الجمهورية الجزائرية الديمقراطية الشعبية

MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH
UNIVERSITY YAHIA FARÈS OF MÉDÉA

FACULTY OF TECHNOLOGY

Department of Electrical Engineering



Final studies dissertation

Presented for obtaining the Master Degree

in Electrical Engineering

Option: Electronics of Embedded Systems

THEME

Deep learning-based method for automatic image annotation

Proposed and directed by,

Dr. MAAZOUZ Mohamed

Dr. HENTABLI Hamza

Prepared and presented by,

Mr. CHABIRA Abdelghani

Academic year :2020/2021

Acknowledgements

I wish to thank my project supervisors for being more than generous with their expertise and precious time, Dr. Maazouz and Dr. Hentabli who have helped me undertake this research; Dr. Maazouz for teaching me the basics of programming and signal processing, and Dr. Hentabli helped me to discover the world of machine learning and artificial intelligence.

A special thanks goes to the members of jury, Dr. Bengherbia and Dr. Kheddar for accepting to examine and evaluate my work in order to valorize the obtained results.

I would like to acknowledge and thank the Department of Electrical Engineering for allowing me to conduct my research and providing any assistance requested. Special thanks goes to Mr.Guedmani, Mr.Morsli, and Mr.Gacemi.

I would like to thank all the professors who left a special touch in me, Pr.Batel has been an ideal teacher and mentor, offering advice and encouragement with a perfect blend of insight and humor. I'm proud of, and grateful for my time studying at his hands. Dr. Maazouz for always answering any question I have and being available all the time. Dr.Kara for encouraging me to attain my goals in life, Dr. Chadli for teaching me the basics of electronics, Dr. Bengherbia and Dr.Lachenani for teaching me the basics of networks and telecommunications, Dr. Kheddar for teaching me object-oriented programming and more advanced concepts in computer science, Dr.Reggai for teaching me assembly and microprocessors, Dr. Saidi for teaching me the basics of chemistry, Dr. Bouzefrane and Dr. Chehba-mouna for teaching me the basics of analysis and probability, Dr. Kechidi and Dr. Amari for teaching me the basics of physics, Dr. Ould-zmirli for teaching me the basics of embedded systems and FPGA design, Dr. Rebai for initiating me to the world of radio-frequency, Dr. Kermezli for teaching me computer science and believing in me from the very beginning, Dr. Bentouati for teaching me control systems and always giving me advice, Dr. Bouhedda and Dr. Benyezza for teaching me industrial electronics and how to program APIs, and Dr. Hamidia for initiating me to the world of numerical analysis which was very useful.

Finally, I would like to thank Dr.Chiba, Mr.Mouloud, and our faculty dean Dr. Bachene for giving me access to the laboratory where I learned a lot of new skills and where I met Mr.Aniba, Mr.Bailish, Dr. skender and Dr. Kaddour and Dr. Boulesnam who have helped me a lot.

DEDICATION

In memory of my dear grand-parents, I dedicate this work.

I dedicate my dissertation work to my family and many friends. A special feeling of gratitude to my loving parents, Amar and Malika whose words of encouragement and push for tenacity ring in my ears. My sister Asma, and my brother Hamza have never left my side and are very special.

I also dedicate this dissertation to my dear friends Abdelhafid and Abdessalem who have supported me throughout the process. I will always appreciate all they have done, especially my childhood friend Azzeddine for helping me develop my technology skills at a young age, Rafik for teaching me the basics of hardware design and repair, Mosaab, Mohammed Bouderballa, Redha and Adel for being always on my side.

I dedicate this work to my dear uncles Djeloul, Abdelkader, Mhammed, Khadidja, Fatma, Zohra, Fatiha, Azzedine, and Houria, through good times and bad, your kindness and extensive support have been ever-present in this important time of my life, for which I am eternally grateful.

