing the whole image by partitioning it into separate parts. The further definition is given by Kim et al. [10, pp. 148] as below:

"CNN structure, which is inspired by the brain's primary visual cortex, is the most popular deep neural network model for image recognition. The operation itself is similar to conventional image processing but the significant difference is that the kernels are learned by a supervised learning process, rather than specified by human experts."

Provided image to the CNN architecture consists of width, height, and depth. When working on color images, the depth will be usually 3 or sometimes 4 for images that have an alpha channel. Every CNN network has some predefined image size to be able to process them correctly. Therefore, some pre-processing operations are done on the images before the images are fed into the CNN networks.

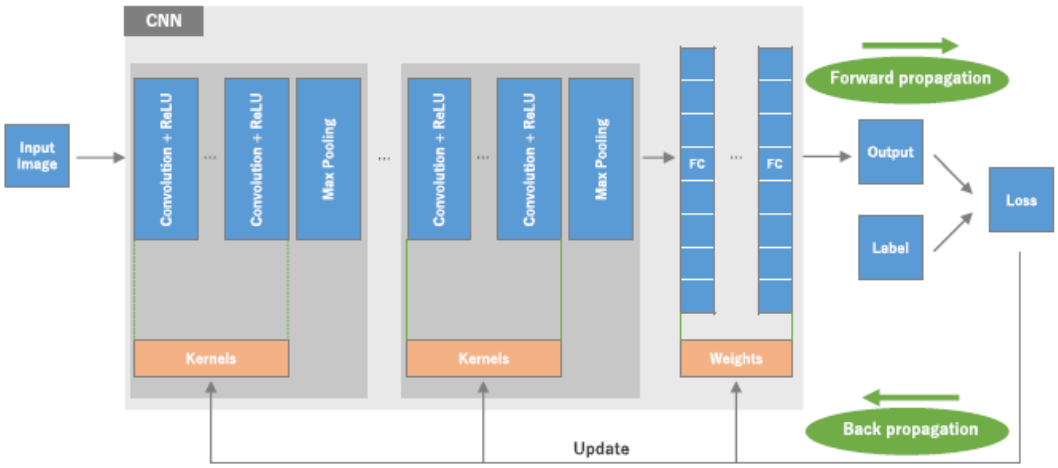

Figure 1.1. An overview of a CNN architecture as a block diagram [11]

The summarized information about internal processes of a CNN architecture and relations between them are given in Figure 1.1. A typical CNN architecture consists of three types of layers. These are convolution layers, pooling layers, and fully-connected layers. The input image is processed within these layers and features are learned by repeating processes many times to be able to make predictions. Every time loss is calculated and backpropagated for correct learning.

There are various variations of models based on CNN architectures. Some examples are InceptionV3 [12], ResNet50 [13] and VGG16 [14]. These architectures are frequently used and shown as successful, especially in the image processing and recognition area. All these 3 models

have similar architecture, the key differences are their layer counts, layer placements, convolutional filter sizes, and computations. These 3 models are available as pre-trained on ImageNet dataset [15] for various use cases. Even the context of the task is different from ImageNet and its objects, still, these models can be used for the transfer learning process [16]. The transfer learning process has two types which are finetuning and feature extraction. Finetuning consists of starting with a pre-trained model and updating all of the parameters of the model for the new task. On the other hand, feature extraction consists of starting with a pre-trained model and updating only the final layer weights for the new task. In this way, the remaining parameters remain the same and the previously learned weights are not gone completely. In this work, feature extraction is applied as a transfer learning type.

$$\Phi\,(X, W1, ..., W^K) = \psi_K\,(\psi_{K-1}\,(\cdots \psi_2\,(\psi_1\,(X\,W^1)\,W^2)\,\cdots W^{K-1})\,W^K). \tag{1.1}$$

The above equation from [17] summarizes the deep learning with convolutional networks as a formula that consists of an output matrix as Φ, an input data (or in this context, image) as X, a matrix after linear transformation, convolution with some filter applied as W, a non-linear activation function as ψ and the layer number as K. At the end of the architecture, Sigmoid or Softmax activation functions can be applied to map the range of the output vector. In this work, Sigmoid activation function is used for the label-based classification which is done in Zero-shot learning problem setup and Softmax activation function is used for the regular, conventional training problem setup in which there is no unseen class in it.

$$f(x_i) = \left(\frac{1}{1 + exp^{-1}}\right) \tag{1.2}$$

The Sigmoid activation function equation [18] is given above. Its input range is between -∞ and +∞ and the output range is between 0 and 1. It simply limits infinite real values to the boundary of 0 and 1. It is useful for multi-label classification and the sum of the probabilities is not equal to zero. Because of these characteristics, it has been selected to be used on label-based (i.e. attribute-based) classification in Zero-shot learning setup.

$$f(x_i) = \left(\frac{\exp\ (x_i)}{\sum_j \exp\ (x_j)} \right) \tag{1.3}$$

On the other hand, the Softmax activation function equation [18] is given above. As its name implies, it is related to the maximum, higher probabilities as it outputs probability values as the sum of them equal to 1. The range of its output is between 0 and 1. It is suitable for multi-class classification and in this work, it is used in regular, conventional training problem setup. Even the two activation functions map the real values to the 0-1 range, they are used in different tasks.

By these two activation functions, the probability vectors from the last layer of models are calculated and the loss is calculated for Sigmoid and Softmax activation functions with BCE (Binary Cross-Entropy) and NLL (Negative Log-Likelihood) loss functions respectively. Also, there is RELU activation function which is mostly used in CNN architectures and is usually placed between inside hidden layers of the models. An overview of these layers and a general view of VGG16 model are given in Figure 1.2. This general view summarizes all the tasks done in CNN model for image classification as partitioning and downscaling of the input image as feature maps. The model in Figure 1.2 uses Softmax at its final layer. In this work, experiments are mainly done on VGG16 architecture.

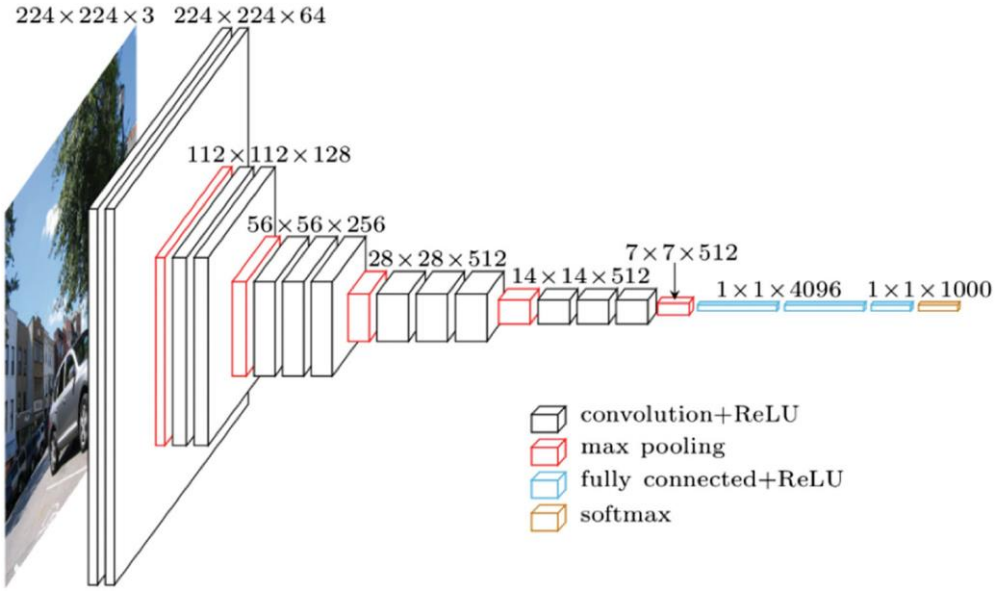

Figure 1.2. Overview of VGG16 architecture [19]

Also, hyperparameters of these CNN models while doing training should be considered and there are different parameters that affect the training process and results such as learning rate, weight decay, batch size, momentum, etc. These parameters are configured throughout the training process and different configurations lead to different results. In addition to these, epoch number is also an important parameter to consider the stopping criteria when the target accuracy value is reached. Some configurations also help to reduce train time.

Before feeding images into the model as inputs, some pre-processing operations should be done on images like cropping, resizing, normalizing, etc. VGG16 and ResNet50 expect image sizes as 224x244 while InceptionV3 expects images as 299x299. Usually, input images are in 3 channels format having no transparent layer (4th channel) in them.

There are various studies that are using CNN architectures in place recognition tasks like [20]. Also, the transfer learning approach is used in these studies to transfer knowledge from one context to another.

1.3. Introduction to Zero-Shot Learning

As the name implies, Zero-shot learning [21] consists of two disjoint sets which are seen and unseen class sets. The important part is such that these seen and unseen groups are completely different classes, not different data. As the regular, conventional training process also consists of seen and unseen data, all data are all actually in the seen classes set. Zero-shot learning aims to make predictions on the unseen classes based on the seen classes by learning features from them. Thinking of relation and similarity between seen and unseen classes, this type of learning is possible by feeding seen classes' data into the model as input and with the help of some auxiliary, meta-data like label matrix which are also known as semantic information [22], predictions can be done on unseen classes. The main advantage of Zero-shot learning is the less need for huge datasets because there exists only seen classes in the training part and with fewer data, predictions can still be done on the target data which are images in this work. In contrast to One-shot learning and Few-shot learning, Zero-shot learning does not include any sample of target classes.

A good definition of Zero-shot learning by Larochelle et al. [21, pp. 1] is given below:

"Zero-data learning is useful for problems where the set of classes to distinguish or tasks to solve is very large and is not entirely covered by the training data."

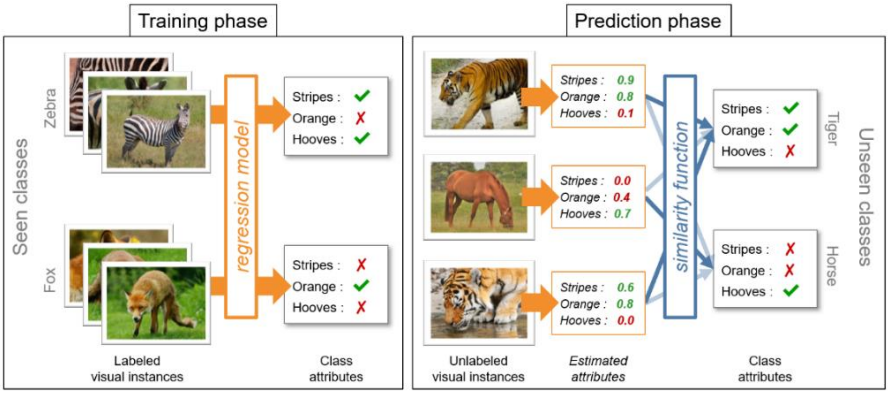

Figure 1.3. Phases of Zero-shot learning [23]

Like Figure 1.3, one way to achieve Zero-shot learning requires semantic descriptions or labels as information to recognize unseen classes by comparing labels using similarity functions. These similarity functions can be distance methods which are for example Euclidean distance, Hamming distance, Cosine distance, Manhattan distance, Minkowski distance, Bhattacharyya distance, etc. These distance formulas can be used to calculate the similarity between two label vectors. In this work, Euclidean, Hamming, and Bhattacharyya distance methods are used in various places like in the inference phase or creation of auxiliary information to calculate the similarity between label vector as output from model and target label vector.

Attribute-based (label-based) classification [24], [25] is typically done by creating a binary matrix that is based on encoding attributes or labels as 0 and 1s. These encoded values indicate that an image consists of these attributes or labels as it contains them as information in its pixels. Labels can vary from physical features to behaviors, the important thing to consider is their explanation level. Labels should be well-defined for images and images should have distinct features to make comparisons on them. In this way, information transfer by label sharing can be done for images.

There are two popular different setups to achieve different results for Zero-shot learning. The first one is typical Zero-shot learning as there are two disjoint sets which are training and test sets and predictions done on the test set are performed by regarding only test labels to recognize test set which has unseen images. The second setup is called Generalized Zero-shot learning (GZSL) [26], [27] which consists of disjoint train and test set but in this setup, predictions done on the test set are performed by regarding all labels which are for both training

and test set and both these sets are available at test time. In this way, a more generalized and realistic approach can be achieved. These two approaches which are ZSL and GZSL can be formulated as below [22], [28].

$$S = f(x_n, y_n), n = 1 \dots N, \quad x_n \in X^{tr}, \; y_n \in Y^{tr} \qquad \text{(Training)} \quad (1.4)$$

$$\mathrm{P} = f(x_n, y_n), n = 1 \dots N, \quad x_n \in X^{ts}, \; y_n \in Y^{ts} = \{ K + 1 \dots M \} \qquad \text{(ZSL)}$$

$$\mathrm{P} = f(x_n, y_n), n = 1 \dots N, \quad x_n \in X^{tr+ts}, \; y_n \in Y^{tr+ts} = \{ 1 \dots M \} \qquad \text{(GZSL)}$$

$$f : X \to Y \qquad \text{(Task)}$$

Let X be the image set, Y be the label set, S be the training set, P be the test set, K be the unseen labels' start index and M be the total seen and unseen labels count. The original task is to predict the label of the image set which is denoted as from X to Y.

Pourpanah et al. [29, pp. 2] have described GZSL with the following words:

"In conventional ZSL techniques, the test set only contains samples from the unseen classes, which is an unrealistic setting and it does not reflect the real-world recognition conditions. In practice, data samples of the seen classes are more common than those from the unseen ones, and it is important to recognize samples from both classes simultaneously rather than classifying only data samples of the unseen classes."

In GZSL, at test time there exist both training and test class labels to assign test images to them. Of course, both in ZSL and GZSL, at training time only training class labels are present. If any unseen class labels were provided at training time, this approach would violate Zero-shot learning. There are some issues that need to be considered when ZSL approach is applied. One of them is having disjoint two sets which are seen and unseen sets. Another issue is if a pre-trained model is being used, then this pre-trained should not be trained on classes that seen or unseen classes of the current ZSL task contain. Also, if a validation set is created, then this set can be created from unseen classes so it can show model success better. If these issues are considered while ZSL approach is being applied for a task, then the trained model can adapt properly to unseen classes.

Also, there is another type of ZSL which is Transductive ZSL that is designed to reduce bias at training time [30], [31]. Because conventional ZSL and generalized ZSL both map inputs to only training set labels at training time, it can create strong bias because visual embeddings are mapped to a fixed size of semantic embeddings. To alleviate this strong bias, TZSL configuration includes both labeled source data and unlabeled target data at the training time.

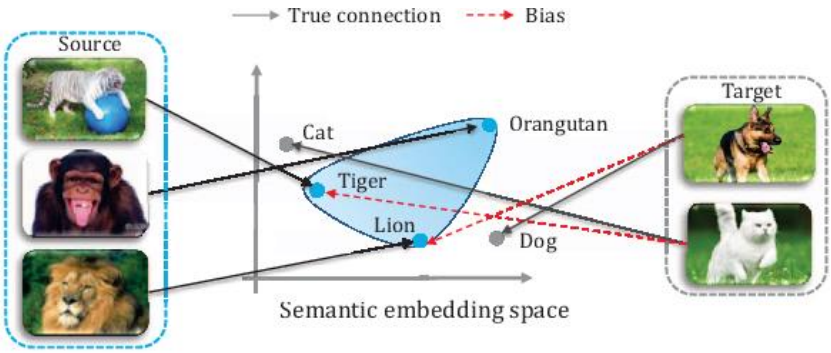

Figure 1.4. Zero-shot learning approach and relation of true connection and bias [30]

In this work, ZSL and GZSL problem setups are applied for city, country, and continent recognition. City classes are split into two or three different disjoint sets which are typically created by training-test split or training-validation-test split. The cross-validation method consists of two disjoint sets and regular, conventional training consists of three disjoint sets. Because of grouping cities into disjoint sets, cities that have similar characteristics or cities that are in the same country or the same continent are placed to different sets. In this way, training is done on seen city classes and predictions are made on unseen city classes. Country and continent accuracy values are calculated by mapping predicted cities into their own country or continent class names.

1.4. Zero-Shot Learning vs Traditional Methods

Until these days, traditional methods which basically work on seen classes show really good results but also the creation of datasets remains problematic. While the creation of datasets is hard, the need for other methods has arisen. Sometimes sample count for an object could not be sufficient for comprehensive tasks and because of that sometimes data augmentation is applied. Of course, data augmentation is usually applied into training classes and by this way, overfitting is tried to be prevented and the model is aimed to make generalizations better. For tasks which are dealing with unseen classes that are different from seen classes but are similar, regular methods may not be suitable thus some other methods are proposed to deal with these issues.

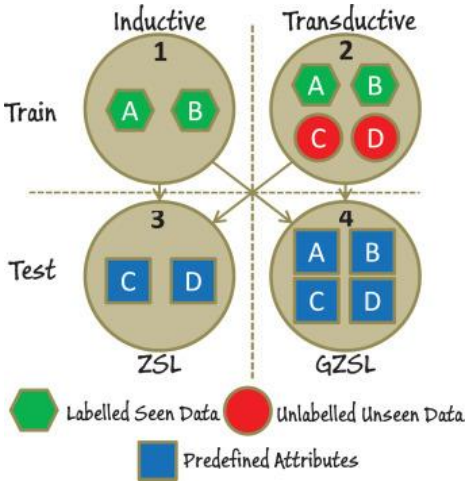

Figure 1.5. Summarized view of different methods [32]

Zero-shot learning is mostly needed when the training set is not enough for the task. This problem can be solved with Few-shot learning, One-shot learning, or Zero-shot learning setups. The main difference between these three setups is the need for samples (i.e. sample count) and ZSL can work on new and unseen classes which can be a subclass of or similar to trained data classes.

A picture from [32] as shown in Figure 1.5 summarizes these configurations. GZSL addresses issues that ZSL has, which is, for example, being restrictive because of having only unseen classes at test time. In GZSL, both seen and unseen classes are present at test time and a more generalized approach is aimed to be reached. Transferring knowledge from seen classes to unseen classes using well-defined semantic description is possible but also to increase the recognition rate, some auxiliary information can be included in the testing stage. A good example of this transferring knowledge subject is given by Pourpanah et al. [29, pp. 1] as below:

"For instance, a child can easily recognize zebra, if he/she has seen horses before and known that zebra looks like a horse with black and white strips. Zero-shot learning (ZSL) [8], [9] techniques offer a good solution to address such challenge."

ImageNet dataset can be used for both traditional and ZSL methods. Currently, this dataset includes more than 14 million images and there are various pre-trained models which are pre-trained on this dataset. Categories of objects this dataset has are important because pre-trained images affect training methods especially ZSL. To not violate ZSL, it is preferable to select classes that ImageNet does not include when a model pre-trained on ImageNet has been selected to be used.

In the traditional training and inference process, the final fully-connected layer outputs model's learned class scores which are then passed into activation functions to get top classes that have higher probability score. These activation functions vary from Softmax, Sigmoid to hyperbolic tangent to determine boundaries of scores and making estimations on them. On the other hand, in ZSL inference process, the final fully-connected layer again outputs the model's learned class scores but this time the outputs are mapped to similar, related classes which are usually unseen ones. In GZSL, both train and test classes are tried to be mapped to related classes with the help of semantic, label-based features.

In both of the methods, the original task is predicting cities by learning from visual features. CNN architecture is suitable for these types of tasks and in this work, both pre-trained (on ImageNet dataset) ResNet50, InceptionV3, and VGG16 models are tested to select a better working one. Each image is pre-processed before being passed as input to CNN models and some transformation operations like color jittering, random rotation, horizontal flip, etc. are applied onto images in both traditional and ZSL methods. The last fully-connected layer is modified according to output class size which is class count when the traditional method is selected and label size when ZSL configuration is selected. Inference time is straightforward for the traditional method. In ZSL, similarity functions (in this work, distance methods), auxiliary information consists of 3D color histogram, GIST features are provided to try to select correct city, country, and continent classes.

In this work, for the traditional training process, Softmax activation function is used for the output of the last layer of fully-connected layer of the model. As Softmax limits model output to probability values as 0 to 1; top-1, top-2, top-3, and top-5 results are calculated by regarding higher probability class scores from each output. For ZSL training process, Sigmoid activation function is used because label-based classification is done and the output label vector is mapped to the label vector of the target class having higher similarity.

Table 1.1. Comparison of different classification techniques [33]

	K = 2	K > 2
L = 1	Binary	Multi-Class
L > 1	Multi-Label	Multi-Output

Comparison between different classification techniques is shown in Table 1.1. Multi-output classification is also known as multi-target or multi-dimensional [33]. All of these techniques are frequently used in deep learning problems. Set of target variables (labels) are denoted as L and set of instances are denoted as K. ZSL technique in this work can be defined better with multi-output because of having multiple classes which are cities, countries, or continents and having multiple labels for each of class instance. Some examples of these labels are water, landmark, road, town, house etc. There are total of 134 labels and each is encoded as 0s and 1s regarding their existence in the majority of the city images.

In conclusion, traditional methods work with only seen classes which all are available in training and test set but of course data is split into sets as having only unseen data of seen classes at test time. On the other hand, ZSL and its variations deal with unseen classes and unseen data of them. This difference is crucial for some of the tasks which especially require huge samples. In addition to these GZSL and TZSL aimed to solve problems that ZSL have like bias or generalization. Also, traditional methods that are explained here are interchangeably used with conventional, regular methods which are basically the most frequently used training methods for most of the image recognition tasks that are working on only seen classes. Another important thing is the unseen classes should not be confused with unseen data. Unseen classes at the test time are completely different classes at training classes and seen and unseen data can share the same class.

1.5. Auxiliary Features and Inference Methods

At test time in ZSL, some auxiliary information is provided in this work to increase the recognition rate of target, unseen classes. This type of extra information can be used in both ZSL, GZSL, TZSL methods to improve the prediction accuracy while comparing the model's output vector and target's label vector with Hamming distance method. Auxiliary features consist of 2 different feature types which are GIST features and 3D color histogram. These features are extracted from the whole dataset before the training process. These 2 features are created for every image of every class by creating a new file for each feature type but because of having a long line size in each file, similar features are removed. Feature size is reduced for every class by discarding most similar features from files with the help of Euclidean and

Bhattacharyya distance methods. Each feature line is compared with others to calculate similarity. This approach is applied to obtain only significant features for each class.

Hamming distance [34] is a metric between two binary label vectors. Model outputs label vector having float values. Each float value is converted into 0s and 1s by taking 0.5 as the threshold value. Then the remaining task is finding the most similar binary label vector from target classes for this binary label vector. After mapping is done, accuracy values are calculated and stored. The formula of Hamming distance metric is given in Equation 1.5. The weight is assumed as 1.

$$\frac{(\sum_{i=1}^{n} | Xi - Yi |)}{n} \; for\; i = 1, 2, \dots, n. \;\; X = \{0, 1\}, \;\; Y = \{0, 1\} \tag{1.5}$$

Because of consisting of binary values, subtraction operation in Equation 1.5 will always result as 0 or 1 and high or low numbers do not affect distance metric because of actually summing count of values that are not equal to each other. This total value can be divided by the total element count to limit its boundaries. This metric is essentially intended to be used in error detecting and correcting. In 1950, this metric, the Hamming distance has been proposed by R. Hamming in The Bell System Technical Journal [34].

In this work, the Hamming distance metric has been chosen to compare two label vectors. Also, some other distance metrics like Euclidean, Cosine, Manhattan, Minkowski are tested for the task of recognizing visual place images using feature vectors [35], [36]. After testing these metrics, the Hamming distance has been selected to be used in label vector comparison because labels can only have two values which are 0 as ‘No’ and 1 as ‘Yes’. Figure 1.6 is an example of its working principle.

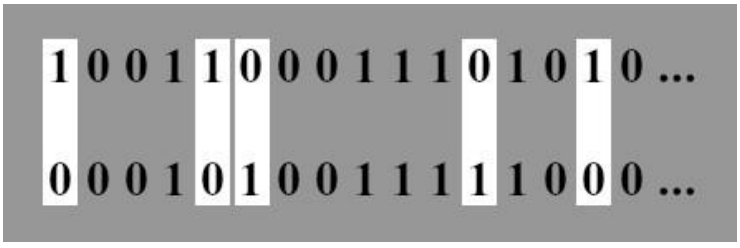

Figure 1.6. Applying Hamming distance on two attribute or label vectors

By taking into consideration only the differences between vectors and summing them, the most similar label vector can be found by selecting a vector that has less difference from the compared vector, and the target class of the selected vector is taken as predicted class at test time. This approach is similar to its original task which is error detecting. After this approach is applied into classes in phase 1, 2 other auxiliary features are processed in other phases of test time. Processing of them is similar to this phase because of using other distance methods for that tasks.

After the comparisons are done on label vectors which are performed in phase 1, 3D color histogram comparison is done in phase 2 of test time. Initially, all 3D color histogram features are extracted from all classes and written into a file. Then, these features are reduced because of considering comparison performance. Most similar features are removed by comparing all features with each other and in this way most distinct features are stored. This operation is like partitioning n items into clusters to do comparison operation only using these clusters, not looking into all the items. 3D color histogram includes information about all channels in RGB images.

To extract 3D color histogram information from images in this work, firstly, pixel values are read from all channels of images, then converted RGB image to BGR image which is simply done as changing the channel order. Then, pixel values of BGR image are converted into float values by subtracting 255 from pixel values. After this operation, BGR to LAB operation is done using OpenCV library [37], [38]. CIELAB color space is selected because of its similarity to human vision, having more natural colors, and representing some colors better than as in RGB. Bin size has been determined as 32, 32, 32 for each channel in CIELAB (L*a*b*) which are L, A, and B respectively. L (L*) range is between 0-100, A (a*) range is between -127-127, and B (b*) range is between -127-127 inclusively. With these configurations, 3D color histogram is calculated by again using OpenCV library and the result is normalized to limit the boundary of the histogram to 0 and 1 as min and max value respectively. The output 3D color histogram has been saved as a single line to the corresponded feature file that is stored in corresponded dataset directory. The benefit of using 3D color histogram is preserving color information which is important for especially current task.

Before extracting 3D color histogram, dominant colors were intended to be used but in city images, this type of information would not be sufficient. Also, 3D color histogram is not

sufficient by only itself but since it is provided as auxiliary, extra information, its help is aimed to be got on city images which have significant landmarks that have significant color information like Giza pyramids, Havana architectures etc.

After the 3D color histogram feature extraction is complete in phase 2, the comparison is done by using Bhattacharyya distance [39] metric between two color histograms at test time. The formula of Bhattacharyya distance metric is given in Equation 1.6. The result is subtracted from 1 because of measuring distance, not similarity. H_1 and H_2 are two 3D color histograms, d is the Bhattacharyya distance method and the total number of histogram bins are denoted as N.

$$d(H_1, H_2) = \sqrt{1 - \frac{1}{\sqrt{\overline{H}_1 \overline{H}_2 N^2}} \sum_I \sqrt{H_1(I) \cdot H_2(I)}} \tag{1.6}$$

With the help of Equation 1.6, the comparison is done on two float vectors which are 3D color histograms. This distance method which is named after Bhattacharyya, A. is suitable for histogram comparison and its range is between 0 and 1. A lower score means there exists high similarity and a higher score means vice versa. Also, there are some other distance methods which are Correlation, Chi-square, and Intersection but in this work, Bhattacharyya distance method is chosen to compare two 3D color histograms.

In phase 3 of the test time, vectors of GIST features which are global image features are compared. Oliva, A. and Torralba, A. explain the motivation of building a scene representation from global image features [3, p. 7] as following words:

"Evidence from the psychophysics literature suggest that our visual system analyzes global statistical summary of the image in a preselective stage of visual processing, or at least with minimal attentional resources (mean orientation, Parkes et al., 2001; mean of set of objects, Ariely, 2001; Chong and Treisman, 2003)."

Because human vision works this way, these global image features serve as summarized information of images. Instead of local features, global features can work better in city images because of having complex objects in them and the need of learning overall information from these images. In this work, initially, GIST features are extracted from all classes' images and stored in a file to make comparisons on them at test time. GIST features are extracted from images with the help of Oliva A. and Torralba, A.'s GIST descriptor [2], and Lear's GIST

implementation [40]. Figure 1.7 shows global and local receptive fields from an image that contains an architecture.

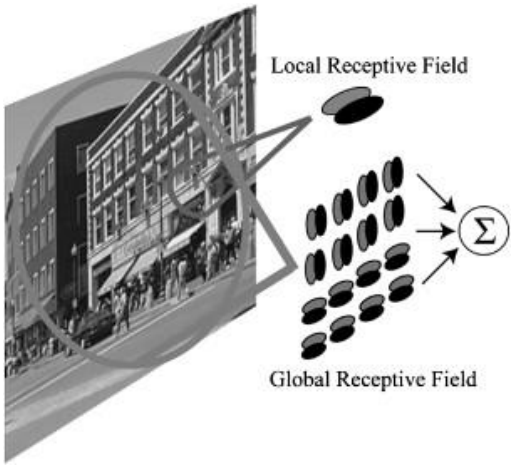

Figure 1.7. Illustration of global and local receptive fields [3]

After these features are extracted and stored in files, line sizes are reduced by comparing all features with each other to store only dissimilar and distinct ones by applying the Euclidean distance method. Euclidean distance method is selected instead of Cosine distance method in this step because of the need of measuring the actual distance and not the angle between vectors and taking magnitude into consideration. Equation 1.7 shows the Euclidean distance formula. The weight is assumed as 1.

$$\sqrt{(\sum_{i}^{n}(x_i - y_i)^2)} \tag{1.7}$$

Taking all this information into consideration and combining them at the test time, the inference process is done on target classes' images, and most similar city, country, and continent classes are aimed to be predicted correctly.

1.6. Contribution of the Thesis

The contributions of the thesis are mainly applying Zero-shot learning approach to recognize visual places which are unknown, making city, country, and continent-based

predictions on them, and using auxiliary features and distance methods at prediction time. CNN architecture is used for this task and found CNN models applicable to extract needed information from images. Also, a regular training setup is tested and compared with ZSL problem setup.

Because there are no or not sufficient work in literature based on this study in this thesis, still there are similar studies like [5], [6], [41], [42], [43], [44], [45], [46], [47], [48] or [49]. These studies are not worked with unseen places at the test time but they still show that predicting visual places is possible by using CNN architecture or other different architectures. Also, secondary, auxiliary features can be important for the prediction part of all the models. There are different approaches to this issue. Some studies use local image descriptors while other studies use global image descriptors based on their work areas and purpose to be achieved.

ZSL solves one, critical problem which is the reliance on huge datasets. Because ZSL and their variants GZSL, TZSL need fewer data to work on compared to traditional, regular training methods, this characteristic feature makes them one step further for some of the tasks.

One of the other contributions of this thesis is applying leave-one-group-out cross-validation approach to the ZSL task. By applying this technique, the class split is done by selecting only one class for the test time and assigning all the other classes to training time and this split is repeated K times which is determined as 6, 10, 14, and 18 in this thesis. To give an example to this split, if K is selected as 6, then there exist 5 city classes in the training set and 1 city class in the test set, and these split operations are repeated 6 times. The important thing is at the end of each step, a new and fresh CNN model is created to prevent learning old features from earlier steps. Accuracy values are recorded at each step and observation is done on the configuration. Of course, choosing classes is important, and to increase the prediction step, every 2 classes are chosen from the same country.

Another contribution of this thesis is applying the label-based approach to city, country, and continent classes. These labels consist of words which are describing parts of the images of the city dataset. At test time, labels are compared by using Hamming distance method and capturing target class names by selecting classes having minimum distances to the model's output label vector.

GIST descriptors and 3D color histogram features used in this work are in 1,536 and 32,768 lengths respectively. These features are included at test time in addition to the label comparison part. Also, these features are reduced into fixed-length after extracting each feature

from each image in the dataset. Euclidean distance method is used to measure the distance between GIST descriptors and Bhattacharyya distance method is used to measure the distance between 3D color histogram features.

In addition to ZSL, two Generalized ZSL approaches are applied. The first one is providing only test classes and both labels of test classes and train classes together at test time without including training classes. Initially, this approach is applied in this study because of splitting training and test classes into two disjoint sets and assuming that the task is predicting only target classes which are unseen classes by regarding all the labels from all sets. While this approach works, there is also another GZSL approach is followed. The other one is the original GZSL approach which is providing both training and test classes at test time with their related labels. Because both labels and image data from seen and unseen city classes have been provided at test time, a more generalized and realistic approach is reached instead of making a prediction on only test classes with both their related labels and labels of training classes.

Trained models have gained benefit from transfer learning by freezing weights of all the network layers except the final fully-connected layer. This type of transfer learning is named feature extraction. In this way, information gained earlier is preserved. These models have been pre-trained on ImageNet dataset earlier.

While processing images of visual places and making predictions on them, some issues are taken into consideration. These are selecting pre-trained CNN models which have not been pre-trained on classes that are the same as the study. The second one of them is not hyper-tuning on seen classes as including them in the validation set. Also, because seen classes and seen data are not the same in this context, seen classes can be defined as city classes that are included in the training set and seen data can be defined as a seen image on which the model is trained on. Thus, both seen and unseen data can have the same class. In this study, seen classes represent cities that are included in the training set and unseen classes represent cities that are included in the test set. Country and continent classes are mapped from city classes and their results are also examined throughout the thesis. Also, classes having the same continent or the same country are chosen to making the model capable of recognizing cities with respect to their origins easier. For example, two classes having the same country or the same continent are split into different disjoint sets. In this way, it is ensured that one city of a country and continent is in a set while the other city of the same country or continent is in another set.

1.7. Thesis Outline

The outline of this thesis is as follows: Literature review of similar studies with examples and their summaries of works are presented in Chapter 2. Chapter 3 explains all the details of the work done in this thesis including the used architecture, extracted features, some considerations, followed approaches and methodologies, splits of city, country, continent classes, all training, validation, and testing methods in detail. Chapter 4 shows the experimental results of this work with the figures and graphs. Also, comparisons between different problem setups including ZSL, GZSL, and regular training methods are made. After comparison, limits and shortcomings of the thesis are given and Chapter 4 is concluded by discussing possible improvements. In Chapter 5 which is the final chapter of the thesis, the work is concluded and summarized information of the work done is given.

2. LITERATURE REVIEW

2.1. Similar Works on Recognizing Cities or Landmarks

There have been various studies on visual place recognition and estimating geographic information from pictures. Some studies work on huge datasets while others work on small datasets but still, there is a need to collect many samples to create successful models with traditional, regular CNN training methods. Also, some studies create their own visual place image dataset by crawling images from the Internet using various sources like Google StreetView, Mapillary, Flickr, or image search engines. While there are some ready-to-use visual place datasets, they are still not applicable to every task and because of this, creating a new dataset can be needed based on the study that is worked on.

Zhang and Kosecka [46] have presented a system for image-based localization in urban environments. The goal of this study is to find the locations of the unlabeled images and to achieve this goal, three main stages are constructed. The first stage consists of location recognition using SIFT features which are based on local details to select the closest view to the query view. Euclidean distance is employed to find the closest descriptor and when the ratio of distances of the nearest and second nearest descriptors with the descriptor to be matched is below some threshold value, a match is considered. In addition to Euclidean distance, Cosine distance is also used as a criterion between descriptors when the cosine of the angle between descriptors are above again some threshold value. After these operations, the top 5 views of the largest number of matches with the query view are retained by the voting scheme. The second stage consists of camera motion estimation with robust estimation. Once the camera motion between the query view and the reference views are computed, at the third stage which is the final localization stage, the top 5 reference views are re-ranked and the top 3 views are chosen, then these views are reduced to 2 best views. At this step triangulation is tried to be applied between query views and reference views and the location of the query view is obtained based on all these criteria and processes.

Shi et al. [47] have investigated the features learned by deep learning architecture which is CNN for this study. VGG11, Resnet18, and some other networks are trained on the two datasets which are Tokyo 24/ 7 [50] and Pittsburgh 250K [51]. In the study, Grad-CAM is used to highlight regions of the test images, t-SNE is used for visualization and clustering, pre-trained

models are used to annotate the objects, and the existence of significant skew towards certain objects is investigated. This study has been concluded by showing the influence of the network architectures on the interpretability of CNN features.

Suresh et al. [42] have demonstrated photo localization with a deep neural network which deep neural network is named as DeepGeo and based on ResNet in this study. The model is trained and tested on newly created datasets, 50States10K and 50States2K respectively. In addition to the standard model, 3 different integration methods are presented which are early integration, medium integration, and late integration. In each of these integration techniques, CNN models work on a group of 4 views that are multiple views of the same location. Also, to show the performance of the proposed network, tests are performed on humans by recording their scores on GeoGuessr game which is a location discovery game.

Hays and Efros [5] have proposed a set of methods and an algorithm to estimate geographic information from an image using a data-driven approach. For this study, over 6 million GPS-tagged images are gathered from Flickr and various features are extracted from images to build a network called Im2GPS. These features are CIE L*a*b* color histogram, texton histogram, line feature, GIST descriptor, and geometric context. Also, as a secondary geographic task, land cover estimates are done as estimating probabilities of having forest, water, mountain, desert, or some other characteristics in the images. The study has shown that most geographically discriminative features are the GIST, color histogram, and texton histogram. Also, some other questions like data are helping or not, how accurate the estimates are answered and different sizes of localizations like city, region, country, continent, or diameter of the earth are selected for comparison. Localizing about 25% of the data within the scale of a country is observed. Some approaches like First Nearest Neighbor Scene Match, Mean Shift Mode, Random Scenes, etc. are followed for the estimations. In addition to all these works, the same authors have created a new version of Im2GPS by adding new features to it [49].

Weyand et al. [6] have created a model which is called PlaNet. More than 125 million images are obtained for this task using geotagged Flickr images. Geolocation scales such as street, city, region, country, continent, and earth diameter are taken into consideration while measuring the geolocation error and obtaining accuracy values for top predictions. The advantage of this model is being trained on very huge data and compared to Im2GPS model [5], PlaNet has some significant advantages like being able to perform better on recognition at

different levels and partitioning earth's surface into cells using Google S2 Cells. The created model is based on CNN architecture and also some other approaches based on different architectures such as LSTM architecture are tested. The model is capable of localizing a large portion of the earth based on the trained data.

Vo et al. [44] have combined deep learning classification methods including dividing the world into cells to predict the correct cell for a given image using the original Im2GPS approach. This combined approach has shown better results compared to both PlaNet and Im2GPS models. The original dataset from Im2GPS is used as the same and for the testing, 2 different test sets are constructed. The authors have presented a deep learning study on image geolocalization and concluded having state-of-the-art accuracy values based on different datasets.

As Schindler et al.'s paper [43] has shown, finding the location of a query image can be possible by constructing a vocabulary using the most informative features. Also, there are numbers of tree search algorithms are introduced. A new dataset is created by driving through a city for this task and obtaining 10 million images. All the images have related GPS coordinates and compass heading. A voting scheme is proposed to find the best-matching database image for the query image. The study has shown that there is a relationship between the performance of a vocabulary tree and the vocabulary chosen. This vocabulary tree is used as organizing and searching feature descriptors instead of explicitly storing them. 100 million features are stored and organized in this tree structure as consisting of the most informative ones. Also, some optimization techniques are given to improve the performance of the structure. Numerous experiments are carried out and results showed that the structure can achieve 65% - 80% recognition rates based on the comparisons done.

Hershey and Wulfe [41] have trained a CNN model which is named LittlePlaNet to recognize city images. To achieve this purpose, 100 thousand street-level images are obtained from Google Street View. GoogLeNet model [52] which is pre-trained on MIT Places205 dataset [53] is chosen for the study. The transfer learning approach is applied while some issues like which layers to retrain are taken into consideration. By selecting a pre-trained model, faster training is achieved. Also, some data augmentation techniques are applied to images. In addition to the Google Street View dataset, some tests are performed using another dataset which consists of Flickr images to test the robustness of the model. As expected, the model does not perform well on this different dataset. Before concluding the work, feature and image visualizations are

investigated, heatmap and saliency masks are created for finding out both which regions of the images are important for making predictions and which cities are performing better.

The study by Camara and Přeučil [45] on visual place recognition has been performed using spatial matching and image filtering. The methodology presented is named Semantic and Spatial Matching Visual Place Recognition (SSMVPR). There is a two-stage system that consists of database creation by processing images, extracting features using CNN at the first stage, and later querying images, again extracting features and selecting corresponded images from the databases based on scores at the second stage. Various datasets are used throughout this study and pixel-based recognition is taken as a basis. Also, there exist some studies like [54], [55] which are similar to the [45] in terms of matching two images but they are regarding features and shapes based on orientation, scale, and affine.

By the study of Chen et al. [48], authors have performed visual place recognition with the created dataset which consists of 2.5 million images by the authors. CNN architectures are used for this purpose and the behavior of the models on the different appearance and viewpoint variations of the images are observed. Visualizations of the weights and layer activations are illustrated to investigate the viewpoint change robustness. The created dataset which is called Specific Places Dataset (SPED) includes environmental variations like season changes and lightning variations. These variations are important for visual place recognition because they can affect results but the results also depend on the dataset. If the variations of the same place are present in the dataset as in this study, then the model can detect the same features as shown in Figure 2.3.

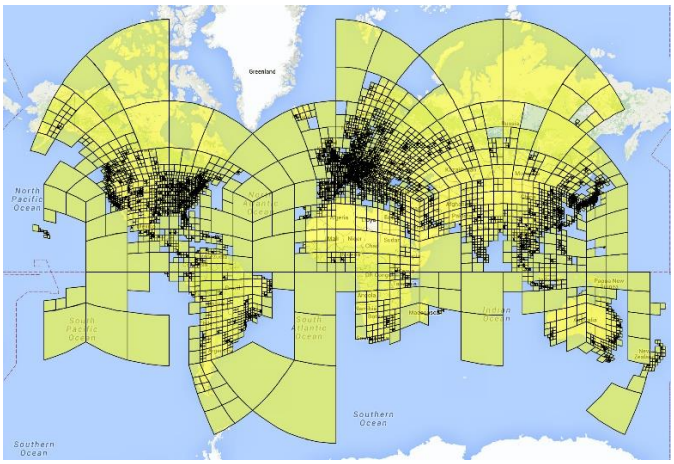

Figure 2.1. Partitioning earth using S2 cells [6]

Visualization can be helpful for both seeing the details and ensuring the model trained on image recognition tasks is working as expected. S2 partitioning that is used in some studies is shown in Figure 2.1. Cells nearly fully cover the lands of the earth. This approach enables the models to work on the whole earth and doing estimations based on the cells. Each cell size can be different based on the population density. Also, t-SNE visualization for the features in the study [41] is given in Figure 2.2. This visualization clearly shows the distinct cities based on the features. Because both image features and auxiliary features are important for the recognition tasks, the distinction between cities regarding these features should be investigated throughout the study. Figure 2.2 shows the features which are extracted from the model. Figure 2.3 shows that the model focuses on the same shapes and features on different variations of the same place. This visualization is intended to detect the robustness of the model in terms of viewpoint changes.

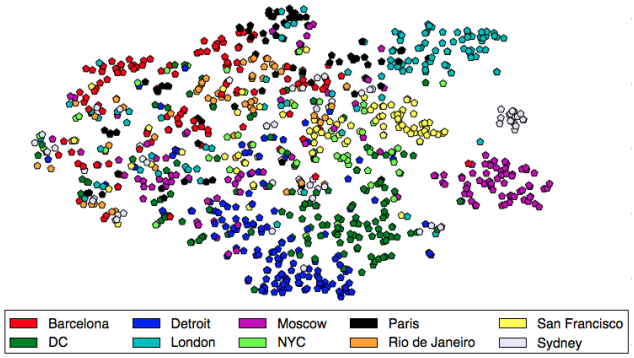

Figure 2.2. t-SNE visualization of features [41]

Figure 2.3. Different viewpoints of the same place and firing of the same convolution filters [48]

Some of the studies like [28] have followed a label-based approach. Each class is annotated with related labels. These attributes or simply labels are defining each class based on their behavior, features, or characteristics. From the study [28], Figure 2.4 shows the example of the attributes that can be used in ZSL context.

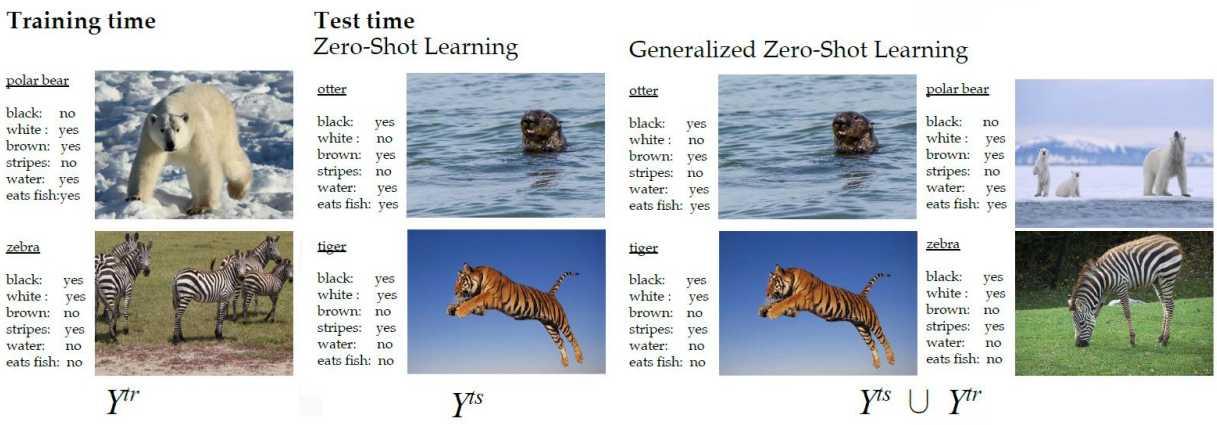

Figure 2.4. Different stages with corresponded attributes [28] (Figure is the upscaled form of the original figure in the study [28])

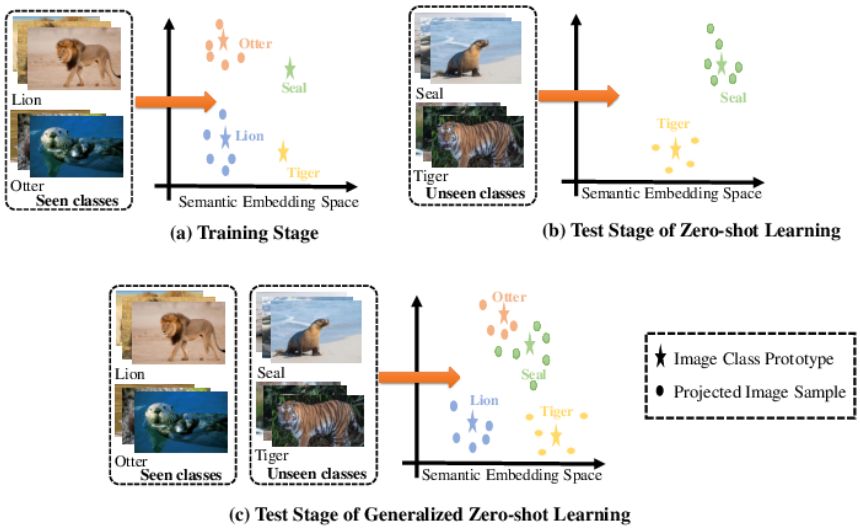

Figure 2.5. Schematic illustration of ZSL and GZSL [29]

Figure 2.5 from the review study [29] shows the differences at different stages in different configurations. Semantic embedding space sets, projected image samples, and image classes can be found in this schematic diagram. The diagram is adapted from [56] in that review study.

In that ZSL study, similar approaches to this thesis are followed such as using attribute annotations and transferring knowledge from one class to another.

2.2. Problems in Similar Works

All the similar studies related to this thesis topic are performing well regarding criteria, methods, datasets, localization regions that are worked on. Since there are various metrics to measure the success of the models and also improve their success, there are still problems in the visual place recognition context.

One of the problems is the need for a huge dataset having many samples of the classes that are worked on. Creating a huge dataset for visual place recognition is problematic in many ways. First of all, the level of detail on the images should be determined. Visual place datasets can have street-level images or landscape images in them. Also, the cover of the earth is important and source and target classes should be selected with some consideration. Another thing is the sample count of each class. There needs to be a sufficient number of samples for each class. In addition to these, images should be crawled regarding GPS locations, tags, labels, or titles. This image gathering process is hard because there can be noisy images like non-related or corrupt ones. The large majority of similar studies have their own dataset because one dataset working for one task may not work well for another task.

Another problem is no model can be capable of recognizing a large portion of earth no matter how many images are fed into the training process. It is not possible to gather images from all over the world covering nearly all the places. For this reason, many similar studies have focused on some specific portion of the earth like focusing on only one country. There are some studies like [6], or [42] which have focused on a large portion of the earth. But even these models perform better than other studies, there needs to be a lot of work done in this context to cover many regions of the earth.

The other problem can be learning features from all the classes instead of predicting target classes based on the learned features from source classes that are trained on the model. This disjoint model approach used by ZSL has benefits in terms of requiring fewer data and predicting similar future locations based on the learned locations. This problem is also related to supervised learning and there may be a need to labeling or reviewing some images manually.

Also, since streets of different cities, countries contain similar shapes like roads, parks, sky, sidewalk, people, etc. the training and prediction done based on these types of images may not be enough to differentiate cities. Some significant landmarks or places that have some significant architectures, shapes may be better for recognition tasks. Some studies have focused on pixel-based matching to get similar place images based on the query image. For general visual place recognition, there needs to be more than pixel-based matching and there needs to be a generalization based on the learned features to recognize different places better. For street-level images, some figures from the study [41] are given in Figure 2.6 and Figure 2.7. Since cars and architectures have an effect on the trained model, the recognition rate is affected positively when these features are detected in the images.

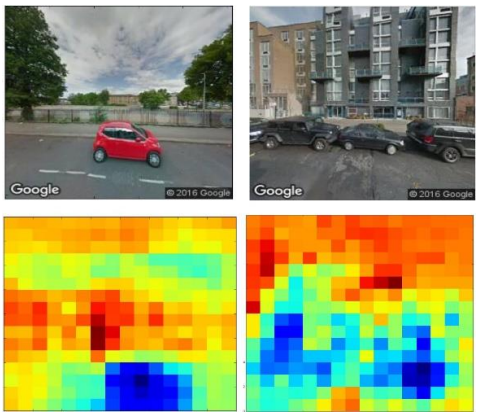

Figure 2.6. Visualization of occlusion heatmaps [41]

Figure 2.7. Visualization of inclusion heatmap [41]

Figure 2.6 and 2.7 show that cars and architecture can be learned as a feature from the images. Inclusion and occlusion techniques are applied in [41] to test this behavior. The occlusion technique consists of removing parts of the images repeatedly and detecting which parts of the images are contributing to the model more than the others. Inclusion is the reverse of this occlusion technique. It consists of including parts of the images repeatedly to detect again

which parts of the images are important for the model. Also, the weather of the images can impact the output of the model. Since images taken at day, night, or having different weather can all affect the results, this is another problem for visual place recognition. With these visualizations of the features and activations, they can help confirm the learned features of the model.

2.3. Discussion and Comparison

Traditional methods have focused on knowing all the classes at first but in real-life applications, it may not be possible every time. Sometimes target classes may be unknown at that time but they can still be predicted based on the previously learned classes. Visual place recognition is a good example of this problem. Because covering all the visual places of the earth is hard, prediction based on similarity is important to overcome this problem.

Zero-shot learning approach is suitable for the problem domains which consists of unknown targets. Some of the studies performed good outcomes based on the huge datasets. These datasets are generally created specifically for the task but some of the other similar tasks also used these datasets to perform benchmarks on their results. Studies [5], [6], and [44] are the most similar ones to this thesis in terms of their work area and used techniques.

In this thesis, there are 3 different types of results which are cities, countries, and continents respectively. Results and accuracy values of these groups are examined independently. Country and continent classes are mapped from city classes. There are similar studies that are worked on these regions of the earth too. Some of these studies have also examined street-level results. In this thesis, the baseline of the classes are cities, and these city images are pre-processed, filtered, and cleaned before being fed into the model to be trained. In Figure 2.8, some of the images from the World Cities dataset [54] are given. In this thesis, a subset of this dataset is created and used.

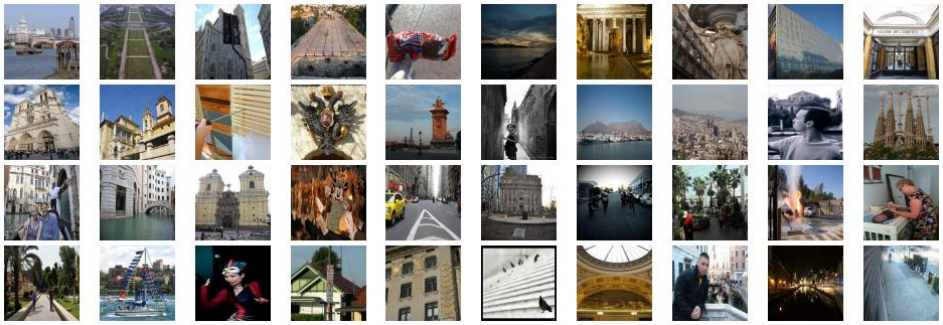

Figure 2.8. Sample images from the original World Cities dataset [54]

Similar to the World Cities dataset [54], there are some datasets that are used by similar studies and have focused on cities and their properties. Some of them are Places205 [53], Tokyo 24/7 [50], Pittsburgh 250K [51], 50States10K [42], Oxford Buildings [57], Paris [58] and Holidays [59]. There exist specifically created datasets that are used in similar studies based on their focus on city classes. There are also some datasets consist of images of visual places but not focused on learning city features and focused on more scene understanding like Cityscapes [60], InstaCities1M [61], SUN [62], and Archive of Many Outdoor Scenes (AMOS) [63], [64]. In this study, the newly-formed dataset has 15,000 city images which are obtained from both the dataset in Figure 2.8 and Flickr. When ZSL is applied, this number is also decreased at training time because of splitting classes into disjoint sets. This study is aimed to recognize cities at an acceptable rate while working on fewer images than the other similar studies.

Some of the similar studies have focused on splitting the dataset into training and testing. In this thesis, also leave-one-group-out cross-validation split technique is applied to observe the changes when the classes are changed in addition to the training-validation-test split. Cross-validation is a good technique if all the possibilities of data and class splitting are needed to be tested. In this study, when ZSL is applied, classes are split into disjoint sets. For this reason, the technique which is called leave one group out is used by leaving only one class out at each step. Every time, only one class is present in the test set, and all the remaining classes are added to the training set. Also, the model is recreated every step of cross-validation.

Some of the ZSL studies have split training and test labels, at training time only the labels of the training classes are regarded and at test time only the labels of the test classes are regarded. While this approach does not violate ZSL and shows good results, the generalized approach is more realistic because of reflecting real-world conditions.

While some studies have focused on recognizing street images, some other studies have worked on recognizing complex visual places like landscapes, nature images to make a prediction from characteristics of different cities. Recognizing visual places task is similar to scene understanding and in terms of processing images with different objects and features in them.

Using labels or attributes is also a different technique that is used in different ZSL and visual place recognition studies. Farhadi et al.'s paper [65] shows describing objects by their attributes. This attribute-based approach can work on ZSL if attributes are selected based on their meaning on the images. In this thesis, labels (i.e. attributes) for visual places which are cities are created regarding the objects included in the images. These labels can define a city picture at some level that is enough to differentiate based on the characteristics like differentiating a coastal city with an inland city. Some papers in the literature regarded behaviors (ex. flies, jumps) when live creatures are worked on. Some of the other studies have focused on attributes (ex. green, big). In this thesis, labels which are simply names to define included objects in the images (ex. ocean, boat, home).

In this study, in addition to gathering images from the World Cities dataset [54] and Flickr to create a subset dataset, also street-view images are obtained from Mapillary and Google StreetView to create a street-level image dataset but considering repetitive street-level images are not a good candidate for the task in this thesis, this dataset is not used. Visual place recognition tasks are sometimes dataset dependent based on the purpose and because of this, the creation of the datasets is an important step and as most of the current studies have created their own datasets, a new dataset is created in this thesis too to make a prediction on the total of 42 cities from various countries and continents. Some of the street-view images can be found in Figure 2.9. The illustration is created using Kapwing online content creation platform. As it is seen from the figure, most of these street-level images contain roads, cars, trees, people, or parks. While recognition of them is possible, covering world cities with their architectures, ocean, landmarks are assumed as a better option in this thesis. Also, each of these images is obtained with random GPS coordinates based on original GPS coordinates of city classes and this street-based dataset may be used for recognition tasks other than recognizing world cities.

Figure 2.9. Some street-view images from the created street-level dataset

Figure 2.9 is given only for illustration purposes. While this street-level dataset is not used throughout this thesis, the illustration of some images from this dataset is given and the gathering of these images to create this dataset is explained in this thesis.

Because the main motivation of this thesis is to recognize cities that are not trained earlier but are similar to the trained cities in terms of shape or architectural structures, in the ZSL configuration, it is determined to include two cities of the same country to detect the similarities of the buildings, landscapes, etc. Some of the examples are given in the following figures which are Figure 2.10 and 2.11.

Figure 2.10. Example of similarities of different cities

As it is seen, there are some similarities in terms of architecture in different cities as they are shown in Figure 2.10, 2.11, and 2.12. The left image in Figure 2.10 is from Beijing while the right image in the same figure is from Hong Kong. In Figure 2.11, the left image is from

Venice while the right image is from Rome. As the last example, in Figure 2.12, the left image is from Berlin while the right image is from Dortmund.

Figure 2.11. Example of similarities of different cities

A very little portion of the images are in grayscale format and they are converted into RGB format and fed into the model. Also, in the figures starting from 2.13, separate channels of RGB, CIELAB color images are shown to illustrate the differences.

Figure 2.12. Example of similarities of different cities

Figure 2.13. Separate RGB color components

Figure 2.14. Separate CIELAB color components

Figure 2.15. Separate CIELAB color components after adjusting contrast

The order of the channels in the Figure 2.13 is R, G, B respectively and in Figure 2.14 and 2.15 are L* (lightness, white color component), a* (green-red color components), b* (blue-yellow color components) respectively [66].

3. MATERIALS AND METHODS

3.1. Gathering of Images and Creation of Dataset

Before creating datasets, the city class count is determined as 42, and latitude and longitude coordinates are obtained for all the city classes. For these latitude and longitude values, World Cities Database which is created using SimpleMaps, Pareto Software, LLC [67] is used. After determining the total class count and obtaining all the GPS coordinates, the radius is determined to obtain the images with respect to this value to gather related city images.

Two different datasets are created in this thesis. The first one which is based on street-level images of cities is created by obtaining images from Google StreetView and Mapillary. Google StreetView API enables obtaining 256x256 images. The radius is determined as 5,000 meters and the field of view is determined as 120 degrees for Google StreetView API. For each latitude and longitude, Google StreetView API is called and images are obtained. For Mapillary API, the radius is determined as 10,000 meters and downloading panorama images is prevented by the API parameter. Mapillary Object Detection V3 API is used to obtain these street-level images and the labels "construction—structure—building", "construction—structure—bridge", "nature" which are used for segmentation in the API are provided as URL parameters to obtain images consist of mostly construction structures and nature. The score is determined as 0.95 to get only related images. By providing these parameters, images having 320x240 pixels are crawled. Also, cleaning downloaded corrupted images are performed after obtaining images for each city. Various parameters are provided to enhance the obtaining of related city images for these street-level based imagery platforms' APIs.

The second dataset is created by combining images from both Flickr and an external dataset because after creating a street-level based image dataset, some experiments are done on this dataset and the results and repetitive images like road images caused the need of creating another dataset. This new second dataset is created specifically for this thesis as a subset of the World Cities dataset by National Technical University of Athens [54] which originally consists of more than 2M images and 40 cities. To increase the count of the city classes of the newly-formed dataset, 2 extra cities are added into the dataset. Also, some new images are obtained

from Flickr based on the GPS coordinates, radius, title, description, and tags of the images on Flickr to augment the dataset.

The new dataset is created both manually and automatically. At the manual obtaining step, images are selected with respect to their details from the World Cities dataset. At automatized obtaining step, Flickr API is used to crawl images based on their latitude and longitude. Also, the city name is provided as a URL parameter to search in the title, tag, and description of the images. Various metrics are also provided at this step too to obtain only related city images and to prevent images of wrong places. These platforms which are Flickr, Google StreetView, and Mapillary are often used in similar studies too.

Also, while obtaining images based on their latitude and longitude, random latitude and longitude coordinates are generated at each step to obtain different images from different regions of the determined boundaries. As a result, this new dataset which is used throughout this thesis is created with the help of both the World Cities dataset [54] and Flickr. A total of 42 cities, 32 countries, 6 continents, and 15,000 images are present in this dataset. Some of the images are given in Figure 3.1 for preview purposes.

Figure 3.1. Illustration of images in the created dataset

All the images in this dataset have JPG extensions that have no transparency layer and they are in various sizes. The minimum size of one edge of the images is determined as 256 while obtaining images. These images are pre-processed before being fed into the model as training, validation, or test data.

In addition to creating the dataset having this size, dataset augmentation techniques are also applied to increase the training set size. These dataset augmentation operations include random resized cropping, color jittering, random rotation, horizontal flip. The training set is duplicated in this way. These operations are applied to the whole training set which is a duplicate of the original dataset. The original dataset remains the same while these dataset augmentation operations are being applied to the images of the duplicated dataset. This approach is followed to create more samples of the city images and to enhance the model recognition rate.

3.2. Dataset Splits

The dataset is split into different sets which are used at training, validation, and test steps using different methodologies. There are two main splits that are related to the validation process. The first one is the traditional one that consists of splitting the dataset into three parts which are training, validation, and test sets. The other one is a cross-validation method which consists of splitting the dataset into two different sets at each validation step regarding the k-fold number. In addition to these splits, different approaches are followed to select classes by determined ratios.

In ZSL and traditional training configurations, different splits are applied to the dataset. Of course, because traditional training works on seen classes, only unseen data is included at test time to make predictions on them regarding all classes. In ZSL, the dataset is basically split into different sets which include seen and unseen classes separately. In GZSL configuration, half of the training set is used at training time and the test set, and the other half of the training set is used at test time.

In traditional training configuration, when all 42 classes are chosen, train, validation, and test ratio is selected as 70%, 15%, and 15% respectively. When only European cities are chosen, these ratios are changed as 60%, 20%, and 20%. These ratios are applied to limit data size in the traditional training process. In ZSL, these ratios are applied to limit class sizes to perform training and testing on them.

In ZSL training configuration, both all classes and only European classes are split into different disjoint sets. In the cross-validation procedure in ZSL, k-fold values are determined as 6, 10, 14, and 18 to observe the accuracy values in each configuration. In each of these configurations, all the classes except one are included at the training time and only one class is included at the test time. Validation is done as changing training and test set at every step and the model is being trained is recreated at beginning of every step.

Splitting of data and classes is important to ensure that model is learning source classes and making generalizations on target classes in the right way. In traditional training configuration, because all the classes are shared by different sets, image data should be split into different sets to prevent training and testing on the same image data. Also, disjoint sets are created in ZSL configuration to prevent violating the rules of ZSL. Sometimes subsets having different class sizes are created from the original dataset throughout this study to observe the effect of class choices on the recognition rate.

Table 3.1. Total city images in each split

Size / **Setup**	**Total size**	**Training set size**	**Validation set size**	**Test set size**
Traditional	15,000	10,633 x 2	2,258	2,109
Traditional (Only 23 classes which are European cities)	8,555	5,097 x 2	1,702	1,756
ZSL (Only 6 classes) (*)	2,366	1,959 x 2	0	407
GZSL (Only 6 classes) (*)	2,366	980 x 2	0	407 + 979
ZSL (Only 10 classes) (*)	3,735	3,328 x 2	0	407
GZSL (Only 10 classes) (*)	3,735	1,666 x 2	0	407 + 1,662
ZSL (Only 14 classes) (*)	5,046	4,639 x 2	0	407
GZSL (Only 14 classes) (*)	5,046	2,323 x 2	0	407 + 2,316
ZSL (Only 18 classes) (*)	6,400	5,999 x 2	0	401
GZSL (Only 18 classes) (*)	6,400	3,005 x 2	0	401 + 2,994

In Table 3.1, total image counts regarding different splits based on selected class sizes are given and details about different setups are shown.

Even image counts that are marked with (*) in Table 3.1 vary based on selected classes, they mostly preserve the ratios in the table because of having a balanced count of images in each city class. Counts of training set sizes are multiplied by 2 because of dataset augmentation. Zeros in the validation part indicate that there is no specific validation dataset because of changing training and test set at every step in the cross-validation technique. Also, even the image sizes are less in the last 2 rows, the model can still learn and is able to recognize cities. In addition to 6 class setups, 10, 14, and 18 class setups have also been experimented as they are shown in Table 3.1. The training set is divided in half in GZSL (as the original GZSL approach) and one of the halves is included at training time while the other half is included at test time to make both training and test sets and their labels available at the test time. The division operation is done by dividing image data regarding their indexes. Even and odd indexes are split into different sets to accomplish this goal.

3.3. City, Country, Continent Classes and Grouping

There are 42 cities, 32 countries, 6 continents present in this study. At both training and test stages, some groupings are done on cities regarding their countries and continents. This grouping is done to enhance the recognition process. For example, having a total of 4 cities in the original set of ZSL setup including Berlin and Dortmund as cities of Germany and Florence and Venice as cities of Italy ensures that at least one city of one country is in the test set while the other city of the same country is in the training set. Even each city has its own characteristics, different cities of the same country may include similar architectures like having mosques, cathedrals, or Chinese architectures like pagodas. This approach is followed while choosing and splitting cities into disjoint sets.

Illustrations of cities, countries, and continents included in the dataset are given in Figure 3.2, 3.3 and 3.4. These figures are generated regarding the original dataset which includes 42 cities but keep in mind that some subset versions of this dataset are created throughout this study as having fixed size of cities like having only 6 city classes or 18 city classes. Illustrations are created using SimpleMaps, Pareto Software, LLC [68].

Figure 3.2. Cities included in the created dataset

The cities in the dataset are Amsterdam, Athens, Barcelona, Beijing, Berlin, Budapest, Buenos Aires, Cairo, Cape Town, Caracas, Chicago, Copenhagen, Delhi, Dortmund, Dubai, Dublin, Edinburgh, Florence, Havana, Helsinki, Hong Kong, Istanbul, Jakarta, Krakow, Lima, Lisbon, London, Los Angeles, Madrid, Melbourne, New York, Paris, Prague, Rome, Sao Paulo, Sheffield, St. Petersburg, Sydney, Toronto, Vancouver, Venice, Vienna.

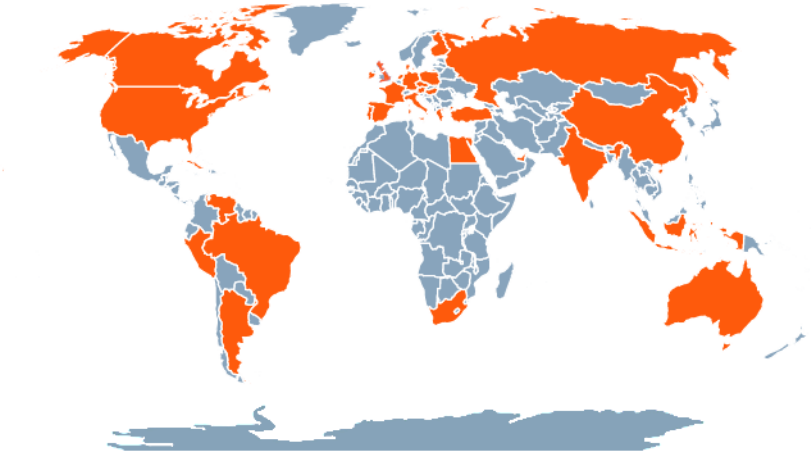

Figure 3.3. Countries included in the created dataset

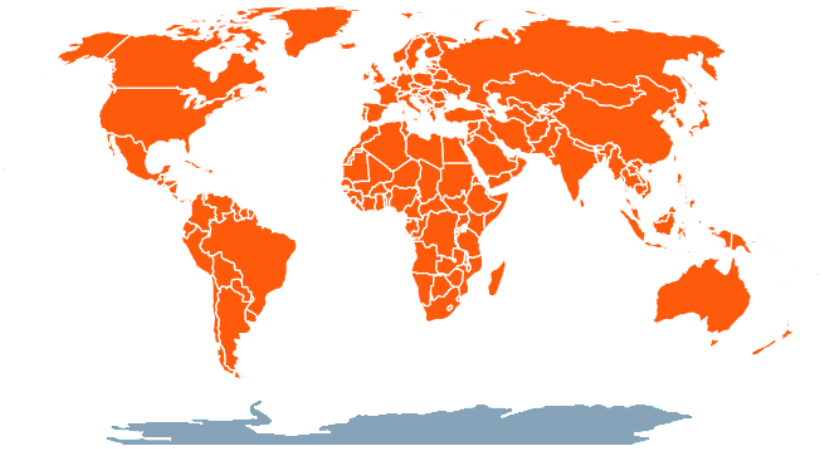

Figure 3.4. Continents included in the created dataset

The countries of these cities in the dataset are Argentina, Australia, Austria, Brazil, Canada, China, Cuba, Czechia, Denmark, Egypt, England, Finland, France, Germany, Greece, Hungary, India, Indonesia, Ireland, Italy, Netherlands, Peru, Poland, Portugal, Russia, Scotland, South Africa, Spain, Turkey, United Arab Emirates, United States, Venezuela and finally the continents in the dataset are Africa, Asia, Europe, North America, Oceania, South America.

3.4. Some Considerations

There are some considerations that are taken into throughout the thesis. Some of them are selecting 3D color histogram and GIST features as auxiliary features to detect similar colors from similar architectures and detect similar global features from city images. As explained in Chapter 3.3, some groupings are applied to cities for enhancing recognition purposes. Besides, since class size is important, experiments are done on different class sizes by creating subset datasets from the original dataset. Original GIST and 3D color histogram features which are extracted from each image in the dataset are cleaned to have only distinct features by eliminating similar ones in the same city features set. To include most of the countries and continents of the earth, cities are selected and images are obtained for them regarding this purpose. Pre-trained models are chosen to preserve previously learned features from the huge image dataset. Transfer learning is useful when tasks are similar even the objects that are being worked on are not similar. By freezing all layers except the fully-connected layer in CNNs, the model is trained as a feature extractor based on the previously learned features. Because training model from scratch is difficult, this approach is used and various information are added to enhance the recognition

like 3D color histogram, GIST features, or label vectors which are compared later with distance metrics like Euclidean, Hamming, or Bhattacharyya.

3.5. Used Convolutional Neural Network Architectures

Initially, 3 CNN architectures which are ResNet50, VGG16, and InceptionV3 are prepared for the current study and after seeing that VGG network achieves the best performance on some challenging datasets in the benchmarks of the Places365 study [69], VGG16 is chosen for the main work, and all the processes are done on VGG16. Transfer learning is applied to pre-trained VGG16 model architecture as using it for feature extraction purposes. The model architectures for different configurations are given in Figure 3.5 and Figure 3.6. These figures are generated using ConvNetDraw [70] by modifying them to adapt for visualizing the model used in this thesis.

Each layer has different width due to the size of feature maps. The filter size is fixed at 3x3 in these layers. Colorings of layers are done to differentiate the layers. There are total of 5 pooling layers. The first shape is the input image which is 224x224 in size. This input image size is also the same in ResNet50 but different in InceptionV3. InceptionV3 architecture works on input images that are 299x299 in size. These input images are also RGB images having 3 channels. If there is a need to work on transparent images (ex. PNG images) having an alpha channel, then the input layers can be modified to support 4D images.

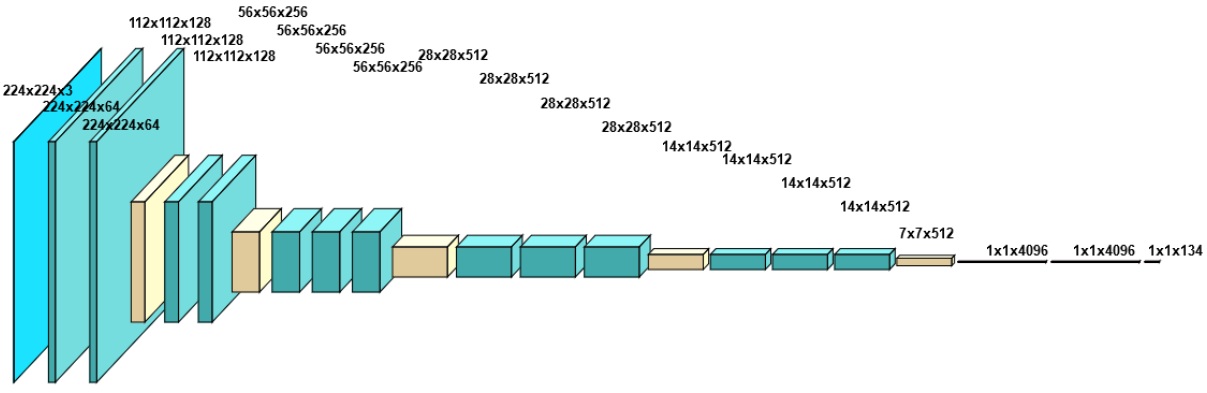

Figure 3.5. Main layers of the model architecture used for the ZSL configuration

The last layer of the architecture used in ZSL configuration is a flattened vector in 1x1x134 size. Because there are total of 134 labels, this layer outputs 134 different float values

after Sigmoid activation function is applied to them. Sigmoid works as a squashing function and ensures limiting the output values to the range between 0 and 1. Then by Hamming distance method, these 134 Sigmoid probabilities are compared with 134 binary labels for each city by thresholding. The threshold value is determined as 0.5 and greater or equal to 0.5 values are replaced by 1 while other remaining values are replaced by 0. Then, of course, other auxiliary features are provided and compared at test time in addition to label comparison. Both ZSL and GZSL configurations use Sigmoid while outputting 134 different probabilities.

The difference between Figure 3.5 and Figure 3.6 is the last layer. Softmax is applied at the last layer of traditional training configuration which outputs 42 probabilities for a total of 42 cities. This number can also change due to selected classes.

There are variants of VGG model architectures. In this study, a variant of VGG16 model architecture which is called VGG16 with Batch Normalization (BN) [71], [72] is used. In this variant, each convolution layer is followed by one batch normalization layer. Figure 3.5 and 3.6 are the same in VGG16 configurations with batch normalization or not. Batch normalization layers are not shown in the figures as the figures are showing only the main layers for simplicity.

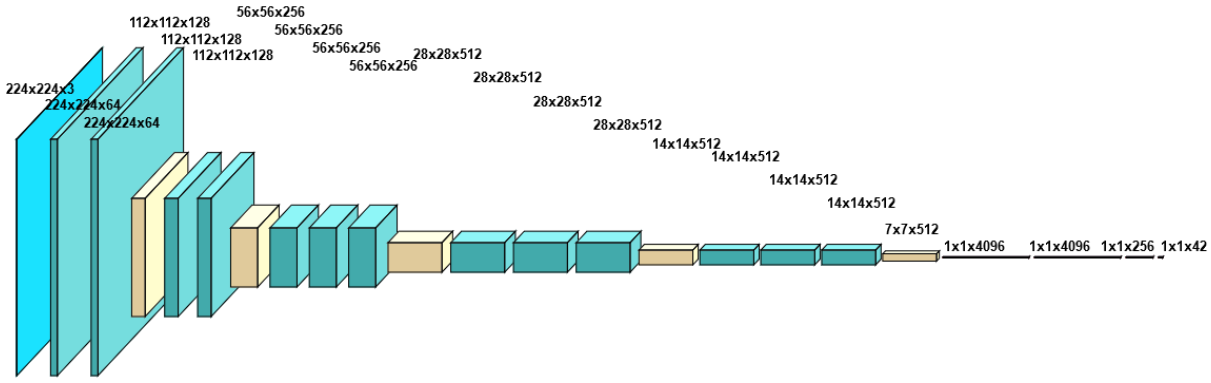

Figure 3.6. Main layers of the model architecture used for traditional training configuration

Adagrad optimizer [73] which is a gradient descent optimization algorithm is selected to be used throughout the study. Learning rate and weight decay are determined as 0.0025 and 0.001, respectively. These values are selected after doing experiments and observing the effect of changing these values to the accuracy values and recognition rate. Even these values are manually determined, the main benefit of AdaGrad algorithm is eliminating the need of tuning the learning rate manually [74]. There are also other gradient descent optimization algorithms

such as Momentum, Nesterov accelerated gradient, Adadelta, RMSprop, Adam, AdaMax, and Nadam. Each of these algorithms has its own characteristics and the choice of using them depends on the subject, task, input data, etc. Because there are both advantages and disadvantages while using them, the selection of one of them should be made carefully.

Also, to see the important regions of the city images, Gradient-weighted Class Activation Mapping (Grad-CAM) [75] is applied to some of them. The illustrations are generated using pytorch-grad-cam [76] and can be found in Figure 3.7 and Figure 3.8. Also, Eigen smoothing is applied to reduce noise in the CAMs by again using pytorch-grad-cam [76].

In Figure 3.7, an image from Hong Kong region is processed. The left one is the Grad-CAM result while the other two are Grad-CAM with Guided Backpropagation results. The middle one has focused more specifically than the right one. In Figure 3.8, the order is the same as Figure 3.7 and the images are from Venice, Italy. Both figures show the important regions that the trained model has focused on which are mostly architectural structures. Also, both figures are generated using the same trained ZSL model.

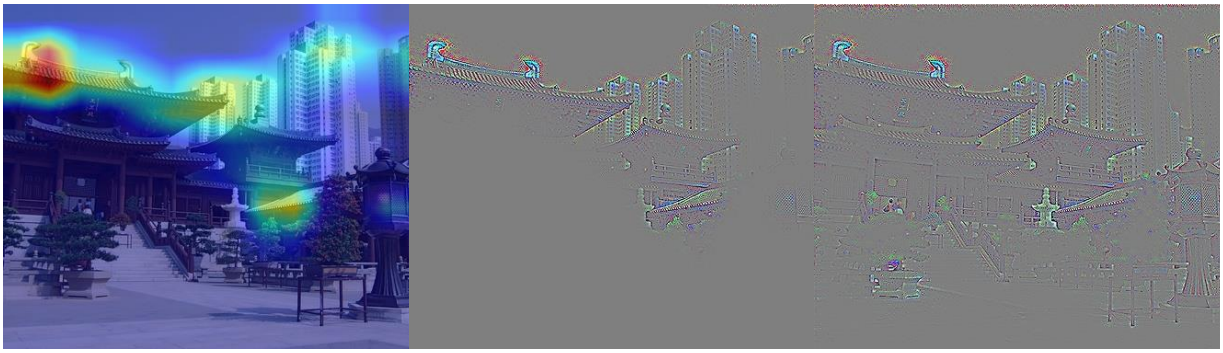

Figure 3.7. Grad-CAM results for an image belonging to Hong Kong as a seen class

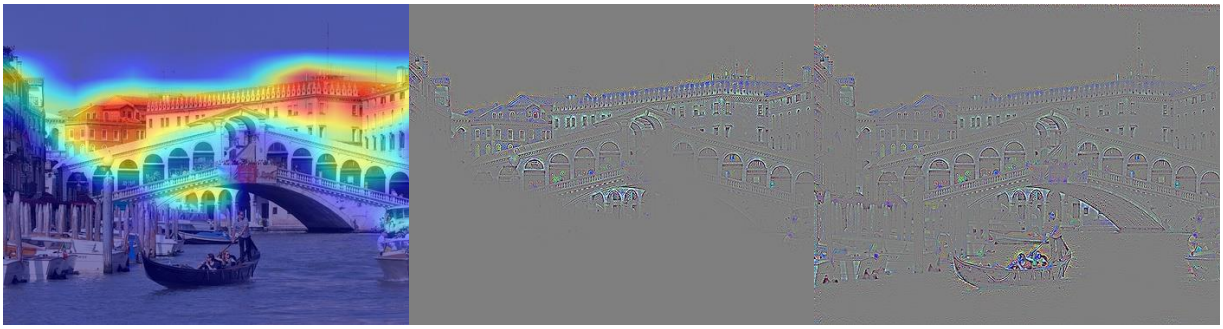

Figure 3.8. Grad-CAM results for an image belonging to Venice as an unseen class

VGG16 with Batch Normalization model architecture has more than 134M parameters and more than 120M trainable parameters. Batch size is determined for this model architecture as 128 and a total of 25 epochs are iterated. The evaluation interval is chosen to be 25 as the same as the epoch number to evaluate the model at the final step, after all the training processes.

3.6. Creating Features for Recognition

There are total of 3 different features which are used throughout the study. These are (i) labels of the cities that are encoded like attributes in similar studies, (ii) 3D color histograms, and (iii) GIST descriptors. Each of these features serves a different purpose.

Labels are created for each image and each city class using Imagga Tagging API [77]. A total of 5 labels are created for each image and a maximum of 20 labels are stored. Labels are sorted from most repetitive to least repetitive ones.

Figure 3.9. Word cloud of the labels used to represent cities

After having 20 labels for each city class, the final labels to represent all city classes are created by again sorting the labels and applying some filtering. After all these operations, a total of 134 labels are created and stored. These labels can be found in Figure 3.9 as a word cloud in the shape of the world. The illustration is created using WordArt.com [78].

Initially, at the start of this study, dominant colors of the images are planned to be extracted. Domain colors are extracted by K-means method by obtaining centroids from that

method. Also, RGB colors are converted into CIELAB color space in K-means method because of having more natural and realistic colors than RGB for the representation of the colors and after finding centroids, conversion to RGB is made to be able to show the colors. The dominant color extraction process consists of obtaining dominant final RGB color components after converting to CIELAB and vice versa, getting their closest name using both CSS3 names and Name that Color library by C. Mehta [79], repeating these steps for each image and storing score values for the colors to obtain most repetitive N colors for a single city class. Mapping to names is also important for grouping similar colors. Then, after some experiments, it is seen that dominant colors are not strong enough to represent cities, and differentiating cities by this way is not possible, 3D color histograms are selected to be used as an auxiliary feature to help the recognition process.

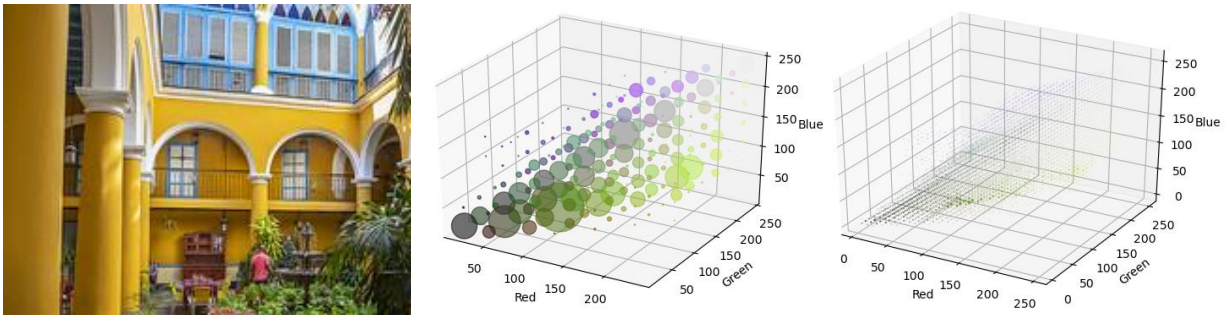

Figure 3.10. 3D color histogram visualizations of a Havana city image

3D color histogram visualizations can be found in Figure 3.10. The first 3D histogram is created with 8 bins while the other is created with 32 bins. The bin count represents the number of pixels within a range by grouping pixels [80]. In this study, bin count is chosen as 32 to group pixels from the image data 224x224 in size. The number 32 for the bin count is not high and also not low for the current configuration. In a similar study by Jain and Vailaya [81], bin count is chosen as 16 for each separate RGB channel of the images which are approximately 200x200 in size. The histogram created with 8 bins is given for illustration purposes. The shapes are more significant in the 8 bins histogram than the 32 bins histogram. After extracting 3D color histogram features for each image which are 32x32x32 in size, features are encoded in a single line by flattening them as 32,768 in size and stored in corresponding files.

GIST features are computed by A. Oliva and A. Torralba's GIST descriptor implementation [2]. The number of blocks is determined as 4 and the number of scales is 4

because orientations per scale are determined as 8, 8, 8, 8. With these configurations, a total of 1536 values (Calculation is shown in Equation 3.1) are computed and stored. The result is also multiplied by 3 because there are 3 channels in the images.

$$C = 3 * \left(\sum_{i}^{n\_scale} nblocks * nblocks * orientations\_per\_scale_i \right) \tag{3.1}$$

Visualizations of GIST descriptors can be found in Figure 3.11. GIST descriptor in the middle is generated as converting RGB image to grayscale image. Normally, the original GIST descriptor script by A. Oliva and A. Torralba [2] converts the color image to a grayscale image and then shows the descriptor. By modifying the original implementation and adapting it to color images, 3D color GIST descriptor is given at the rightmost of Figure 3.11.

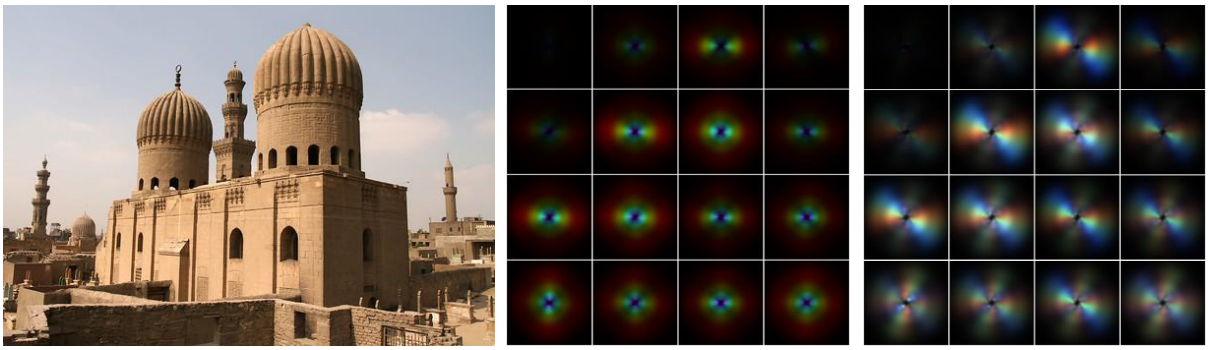

Figure 3.11. Visualizations of GIST descriptors

After extracting these 2 auxiliary features, they are filtered to eliminate the most similar ones and stored in the files to gain performance at the recognition process. Because each of these features focuses on different parts, each of them has different benefits at test time.

Also, some threshold values are determined at the test time at the comparison stage. Since distance calculation of each feature is done by different distance methods, their threshold values and their similarity values are different. Threshold values are determined by regarding minimum, maximum and arithmetic mean values of the distance values which are obtained after iterating on the dataset. Features having distance values smaller than these thresholds are processed.

3.7. Training Methods

Mini-batch gradient descent technique is applied at the training stage by determining the batch size as 128. The parameters are updated after each image data group which is called mini-batch that consists of 128 images. The batch size is selected as 128 because of speeding up the process while still preserving the advantages of the mini-batch gradient descent technique. In the study by Bengio [82, pp. 9], the author explains the selection of batch size as following words:

"The mini-batch size (B in Eq. (1)) is typically chosen between 1 and a few hundreds, e.g. B = 32 is a good default value, with values above 10 taking advantage of the speed-up of matrix-matrix products over matrix-vector products."

Algorithm 1: Traditional training algorithm

```
batchSize ← 128
// Zero-Shot Learning: False, Label Based Classification: False
Initialize the training dataset // Training set includes all the classes that test
    set includes
Initialize the model // ex. VGG-16 with Batch Normalization (LogSoftMax is
    applied in the last layer)
Initialize the optimizer // ex. AdaGrad
for e ← 1 to epochNumber do
    Set totalImageCount to total image count of training set
    stepSize ← ceil(totalImageCount\batchSize)
    for i ← 1 to stepSize do
        Set batchImages to batch of images
        Set batchGroundTruthIndexes to the original image indexes
        batchOutputs ← model(batchImages) // Get results of model by feeding
            images into the model
        Calculate Negative Log-Likelihood loss by comparing batchGroundTruthIndexes with
         batchOutputs
        Update model parameters by backpropagation
    end
end
```

Figure 3.12. Pseudocode of traditional training process

Pseudocodes of the different training processes can be found in Figure 3.12 and 3.13. While problem setups are different, the implementations are similar. Mainly, loss functions and model outputs are different in these different problem setups. Also, label-based classification indicates that the vector in length 134 is used similar to the attribute vector in similar studies. The label set can be found in Figure 3.9 as an illustration.

Algorithm 2: Training algorithm using Zero-shot learning

```
batchSize ← 128
// Zero-Shot Learning: True, Label Based Classification: True
Initialize the training dataset // Training set includes classes that validation
    and test sets do not
Initialize the model // ex. VGG-16 with Batch Normalization
Initialize the optimizer // ex. AdaGrad
for e ← 1 to epochNumber do
    Set totalImageCount to total image count of training set
    stepSize ← ceil(totalImageCount\batchSize)
    for i ← 1 to stepSize do
        Set batchImages to batch of images
        Set batchGroundTruthLabels to the original image labels // Size of one label
            vector: 1x134
        batchOutputs ← model(batchImages) // Get results of model by feeding
            images into the model
        batchSigmoidOutputs ← sigmoid(batchOutputs)
        Calculate Binary Cross-Entropy loss by comparing batchGroundTruthLabels with
         batchSigmoidOutputs
        Update model parameters by backpropagation
    end
end
```

Figure 3.13. Pseudocode of ZSL training process

If TZSL is applied at this, training stage, then there will be images of the test classes merged into the training set without their related labels or any information. The predicted labels from the target classes and the ground truth labels from source classes can be used for the loss function to backpropagate and update the parameters of the model. The implementations of all of these are straightforward at the training stage. The aim is to show images to the model as seen classes and to either make a prediction on seen classes again as it is in traditional training configuration or to make a prediction on unseen classes which are in the disjoint test set as it is in ZSL configuration. Also, as another option, a prediction can be made with a combined set that includes both seen and unseen classes at test time.

Algorithm 3: Training algorithm using Transductive Zero-shot learning

$batchSize \leftarrow 128$
// Zero-Shot Learning: True, Label Based Classification: True
Initialize the training dataset // Training set includes classes that validation and test sets do not
Merge test dataset into the training dataset Initialize the model // ex. VGG-16 with Batch Normalization
Initialize the optimizer // ex. AdaGrad
for $e \leftarrow 1$ ***to*** $epochNumber$ **do**
    Set $totalImageCount$ to total image count of training set
    $stepSize \leftarrow ceil(totalImageCount \backslash batchSize)$
    for $i \leftarrow 1$ ***to*** $stepSize$ **do**
        Set $batchSourceImages$ to batch of images of the training set
        Set $batchTargetImages$ to batch of images of the test set
        Set $batchGroundTruthLabels$ to the original image labels of the training set // Size of one label vector: 1x134
        $batchSourceOutputs \leftarrow model(batchSourceImages)$ // Get results of model by feeding source images into the model
        $batchTargetOutputs \leftarrow model(batchTargetImages)$ // Get results of model by feeding target images into the model
        $batchSourceSigmoidOutputs \leftarrow sigmoid(batchSourceOutputs)$
        $batchTargetSigmoidOutputs \leftarrow sigmoid(batchTargetOutputs)$
        Set $batchPredictedLabels$ to the predicted image labels from $batchTargetSigmoidOutputs$ by using label, 3D color histogram and GIST descriptor features
        Calculate Binary Cross-Entropy loss by comparing $batchGroundTruthLabels$ with $batchSourceSigmoidOutputs$ and $batchPredictedLabels$ with $batchTargetSigmoidOutputs$
        Update model parameters by backpropagation
    end
end

Figure 3.14. Pseudocode of TZSL training process

The TZSL approach can be found as pseudocode in Figure 3.14. The source classes are seen classes from the training set while the target classes are unseen from the test set. The prediction part is identical to its counterpart given in Figure 3.19. Three different distance methods are used to predict the label vector which is in 1x134 size.

Pseudocodes in these figures are created using LaTeX document preparation system and a LaTeX template which is written by S. A. Rather [83] is used. Also, other pseudocodes in the following chapters are created in this way.

The dataset augmentation techniques which are random resized cropping, color jittering, random rotation, and horizontal flip are applied to the images at the training time. Also, when a pre-trained model is used and if this pre-trained model is trained on ImageNet dataset earlier, then the images are normalized by mean and standard deviation values which are 0.485, 0.456, 0.406, and 0.229, 0.224, 0.225 respectively.

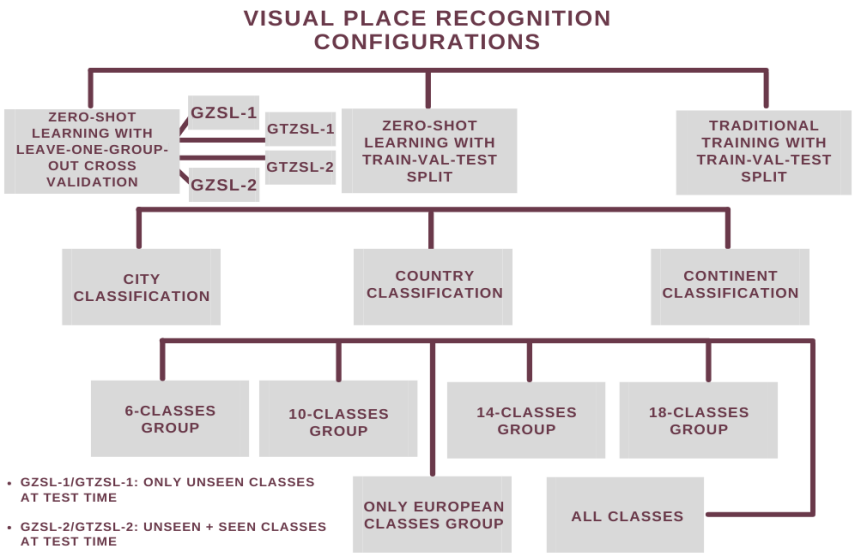

Figure 3.15. Summary of the experimented different techniques throughout the study

Since two different GZSL approach is applied in this thesis, if the first GZSL approach is applied, the whole training set is provided to the model at the training time while if the second GZSL approach is applied which is the original GZSL approach, only the half of the training set is provided to the model at the training time because providing the other half of the training set at the test time. The only difference between these two GZSL techniques is including half of the training classes by merging them into the test set. The classes of these training data are seen but the image data are unseen because the model is trained with only half of the original training set at the training stage while the other half is used at the test stage.

All of these different techniques are applied to observe the results when the configurations are changed. To summarize these different techniques, an illustration that is created using Canva.com [84] is given in Figure 3.15.

3.8. Validation Methods

3.8.1. Train-Validation-Test Split

Train-validation-test split is the common split technique that is used for many similar studies. The validation set is useful when there needs to be hyperparameter

optimizing and by observing the loss of the validation set, some changes can be made on the model or dataset to increase the accuracy of the to-be-trained model.

Algorithm 4: Train-validation-test split validation algorithm

```
batchSize ← 128
Initialize the validation dataset // If ZSL approach is followed then disjoint set
    is used, else subset of the training set is used
Initialize the model // ex.  VGG-16 with Batch Normalization (If mode is not
    ZSL, then LogSoftMax is applied in the last layer)
Initialize the optimizer // ex.  AdaGrad
for e ← 1 to epochNumber do
    Set totalImageCount to total image count of training set
    stepSize ← ceil(totalImageCount\batchSize)
    for i ← 1 to stepSize do
        Set batchImages to batch of images
        if mode =' ZSL' then
            Set batchGroundTruthLabels to the original image labels // Size of one label
                vector:  1x134
        else
            Set batchGroundTruthIndexes to the original image indexes
        end
        batchOutputs ← model(batchImages) // Get results of model by feeding
            images into the model
        if mode =' ZSL' then
            batchSigmoidOutputs ← sigmoid(batchOutputs)
            Calculate Binary Cross-Entropy loss by comparing batchGroundTruthLabels with
              batchSigmoidOutputs
        else
            Calculate Negative Log-Likelihood loss by comparing batchGroundTruthIndexes with
              batchOutputs
        end
        Observe the loss // Do not backpropagate
    end
end
```

Figure 3.16. Pseudocode of validation process using train-validation-test split

The sample pseudocode for this technique is given in Figure 3.16. Also, one should note that image data in the validation sets are different from the other sets. If the same images which are fed into the model at the training time or queried at the test time are used for at validation time, then hyperparameter optimization would not be possible because of not obtaining correct and realistic results.

3.8.2. Leave-one-group-out Cross-Validation Split

This split is a type of cross-validation split. Basically, one group which is the city class in this context is left for the testing stage and all the other classes are included at the training stage.

Algorithm 5: Leave-one-group-out cross validation algorithm

```
batchSize ← 128
// There is no separate validation set in this configuration
Initialize the training and test set // ex. If there exists 6 classes, then there
    will be 5 classes in the training set while there will be 1 class
    in the test set
Initialize the model // ex. VGG-16 with Batch Normalization (If mode is not
    ZSL, then LogSoftMax is applied in the last layer)
Initialize the optimizer // ex. AdaGrad
for k ← 1 to maxK do
    for e ← 1 to epochNumber do
        Set totalImageCount to total image count of training set
        stepSize ← ceil(totalImageCount\batchSize)
        for i ← 1 to stepSize do
            Set batchImages to batch of images
            if mode =' ZSL' then
                Set batchGroundTruthLabels to the original image labels // Size of one
                    label vector: 1x134
            else
                Set batchGroundTruthIndexes to the original image indexes
            end
            batchOutputs ← model(batchImages) // Get results of model by feeding
                images into the model
            if mode =' ZSL' then
                batchSigmoidOutputs ← sigmoid(batchOutputs)
                Calculate Binary Cross-Entropy loss by comparing batchGroundTruthLabels with
                 batchSigmoidOutputs
            else
                Calculate Negative Log-Likelihood loss by comparing batchGroundTruthIndexes with
                 batchOutputs
            end
            Update model parameters by backpropagation
        end
    end
    Evaluate the model with test set
    Reinitialize the training and test set
    Reinitialize the model
    Reinitialize the optimizer
end
```

Figure 3.17. Pseudocode of validation process using cross-validation split

In this study, there is no specific validation set configured for this split. If it is needed, one more class can be left as it is used in the validation set and all the other classes (which are actually k-2 classes) can be added to the training set. Because each step train and test set are changed, the separate validation set is not used in this configuration. The sample pseudocode for this split can be found in Figure 3.17.

This split is useful to detect which classes the model recognizes the best and which classes do not. The model is recreated after all the epochs and all the evaluations are done after the epoch index reaches its maximum value.

There are two different training modes which are ZSL and traditional learning. Different loss functions are used when the training mode is changed.

3.9. Testing Methods

At the test stage, the labels and the extra 2 auxiliary features are compared by distance methods. There are total of 3 different distance methods used at this stage, which are Hamming, Euclidean, and Bhattacharyya. Sample implementations of 2 different configurations are given in Figure 3.18 and 3.19.

Algorithm 6: Traditional testing algorithm

```
batchSize ← 128
// Zero-Shot Learning: False, Label Based Classification: False,
   Mode: City, Country or Continent
Initialize the test set // ex. If there exists 6 classes in the training set,
   then there will be 6 classes in the test set too
Initialize the model // ex. VGG-16 with Batch Normalization (If mode is not
   ZSL, then LogSoftMax is applied in the last layer)
Initialize the optimizer // ex. AdaGrad
Set totalImageCount to total image count of training set
stepSize ← ceil(totalImageCount\batchSize)
for i ← 1 to stepSize do
    Set batchImages to batch of images
    Set batchGroundTruthIndexes to the original image indexes
    Set batchGroundTruthNames to the original image city, country or continent names
    batchOutputs ← model(batchImages) // Get results of model by feeding
       images into the model
    Calculate Negative Log-Likelihood loss by comparing batchGroundTruthIndexes with
     batchOutputs
    Observe the loss // Do not backpropagate
    // Begin evaluating the model with the batch of images from the
       test set
    Set topKClassIndexes to the predicted top-k class indexes from batchOutputs
    Set batchPredictedNames from topKClassIndexes by converting indexes to names
    Compare batchGroundTruthNames with batchPredictedNames and calculate accuracy
end
```

Figure 3.18. Pseudocode of traditional testing algorithm

The testing stage consists of making a label-based comparison and iterating over all the files of auxiliary features to compare with the queried image. This stage is the longest part in terms of computation time compared to all the other stages which are training and validation.

Algorithm 7: Testing algorithm using Zero-shot learning

$batchSize \leftarrow 128$
// Zero-Shot Learning: True, Label Based Classification: True, Mode: City, Country or Continent
Initialize the test set // ex. If testMode is 'Generalized', then half of the training set is merged into test set
Initialize the model // ex. VGG-16 with Batch Normalization (If mode is not ZSL, then LogSoftMax is applied in the last layer)
Initialize the optimizer // ex. AdaGrad
Set $totalImageCount$ to total image count of test set
$stepSize \leftarrow ceil(totalImageCount \backslash batchSize)$
for $i \leftarrow 1$ ***to*** $stepSize$ **do**
    Set $batchImages$ to batch of images
    Set $batchGroundTruthLabels$ to the original image labels // Size of one label vector: 1x134
    Set $batchGroundTruthNames$ to the original image city, country or continent names
    $batchOutputs \leftarrow model(batchImages)$ // Get results of model by feeding images into the model
    $batchSigmoidOutputs \leftarrow sigmoid(batchOutputs)$
    Calculate Binary Cross-Entropy loss by comparing $batchGroundTruthLabels$ with $batchSigmoidOutputs$
    Observe the loss // Do not backpropagate
    // Begin evaluating the model with the batch of images from the test set
    Threshold the values of $batchSigmoidOutputs$ to 0 and 1 as binary values // Values smaller than 0.5 are assigned to 0 while remaning values are asssigned to 1
    if $testMode =' Generalized'$ **then**
        Compare $batchGroundTruthLabels$ with the labels of all the classes by Hamming distance method
        Get 3D color histogram features of $batchImages$ and compare them with the 3D color histogram features of all the classes by Bhattacharyya distance method // Size of one color histogram vector: 1x32768
        Get GIST features of $batchImages$ and compare them with the GIST features of all the classes by Euclidean distance method // Size of one GIST vector: 1x1536
    else
        Compare $batchGroundTruthLabels$ with the labels of only the test classes by Hamming distance method
        Get 3D color histogram features of $batchImages$ and compare them with the 3D color histogram features of only the test classes by Bhattacharyya distance method // Size of one color histogram vector: 1x32768
        Get GIST features of $batchImages$ and compare them with the GIST features of only the test classes by Euclidean distance method // Size of one GIST vector: 1x1536
    end
    Store the city indexes having smallest distances by earlier comparisons
    Set $batchPredictedNames$ to city, country or continent names based on the mode by converting city indexes to names
    Compare $batchGroundTruthNames$ with $batchPredictedNames$ and calculate accuracy
end

Figure 3.19. Pseudocode of ZSL testing algorithm

Pseudocodes in these figures are given as a sample of the separate processes. Because these separate processes share some common parts, some of these parts are not shown and only

the important parts of each process are given. Lastly, in Figure 3.19, whole testing methods can be seen as they are the main tasks of this study.

Because each distance method outputs different values at the comparison stage, some additional threshold values are used to capture only the most similar cities, countries, or continents while comparing features with these distance methods. Hamming and Bhattacharyya results are scaled into a range of [0, 1] while there is no fixed range for Euclidean results. Initially, all the comparisons and processes are done on cities as city indexes or names and these indexes or names are converted to country or continent names for the comparison process later.

In summary, testing methods include comparisons of both label and auxiliary features by different distance methods. After all these operations, cities are predicted regarding their scores.

Top-k values are determined as 1, 2, 3, and 5. Each of these values is important to make decisions about the success of the model. Top-1 is straightforward while the other top-k values are calculated by predicting more classes for the single prediction.

Performance of the test stage is increased by comparing feature vectors with the stored feature files in parallel. Also, distance methods used in this stage are optimized.

3.10. Final Enhancements

After all these stages and operations, some enhancements like selecting subsets (ex. 6, 10, 14, 18 classes as subsets) of the original dataset, splitting cities into disjoint sets considering their country or continent, determining 1,2,3 and 5 for top accuracy values, considering original GZSL approach which is including images of both training classes as seen classes and test classes as unseen classes with all of the labels at the test time instead of providing only test classes with all of the labels, modifying scripts to work on external pre-trained models like VGG16 model pre-trained on Places365 dataset [69], configurations on layers of the models are done.

Also, some performance enhancements are done for top-k calculations and for result-gathering processes. To fasten the processes, some operations are converted into multi-thread tasks and some optimizations are done on array operations and distance methods. After all these enhancements and experiments, the final results are demonstrated in the next chapter.

4. RESULTS AND DISCUSSION

4.1. Results of Zero-Shot Learning Approach

All the results related to ZSL training, validation, and testing process are given in this chapter. Results of each different configuration are discussed and their comparisons and illustrations are shown in detail.

Table 4.1. Zero-shot learning results of city classification having only test classes at test time

Class Size (Cities)	Method	Training Set Size (Classes)	Validation Set Size (Classes)	Test Set Size (Classes)	Mean Target Accuracy (Cities)
6	GZSL-1 (Cross Val)	5	-	1	Top-1: 0.146 Top-2: 0.405 Top-3: 0.515 Top-5: 0.681
10	GZSL-1 (Cross Val)	9	-	1	Top-1: 0.182 Top-2: 0.286 Top-3: 0.373 Top-5: 0.533
14	GZSL-1 (Cross Val)	13	-	1	Top-1: 0.161 Top-2: 0.219 Top-3: 0.283 Top-5: 0.421
18	GZSL-1 (Cross Val)	17	-	1	Top-1: 0.133 Top-2: 0.183 Top-3: 0.229 Top-5: 0.341

In Table 4.1 and 4.2, city classification is done and leave-one-group-out cross-validation technique is used in the Zero-shot learning setup. Unseen accuracy values are related to the test classes while seen and unseen accuracy values together are related to both train and test classes. The arithmetic mean values of these accuracy scores are given in the tables. Arithmetic mean top-k accuracy results are calculated at the end of all steps. In leave-one-group-out cross-validation configurations, only one class is provided at test time while other remaining classes are provided at the training time and the model and the optimizer are recreated at each step.

In most of the tables, results of two different generalized ZSL configurations which are labeled by GZSL-1 and GZSL-2, are given. The difference between these two configurations are including seen classes at test time or not. In GZSL-2, training classes are provided at test time while in GZSL-1, which is the alternative generalized approach in this study, they are not and only test classes are provided regarding all labels of all classes.

Top-k values are determined as 1, 2, 3, and 5 and shown in all the tables separately. They are calculated by taking average of all results. Mainly, there are 5 different configurations which are ZSL, GZSL-1, GZSL-2, GTZSL-1, and GTZSL-2. GTZSL-1 and GTZSL-2 are similar to GZSL-1 and GZSL-2 respectively because of applying a generalized approach (including only unseen classes at test time or including both seen and unseen classes at test time). GTZSL-1 is the transductive version of GZSL-1, while GTZSL-2 is the transductive version of GZSL-2.

Table 4.2. Zero-shot learning results of city classification having both seen and unseen classes at test time

Class Size (Cities)	**Method**	**Training Set Size (Classes)**	**Validation Set Size (Classes)**	**Test Set Size (Classes)**	**Mean Target Accuracy (Cities)**
6	GZSL-2 (Cross Val)	5	-	1 + 5	Top-1: 0.340 Top-2: 0.552 Top-3: 0.670 Top-5: 0.815
10	GZSL-2 (Cross Val)	9	-	1 + 9	Top-1: 0.301 Top-2: 0.430 Top-3: 0.520 Top-5: 0.666
14	GZSL-2 (Cross Val)	13	-	1 + 13	Top-1: 0.263 Top-2: 0.352 Top-3: 0.425 Top-5: 0.553
18	GZSL-2 (Cross Val)	17	-	1 + 17	Top-1: 0.210 Top-2: 0.288 Top-3: 0.350 Top-5: 0.462

In Table 4.3, 4.4, 4.5, and 4.6, country and continent versions of Table 4.1 and 4.2 are given. Country and continent accuracy values are calculated by mapping city classes into their related country and continent labels.

Table 4.3. Zero-shot learning results of country classification having only unseen classes at test time

Class Size (Cities)	Method	Training Set Size (Classes)	Validation Set Size (Classes)	Test Set Size (Classes)	Mean Target Accuracy (Countries)
6	GZSL-1 (Cross Val)	5	-	1	Top-1: 0.415 Top-2: 0.622 Top-3: 0.763 Top-5: 0.856
10	GZSL-1 (Cross Val)	9	-	1	Top-1: 0.314 Top-2: 0.444 Top-3: 0.564 Top-5: 0.738
14	GZSL-1 (Cross Val)	13	-	1	Top-1: 0.248 Top-2: 0.354 Top-3: 0.463 Top-5: 0.650
18	GZSL-1 (Cross Val)	17	-	1	Top-1: 0.240 Top-2: 0.346 Top-3: 0.448 Top-5: 0.601

Table 4.4. Zero-shot learning results of country classification having both seen and unseen classes at test time

Class Size (Cities)	Method	Training Set Size (Classes)	Validation Set Size (Classes)	Test Set Size (Classes)	Mean Target Accuracy (Countries)
6	GZSL-2 (Cross Val)	5	-	1 + 5	Top-1: 0.509 Top-2: 0.679 Top-3: 0.808 Top-5: 0.901
10	GZSL-2 (Cross Val)	9	-	1 + 9	Top-1: 0.438 Top-2: 0.564 Top-3: 0.677 Top-5: 0.823
14	GZSL-2 (Cross Val)	13	-	1 + 13	Top-1: 0.346 Top-2: 0.464 Top-3: 0.561 Top-5: 0.712

18	GZSL-2 (Cross Val)	17	-	1 + 17	Top-1: 0.310 Top-2: 0.426 Top-3: 0.519 Top-5: 0.650

Table 4.5. Zero-shot learning results of continent classification having only unseen classes at test time

Class Size (Cities)	**Method**	**Training Set Size (Classes)**	**Validation Set Size (Classes)**	**Test Set Size (Classes)**	**Mean Target Accuracy (Continents)**
6	GZSL-1 (Cross Val)	5	-	1	Top-1: 0.607 Top-2: 0.794 Top-3: 0.875 Top-5: 0.912
10	GZSL-1 (Cross Val)	9	-	1	Top-1: 0.387 Top-2: 0.534 Top-3: 0.663 Top-5: 0.817
14	GZSL-1 (Cross Val)	13	-	1	Top-1: 0.447 Top-2: 0.595 Top-3: 0.709 Top-5: 0.828
18	GZSL-1 (Cross Val)	17	-	1	Top-1: 0.493 Top-2: 0.641 Top-3: 0.736 Top-5: 0.822

Table 4.6. Zero-shot learning results of country classification having both seen and unseen classes at test time

Class Size (Cities)	**Method**	**Training Set Size (Classes)**	**Validation Set Size (Classes)**	**Test Set Size (Classes)**	**Mean Target Accuracy (Continents)**
6	GZSL-2 (Cross Val)	5	-	1 + 5	Top-1: 0.693 Top-2: 0.833 Top-3: 0.909 Top-5: 0.944
10	GZSL-2 (Cross Val)	9	-	1 + 9	Top-1: 0.503 Top-2: 0.645 Top-3: 0.758 Top-5: 0.882

14	GZSL-2 (Cross Val)	13	-	1 + 13	Top-1: 0.514 Top-2: 0.657 Top-3: 0.752 Top-5: 0.853
18	GZSL-2 (Cross Val)	17	-	1 + 17	Top-1: 0.546 Top-2: 0.684 Top-3: 0.768 Top-5: 0.850

In Table 4.7, the effect of enabling and disabling auxiliary features which are 3D color histogram features and GIST descriptor features are observed.

Table 4.7. Zero-shot learning results of city classification with or without enabling auxiliary features at test time

Class Size (Cities)	**Method**	**Training Set Size (Classes)**	**Validation Set Size (Classes)**	**Test Set Size (Classes)**	**Mean Target Accuracy (Cities)**
6	GZSL-1 (Cross Val) (Auxiliary features are enabled)	5	-	1	Top-1: 0.146 Top-2: 0.405 Top-3: 0.515 Top-5: 0.681
6	GZSL-2 (Cross Val) (Auxiliary features are enabled)	5	-	1 + 5	Top-1: 0.340 Top-2: 0.552 Top-3: 0.670 Top-5: 0.815
6	GZSL-1 (Cross Val) (Auxiliary features are disabled)	5	-	1	Top-1: 0.000 Top-2: 0.249 Top-3: 0.280 Top-5: 0.307
6	GZSL-2 (Cross Val) (Auxiliary features are disabled)	5	-	1 + 5	Top-1: 0.402 Top-2: 0.563 Top-3: 0.611 Top-5: 0.630

Even the mean accuracy values are high in GZSL-2 configuration with disabling the auxiliary features, it is observed that the prediction accuracy of the unseen classes to be predicted

at each stage is low. Also, the reason for this is due to computing the arithmetic mean of accuracies.

Table 4.8. Zero-shot learning results of city classification having only European classes

Class Size (Cities)	Method	Training Set Size (Classes)	Validation Set Size (Classes)	Test Set Size (Classes)	Target Accuracy (Cities)
23	ZSL (Train-Val-Test Split)	13	4	6	Top-1: 0.220 Top-2: 0.405 Top-3: 0.576 Top-5: 0.831
23	GZSL-1 (Train-Val-Test Split)	13	4	6	Top-1: 0.067 Top-2: 0.108 Top-3: 0.141 Top-5: 0.229
23	GZSL-2 (Train-Val-Test Split)	13	4	6 + 13	Top-1: 0.156 Top-2: 0.213 Top-3: 0.254 Top-5: 0.342

Table 4.9. Zero-shot learning results of city classification having all 42 classes

Class Size (Cities)	Method	Training Set Size (Classes)	Validation Set Size (Classes)	Test Set Size (Classes)	Target Accuracy (Cities)
42	ZSL (Train-Val-Test Split)	21	9	12	Top-1: 0.185 Top-2: 0.264 Top-3: 0.353 Top-5: 0.524
42	GZSL-1 (Train-Val-Test Split)	21	9	12	Top-1: 0.119 Top-2: 0.145 Top-3: 0.166 Top-5: 0.206
42	GZSL-2 (Train-Val-Test Split)	21	9	12 + 21	Top-1: 0.144 Top-2: 0.180 Top-3: 0.207 Top-5: 0.255

In summary, in ZSL configuration, labels and image data of only unseen classes are provided, while in GZSL-1 configuration, labels of all classes and image data of only unseen classes are provided and in GZSL-2 configuration, labels and image data of all classes are provided.

Table 4.10. Transductive Zero-shot learning results of city classification having 6 classes

Class Size (Cities)	Method	Training Set Size (Classes)	Validation Set Size (Classes)	Test Set Size (Classes)	Mean Target Accuracy (Cities)
6	GTZSL-1 (Cross Val)	5	-	1	Top-1: 0.152 Top-2: 0.408 Top-3: 0.520 Top-5: 0.687
6	GTZSL-2 (Cross Val)	5	-	1 + 5	Top-1: 0.350 Top-2: 0.564 Top-3: 0.688 Top-5: 0.840

In the first generalized configurations (GZSL-1, GTZSL-1) which are denoted by 1, because the unseen set includes only one class, the accuracy values of inductive ZSL (standard ZSL) are not given as they are simply 1.0.

In both GTZSL-2 and GZSL-2 configurations, split is done on training and test sets and they are split into two halves. In GZSL-2 configuration, the training set is split into two halves while in GTZSL-2 configuration, both training and test sets are split into two halves. One part of the test set is used at the training stage while the other part is used at the evaluation stage [30].

In conclusion, after examining all the results, it is seen that GZSL-2 and similarly GTZSL-2 are the best performer ones in terms of their recognition rates when the target set size is not very high. When the target set size is high, inductive ZSL yields better results than GZSL-2. GZSL-1 performs poorer compared to inductive ZSL because of taking all labels into consideration of both seen and unseen classes at the test time. Also, it is observed that when the size of processed sets increased, there may be a drop in the accuracy values. GTZSL-1 and GTZSL-2 are similar to GZSL-1 and GZSL-2 respectively. Because some of the city classes

share the same country and continent, there are few classes in the country and continent sets compared to city sets. Thus, country and continent classification performs better than city classification. Also, sometimes predicting cities might be difficult due to complex scenes in images but country and especially continent classification is easier to make because of the learned similar shapes and features from different cities in the same country or the same continent. Furthermore, some of the results are not simply 1.0 even the top-k number is equal to or bigger than the target set size. This is due to selecting targets based on the threshold values.

4.2. Comparison of Different Problem Setups and Approaches

In this section, comparisons are mainly focused on ZSL and traditional learning processes. Results of these two different problem setups are given in the next tables. Also, each of these different setups has its own characteristics, advantages, and also disadvantages.

In Table 4.11 and 4.12, two different configurations are tested which are including only European classes and all the classes respectively. Results are obtained as top-k values and k values are determined as 1, 2, 3, and 5 again.

In the non-ZSL, traditional training setup, the classes are shared by different sets but image data are split into these sets which are train, validation, and test separately. Same image data is included in only one set and not shared with the others.

Training, validation, and test set split is used in the traditional training stage and split percentages are determined as 70%, 15%, 15% for training, validation, and test sets respectively when all classes are included. When only European classes are included, these percentages are changed as 60%, 20%, and 20% respectively.

Table 4.11. Traditional training results of city classification having only European classes

Class Size (Cities)	Method	Training Set Size (Classes)	Validation Set Size (Classes)	Test Set Size (Classes)	Target Accuracy (Cities)
23	Traditional Training (Train-Val-Test Split)	23	23	23	Top-1: 0.313 Top-2: 0.427 Top-3: 0.503 Top-5: 0.614

Table 4.12. Traditional training results of city classification having all the classes

Class Size (Cities)	Method	Training Set Size (Classes)	Validation Set Size (Classes)	Test Set Size (Classes)	Target Accuracy (Cities)
42	Traditional Training (Train-Val-Test Split)	42	42	42	Top-1: 0.276 Top-2: 0.369 Top-3: 0.435 Top-5: 0.532

Table 4.13. Traditional training results of city classification having different sizes of classes

Class Size (Cities)	Method	Training Set Size (Classes)	Validation Set Size (Classes)	Test Set Size (Classes)	Target Accuracy (Cities)
6	Traditional Training (Train-Val-Test Split)	6	6	6	Top-1: 0.600 Top-2: 0.806 Top-3: 0.887 Top-5: 0.980
10	Traditional Training (Train-Val-Test Split)	10	10	10	Top-1: 0.460 Top-2: 0.630 Top-3: 0.726 Top-5: 0.855
14	Traditional Training (Train-Val-Test Split)	14	14	14	Top-1: 0.458 Top-2: 0.607 Top-3: 0.693 Top-5: 0.793
18	Traditional Training (Train-Val-Test Split)	18	18	18	Top-1: 0.403 Top-2: 0.519 Top-3: 0.584 Top-5: 0.703

In Table 4.13, the results of the different subsets from the original dataset are given. It is seen from the traditional learning results that the size of the sets affects the accuracy values as in the other configurations.

Cross-validation split is used only in the ZSL configuration, thus only training-validation-test split is applied in the traditional learning configuration.

4.3. Limits and Shortcomings of the Thesis

As every study has some limits and shortcomings, there are some limits and shortcomings in this thesis, as well. Visual place recognition task is a challenging topic and using Zero-shot learning for this task also introduces some challenging issues.

One of them is the image gathering process which relies on various sources like Flickr or external datasets. While the gathering process involves downloading images based on their latitude, longitude coordinates, titles, descriptions, tags, there is still a need to obtain the images from more trustworthy sources. Even there are many map applications like Google Maps, Mapillary to solve this problem, they are still not the best source for this city recognition task. While some popular landmarks of various cities can be obtained in these ways, the original task is to predict the city independent of their popular structures. Creating a dataset by only cherry-picked images of the cities is not applicable and the need for random and non-cherry-picked images exists. In this study, there are both some cherry-picked images from the external dataset and random images from Flickr based on some fixed radius and GPS coordinates.

Because visual place recognition mainly depends on the specific datasets, the dataset which is used throughout the study is important. Most of the similar studies in the literature have created their own and unique dataset because of their tasks and goals. Samples are the most important part of any recognition task and sometimes there is no available dataset for the task to be worked on.

The other limit is the computing resources that are used to train and test the model. Due to the fact that deep learning architectures usually require significant computing power, in this thesis, up to 25 epochs are tested and some issues are taken into consideration to increase the speed and performance of both the training and test stage. In this thesis, NVIDIA GTX 1080 TI is mainly used for all the computations and processes. Also, the batch size is determined as 128, and results are obtained after all the epochs are done.

There are no or not enough studies for visual place recognition using Zero-shot learning techniques. Zero-shot learning is considered a new topic, at least compared to traditional learning techniques.

While predicting a city is possible by learning features like architectural structures at earlier steps from source, seen classes, some cities may have similar architectures or objects.

This issue may result in false predictions. To overcome this problem, it is good to have distinct or nearly distinct images in the dataset for different cities to be able to differentiate cities.

4.4. Discussion and Possible Improvements for Future

Even various approaches are followed and various methods are applied, of course, there is still a need for some improvements. Even the Zero-shot learning technique does not require a huge dataset by its nature, additional images may be provided to strengthen the model which works on only seen classes in the ZSL technique.

In addition to 3D color histogram features and GIST descriptors, additional auxiliary features may be provided at the test time. Auxiliary features are basically helpers for the model to avoid wrong predictions. While the extraction process of auxiliary features is not straightforward, there are still some issues that are needed to be taken into consideration. Also, since creating and training a model from scratch is not easy, pre-trained models are used to apply transfer learning like freezing all the layers except the last layer and use the model as a feature extractor. In this study, mainly the pre-trained VGG16 model with Batch Normalization is used and this model is trained on ImageNet dataset earlier. There are various studies on training and testing on ImageNet dataset like the study of Krizhevsky et al. [85]. Also, in addition to the pre-trained on ImageNet dataset models, there are more specific pre-trained models like trained on visual places datasets. These pre-trained models may be used to increase the success of the model. Furthermore, there are various techniques of extracting image descriptors as they are discussed by Zeng et al. [86].

5. CONCLUSION

Predicting visual places consists of many steps and there are different approaches to follow to achieve this goal. While images related to any visual place can have many different objects in them, CNN architecture is capable of focusing and capturing the important features from these images. Also, repetitive structures are an important factor in the learning process. There are also different architectures like LSTMs which are a subset of RNN architectures. Because CNN architectures are powerful and mostly used for the tasks like this study, they are used throughout this study.

Various distance methods are used throughout this study for both label-based comparisons and comparisons of auxiliary features. A label-based approach is followed to assign various tags to cities regarding their frequency of occurrence. 3D color histogram and GIST descriptors are extracted and filtered from the images of the dataset and they are used at the test stage to increase the accuracy and provide further information about the classes.

Recognition of visual places is done on three different levels which are city, country, and continent. After making city predictions, comparisons of countries and continents are made and accuracy values are calculated regarding these 3 different levels. Also, different subsets are created from the original dataset to include different cities, countries, and continents. The choice of these classes is important for the recognition process. City classes are selected from the countries which have more than one city in the dataset. Like separating cities in the leave-one-group-out cross-validation method, one city class of a country is included in one of the sets while the other city class of the same country is moved to another set.

Some different validation techniques are applied which are basically train-validation-test split and leave-one-group-out cross-validation. While the train-validation-test split is mainly used in the traditional learning process, the leave-one-group-out cross-validation split is mainly used in the ZSL technique.

Zero-shot learning is an important technique, especially for this task, to predict unseen classes based on seen classes which are learned earlier. Some different versions of ZSL (ex. GZSL) are applied throughout this study and their results and comparisons are shown. Because GZSL is a more realistic approach, the main focus is given to this approach throughout the thesis.

In addition to ZSL problem setup, traditional learning is also implemented and experimented with. Comparisons on these 2 different techniques are made and results are shown. Because traditional learning does not have disjoint sets like in ZSL approach, training is done on all the classes while ZSL requires working on only target classes which are unseen ones and different from the source classes which are seen ones.

From the final results, it is seen that the generalized approach when both seen and unseen sets are included at test time performs better compared to the alternative generalized approach which consists of including only unseen classes with all labels of all classes. Because of the search space constraint, the alternative generalized approach does not yield the best results. Also, it is seen that the inductive ZSL which works on only unseen classes with their related labels performs better than the alternative generalized approach and two different transductive approaches perform similar to their related generalized approaches.

In conclusion, visual place recognition with Zero-shot learning is an open research topic and, it is believed that there may be more studies in this field in the future.

REFERENCES

[1] GeoGuessr.com. (2021). GeoGuessr AB Accessed: Aug. 07, 2021. [Online]. Available: https://www.geoguessr.com/

[2] A. Oliva and A. Torralba, "Modeling the shape of the scene: A holistic representation of the spatial envelope," *Int. J. Comput. Vis.*, vol. 42, no. 3, pp. 145–175, 2001, doi: 10.1023/A:1011139631724.

[3] A. Oliva and A. Torralba, "Chapter 2 Building the gist of a scene: the role of global image features in recognition," *Prog. Brain Res.*, vol. 155 B, pp. 23–36, 2006, doi: 10.1016/S0079-6123(06)55002-2.

[4] G. G. Marcu and S. Abe, "3D histogram visualization in different color spaces with application in color clustering classification," in *Device-Independent Color Imaging II*, Apr. 1995. doi: 10.1117/12.206542.

[5] J. Hays and A. A. Efros, "IM2GPS: Estimating geographic information from a single image," *26th IEEE Conf. Comput. Vis. Pattern Recognition, CVPR*, vol. 05, 2008, doi: 10.1109/CVPR.2008.4587784.

[6] T. Weyand, I. Kostrikov, and J. Philbin, "Planet - photo geolocation with convolutional neural networks," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 9912 LNCS, pp. 37–55, 2016, doi: 10.1007/978-3-319-46484-8_3.

[7] Y. Lecun *et al.*, "Handwritten Digit Recognition with a Back-Propagation Network," in *Advances in Neural Information Processing Systems 2*, 1990, pp. 396--404.

[8] K. Fukushima, "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biol. Cybern.*, vol. 36, no. 4, pp. 193–202, 1980, doi: 10.1007/BF00344251.

[9] J. Gu et al., “Recent advances in convolutional neural networks,” Pattern Recognition, vol. 77, pp. 354–377, May 2018, doi: 10.1016/j.patcog.2017.10.013.

[10] J. Kim, O. Sangjun, Y. Kim, and M. Lee, “Convolutional Neural Network with Biologically Inspired Retinal Structure,” *Procedia Comput. Sci.*, vol. 88, pp. 145–154, 2016, doi: 10.1016/j.procs.2016.07.418.

[11] R. Yamashita, M. Nishio, R. K. G. Do, and K. Togashi, “Convolutional neural networks: an overview and application in radiology,” *Insights Imaging*, vol. 9, no. 4, pp. 611–629, Jun. 2018, doi: 10.1007/s13244-018-0639-9.

[12] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the Inception Architecture for Computer Vision,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2016-Decem, pp. 2818–2826, 2016, doi: 10.1109/CVPR.2016.308.

[13] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2016-Decem, pp. 770–778, 2016, doi: 10.1109/CVPR.2016.90.

[14] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *3rd Int. Conf. Learn. Represent. ICLR 2015 - Conf. Track Proc.*, pp. 1–14, 2015.

[15] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. S. Bernstein et al., “Imagenet large scale visual recognition challenge,” International Journal of Com- puter Vision, vol. 115, no. 3, pp. 211–252, 2015.

[16] R. Caruana, D. L. Silver, J. Baxter, T. M. Mitchell, L. Y. Pratt, and S. Thrun, “Learning to learn: knowledge consolidation and transfer in inductive systems.” 1995.

[17] R. Vidal, J. Bruna, R. Giryes, and S. Soatto, “Mathematics of Deep Learning,” 2017, [Online]. Available: http://arxiv.org/abs/1712.04741.

[18] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, “Activation Functions: Comparison of trends in Practice and Research for Deep Learning,” pp. 1–20, 2018, [Online]. Available: http://arxiv.org/abs/1811.03378.

[19] W. Nash, T. Drummond, and N. Birbilis, “A review of deep learning in the study of materials degradation,” *npj Mater. Degrad.*, vol. 2, no. 1, pp. 1–12, 2018, doi: 10.1038/s41529-018-0058-x.

[20] Z. Chen, O. Lam, A. Jacobson, and M. Milford, “Convolutional neural network-based place recognition,” *Australas. Conf. Robot. Autom. ACRA*, vol. 02-04-December-2014, 2014.

[21] H. Larochelle, D. Erhan, and Y. Bengio, “Zero-data learning of new tasks,” *Proc. Natl. Conf. Artif. Intell.*, vol. 2, pp. 646–651, 2008.

[22] X. Song, H. Zeng, S. Zhang, L. Herranz, and S. Jiang, “Generalized Zero-shot Learning with Multi-source Semantic Embeddings for Scene Recognition,” *MM 2020 - Proc. 28th ACM Int. Conf. Multimed.*, pp. 3976–3985, 2020, doi: 10.1145/3394171.3413568.

[23] Y. Le Cacheux, H. Le Borgne, and M. Crucianu, “Zero-shot Learning with Deep Neural Networks for Object Recognition,” pp. 1–18, 2021, [Online]. Available: http://arxiv.org/abs/2102.03137.

[24] C. H. Lampert, H. Nickisch, and S. Harmeling, “Attribute-Based Classification for Zero-Shot Visual Object Categorization,” 2013.

[25] Z. Akata, F. Perronnin, Z. Harchaoui and C. Schmid, “Label-Embedding for Image Classification,” in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 7, pp. 1425-1438, 1 July 2016, doi: 10.1109/TPAMI.2015.2487986.

[26] W. L. Chao, S. Changpinyo, B. Gong, and F. Sha, “An empirical study and analysis of generalized zero-shot learning for object recognition in the wild,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 9906 LNCS, pp. 52–68, 2016, doi: 10.1007/978-3-319-46475-6_4.

[27] W. J. Scheirer, A. de Rezende Rocha, A. Sapkota and T. E. Boult, "Toward Open Set Recognition," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 7, pp. 1757-1772, July 2013, doi: 10.1109/TPAMI.2012.256.

[28] Y. Xian, C. H. Lampert, B. Schiele and Z. Akata, "Zero-Shot Learning—A Comprehensive Evaluation of the Good, the Bad and the Ugly," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 41, no. 9, pp. 2251-2265, 1 Sept. 2019, doi: 10.1109/TPAMI.2018.2857768.

[29] F. Pourpanah *et al.*, "A Review of Generalized Zero-Shot Learning Methods," pp. 1–24, 2020, [Online]. Available: http://arxiv.org/abs/2011.08641.

[30] J. Song, C. Shen, Y. Yang, Y. Liu, and M. Song, "Transductive Unbiased Embedding for Zero-Shot Learning," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 1024–1033, 2018, doi: 10.1109/CVPR.2018.00113.

[31] Y. Fu, T. M. Hospedales, T. Xiang, and S. Gong, "Transductive Multi-View Zero-Shot Learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 11, pp. 2332–2345, 2015, doi: 10.1109/TPAMI.2015.2408354.

[32] H. Zhang, L. Liu, Y. Long, Z. Zhang, and L. Shao, "Deep transductive network for generalized zero shot learning," *Pattern Recognit.*, vol. 105, 2020, doi: 10.1016/j.patcog.2020.107370.

[33] J. Read. (2016). Multi-label learning from batch and streaming data [PowerPoint slides]. Available: https://jmread.github.io/talks/slides_MAESTRA.pdf

[34] R. W. Hamming, "Error detecting and error correcting codes," in *The Bell System Technical Journal*, vol. 29, no. 2, pp. 147-160, April 1950, doi: 10.1002/j.1538-7305.1950.tb00463.x.

[35] M. Sharma and A. Batra, "Analysis of Distance Measures in Content Based Image Retrieval," *Global Journal of Computer Science and Technology: G Interdisciplinary*, vol. 14, no. 2, 2014.

[36] N. M. Varma and A. Choudhary, "Evaluation Of Distance Measures In Content Based Image Retrieval," *2019 3rd International conference on Electronics, Communication and Aerospace Technology (ICECA)*, 2019, pp. 696-701, doi: 10.1109/ICECA.2019.8821957.

[37] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[38] K. Pulli, A. Baksheev, K. Kornyakov, and V. Eruhimov, "Real-time computer vision with OpenCV," *Commun. ACM*, vol. 55, no. 6, pp. 61–69, Jun. 2012, doi: 10.1145/2184319.2184337.

[39] A. Bhattacharyya, "On a measure of divergence between two statistical populations defined by their probability distributions," Bull. Calcutta Math. Soc., vol. 35, pp. 99–109, 1943.

[40] M. Douze, H. Jégou, H. Sandhawalia, L. Amsaleg, and C. Schmid, "Evaluation of GIST descriptors for web-scale image search," *CIVR 2009 - Proc. ACM Int. Conf. Image Video Retr.*, pp. 140–147, 2009, doi: 10.1145/1646396.1646421.

[41] D. Hershey and B. Wulfe, "Recognizing Cities from Street View Images," 2015, [Online]. Available: http://cs231n.stanford.edu/reports/2016/pdfs/422_Report.pdf.

[42] S. Suresh, N. Chodosh, and M. Abello, "DeepGeo: Photo Localization with Deep Neural Network," pp. 2–8, 2018, [Online]. Available: http://arxiv.org/abs/1810.03077.

[43] G. Schindler, M. Brown and R. Szeliski, "City-Scale Location Recognition," *2007 IEEE Conference on Computer Vision and Pattern Recognition*, 2007, pp. 1-7, doi: 10.1109/CVPR.2007.383150.

[44] N. Vo, N. Jacobs, and J. Hays, "Revisiting IM2GPS in the Deep Learning Era," *Proc. IEEE Int. Conf. Comput. Vis.*, vol. 2017-Octob, no. 1, pp. 2640–2649, 2017, doi: 10.1109/ICCV.2017.286.

[45] L. G. Camara and L. Přeučil, "Visual Place Recognition by spatial matching of high-level CNN features," *Rob. Auton. Syst.*, vol. 133, p. 103625, 2020, doi: 10.1016/j.robot.2020.103625.

[46] W. Zhang and J. Kosecka, "Image Based Localization in Urban Environments," *Third International Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT'06)*, 2006, pp. 33-40, doi: 10.1109/3DPVT.2006.80.

[47] X. Shi, S. Khademi, and J. van Gemert, "Deep Visual City Recognition Visualization," 2019, [Online]. Available: http://arxiv.org/abs/1905.01932.

[48] Z. Chen et al., "Deep learning features at scale for visual place recognition," presented at the 2017 IEEE International Conference on Robotics and Automation (ICRA), May 2017. doi: 10.1109/icra.2017.7989366.

[49] J. Hays and A. A. Efros, "Large-Scale Image Geolocalization," in *Multimodal Location Estimation of Videos and Images*, Springer International Publishing, 2014, pp. 41–62. doi: 10.1007/978-3-319-09861-6_3.

[50] A. Torii, R. Arandjelović, J. Sivic, M. Okutomi and T. Pajdla, "24/7 place recognition by view synthesis," *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1808-1817, doi: 10.1109/CVPR.2015.7298790.

[51] A. Torii, J. Sivic, T. Pajdla, and M. Okutomi, "Visual place recognition with repetitive structures," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 883–890, 2013, doi: 10.1109/CVPR.2013.119.

[52] C. Szegedy *et al*., "Going deeper with convolutions," *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1-9, doi: 10.1109/CVPR.2015.7298594.

[53] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva, "Learning deep features for scene recognition using places database," in *In Advances in Neural Information Processing Systems*, 2014.

[54] G. Tolias and Y. Avrithis, "Speeded-up, relaxed spatial matching," *2011 International Conference on Computer Vision*, 2011, pp. 1653-1660, doi: 10.1109/ICCV.2011.6126427.

[55] Y. S. Avrithis and G. Tolias, “Hough pyramid matching: Speeded- up geometry re-ranking for large scale image retrieval,” *International Journal of Computer Vision*, vol. 107, no. 1, pp. 1–19, 2014.

[56] Z. Jia, Z. Zhang, L. Wang, C. Shan and T. Tan, “Deep Unbiased Embedding Transfer for Zero-Shot Learning,” in *IEEE Transactions on Image Processing*, vol. 29, pp. 1958-1971, 2020, doi: 10.1109/TIP.2019.2947780.

[57] J. Philbin, O. Chum, M. Isard, J. Sivic and A. Zisserman, “Object retrieval with large vocabularies and fast spatial matching,” *2007 IEEE Conference on Computer Vision and Pattern Recognition*, 2007, pp. 1-8, doi: 10.1109/CVPR.2007.383172.

[58] J. Philbin, O. Chum, M. Isard, J. Sivic and A. Zisserman, “Lost in quantization: Improving particular object retrieval in large scale image databases,” *2008 IEEE Conference on Computer Vision and Pattern Recognition*, 2008, pp. 1-8, doi: 10.1109/CVPR.2008.4587635.

[59] H. Jegou, et al, “Hamming embedding and weak geometric consistency for large scale image search,” In *Proc. ECCV*, 2008.

[60] M. Cordts *et al.*, “The Cityscapes Dataset for Semantic Urban Scene Understanding,” *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 3213-3223, doi: 10.1109/CVPR.2016.350.

[61] R. Gomez, L. Gomez, J. Gibert, and D. Karatzas, “Learning to Learn from Web Data Through Deep Semantic Embeddings,” in Lecture Notes in Computer Science, Springer International Publishing, 2018, pp. 514–529. doi: 10.1007/978-3-030-11024-6_40.

[62] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva and A. Torralba, “SUN database: Large-scale scene recognition from abbey to zoo,” *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2010, pp. 3485-3492, doi: 10.1109/CVPR.2010.5539970.

[63] N. Jacobs et al., “The global network of outdoor webcams,” presented at the the 17th ACM SIGSPATIAL International Conference, 2009. doi: 10.1145/1653771.1653789.

[64] N. Jacobs, N. Roman and R. Pless, "Consistent Temporal Variations in Many Outdoor Scenes," *2007 IEEE Conference on Computer Vision and Pattern Recognition*, 2007, pp. 1-6, doi: 10.1109/CVPR.2007.383258.

[65] A. Farhadi, I. Endres, D. Hoiem and D. Forsyth, "Describing objects by their attributes," *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 1778-1785, doi: 10.1109/CVPR.2009.5206772.

[66] O. Cuadros Linares, G. M. Botelho, F. A. Rodrigues, and J. B. Neto, "Segmentation of large images based on super-pixels and community detection in graphs," *IET Image Processing*, pp. 60–80, 2017.

[67] *World Cities Database*. (2021). SimpleMaps.com, Pareto Software, Jul. 2021. [Online]. Available: https://simplemaps.com/data/world-cities.

[68] *SimpleMaps.com*. (2021). Pareto Software, Accessed: Jul. 16, 2021. [Online]. Available: https://simplemaps.com/custom/world.

[69] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva and A. Torralba, "Places: A 10 Million Image Database for Scene Recognition," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 6, pp. 1452-1464, 1 June 2018, doi: 10.1109/TPAMI.2017.2723009.

[70] *Convnetdraw*. (2019). C. Bovar. Accessed: Jul. 18, 2021. [Online]. Available: https://github.com/cbovar/ConvNetDraw.

[71] N. Pattanajak and H. Malekmohamadi, "Improving a 3-D Convolutional Neural Network Model Reinvented from VGG16 with Batch Normalization," *2019 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*, 2019, pp. 45-50, doi: 10.1109/SmartWorld-UIC-ATC-SCALCOM-IOP-SCI.2019.00052.

[72] L. Chen, H. Fei, Y. Xiao, J. He and H. Li, “Why batch normalization works? a buckling perspective,” *2017 IEEE International Conference on Information and Automation (ICIA)*, 2017, pp. 1184-1189, doi: 10.1109/ICInfA.2017.8079081.

[73] J. Duchi, E. Hazan, and Y. Singer, “Adaptive subgradient methods for online learning and stochastic optimization,” *Journal of Machine Learning Research*, vol. 12, pp. 2121–2159, 2011.

[74] S. Ruder, “An overview of gradient descent optimization algorithms,” arXiv:1609.04747, 2016.

[75] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, “Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization,” *Int J Comput Vis*, vol. 128, no. 2, pp. 336–359, Oct. 2019, doi: 10.1007/s11263-019-01228-7.

[76] *pytorch-grad-cam*. (2021). J. Gildenblat. Accessed: Jul. 19, 2021. [Online]. Available: https://github.com/jacobgil/pytorch-grad-cam.

[77] *Imagga Tagging*. (2021). Imagga Technologies Ltd. Accessed: Oct. 3, 2020. [Online]. Available: https://imagga.com/solutions/auto-tagging.

[78] *Word Art Creator*. (2021). WordArt.com. Accessed: Jul. 20, 2021. [Online]. Available: https://wordart.com/create.

[79] *Name that Color*. (2007). C. Mehta. Accessed: Jun. 14, 2020. [Online]. Available: https://chir.ag/projects/ntc/ntc.js.

[80] R. Chakravarti and X. Meng, “A Study of Color Histogram Based Image Retrieval,” *2009 Sixth International Conference on Information Technology: New Generations*, 2009, pp. 1323-1328, doi: 10.1109/ITNG.2009.126.

[81] A. K. Jain and A. Vailaya, “Image retrieval using color and shape,” *Pattern recognition*, vol. 29, no. 8, pp. 1233–1244, 1996.

[82] Y. Bengio, “Practical recommendations for gradient-based training of deep architectures,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 7700 LECTU, pp. 437–478, 2012.

[83] S. A. Rather. *PseudoCode-AlgorithmTemplate*. (2021). Accessed: Jul. 23, 2021. [Online]. Available: https://v1.overleaf.com/latex/templates/pseudocode-algorithmtemplate/zrqcdnkhqvgb.pdf

[84] *Canva.com*. (2021). Canva Pty Ltd. Accessed: Aug. 02, 2021. [Online]. Available: https://www.canva.com/design/play

[85] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” *Commun. ACM*, vol. 60, no. 6, pp. 84–90, May 2017, doi: 10.1145/3065386.

[86] Z. Zeng, J. Zhang, X. Wang, Y. Chen, and C. Zhu, “Place Recognition: An Overview of Vision Perspective,” Applied Sciences, vol. 8, no. 11, p. 2257, Nov. 2018, doi: 10.3390/app8112257.