ملخص

من أجل تسهيل العملية البحثية، عادة ما يتم استخدام طرق يدوية للتعليق على الصور من أجل إنشاء علامات توضحية أو وصفية، الأمر الذي قد يستغرق وقتا طويلا. تم الاستفادة من الخصائص التي توفرها شبكة عصبونية إلتفافية من أجل تصنيف الصور و ضغط الميزات باستخدام مشفر آلي، إذ يمكن إعتبار هذه الخطوة مرحلة أولية تسبق عملية أتمتة مهمة التعليق التوضيحي التلقائي للصورة، مما سيسمح باسترجاع الصور وتوفير روابط معلومات حولها، كالصور المشابهة والعبارات التوضيحية.

نظرا لكونها الخطوة الأهم في التعليق التوضيحي الآلي، تم إعطاء خاصية الإستخراج الأهمية الأكبر والتي بدونها لن تكون المرحلة النهائية ممكنة. كما تم استخدام التحليل الإحصائي ومختلف التجارب على الخصائص المستخرجة لتحديد أهم سماتها وكذا موثوقيتها.

ومن أجل التقليل من الفجوة في المعاني (الدلالية)، إقترحت الدراسة طريقة بسيطة لقياس الحالات المتشابهة، على إعتبار أن كل فئة من العلامات تمثل بشعاع مميز وحيد، هو نفسه يصبح المعرف الخاص بتلك الفئة، وعلى هذا الأساس تتم عملية المقارنة.

كلمات مفتاحية: التعليق التوضيحي التلقائي للصورة، التعلم العميق ، الشبكة العصبية التلافيفية ، أشعة الميزات ، التشفير التلقائي ، الفجوة الدلالية

Abstract

Annotating images in order to create tags or meta data for easy searching is often a manual, and thus a time-consuming process. By utilizing a convolutional neural network to perform image classification, and then extracting the features produced by this neural network, one can automate the annotation process, an Autoencoder is then used to compress these feature vectors; this can be considered as a preliminary step which precedes the final task that is to automatically generate basic search tags. These tags can then be used to facilitate the task of image retrieval making the images easier to find, and providing links to other information such as related images and explanatory texts.

A very special attention was given to the feature extraction step since it is considered as the cornerstone of automatic image annotation, and without it, the final annotation would not be possible; a statistical analysis and various experiments were conducted on the extracted features in order to determine their characteristics and test out their reliability.

To reduce the semantic gap problem, we propose a simple similarity measure method where we assume that each class of tags is represented by a single universal feature vector, this

feature vector then becomes an identifier to that particular class, a new test image's extracted feature vector will then be compared with this identifier in order to determine whether it belongs to that particular class or not.

Keywords: Automatic Image annotation, Deep Learning, Convolutional Neural Network, Feature Vectors, Autoencoder, Semantic gap

Résumé

Annoter des images afin de créer des étiquettes ou des métadonnées pour une recherche facile est souvent un processus manuel, et donc épuisant.

En utilisant un réseau neuronal convolutif, pour effectuer une classification d'images, puis en extrayant les caractéristiques produites par ce réseau de neurones, on peut automatiser ce processus d'annotation, un auto-encodeur est ensuite utilisé pour compresser ces vecteurs de caractéristiques ; cela peut être considéré comme une étape préliminaire qui précède la tâche finale qui génère automatiquement des étiquettes.

Ces annotations peuvent ensuite être utilisées pour faciliter la tâche de récupération d'images, rendant les images plus faciles à trouver et fournissant des liens vers d'autres informations, telles que des images similaires et des textes explicatifs.

Une attention toute particulière a été portée à l'étape d'extraction des caractéristiques puisqu'elle est considérée comme la base de l'annotation automatique d'images, et sans elle, l'annotation finale ne serait pas possible ; une analyse statistique et diverses expérimentations ont été menées sur les caractéristiques extraites afin de déterminer leurs propriétés et de tester leur fiabilité.

Pour réduire le problème d'écart sémantique, nous proposons une méthode de mesure de similarité simple où nous supposons que chaque classe d'étiquettes est représentée par un seul vecteur de caractéristique universel, ce vecteur de caractéristique devient alors un identifiant pour

cette classe particulière, un nouveau vecteur de caractéristiques extrait d'une image de test sera ensuite comparé avec cet identifiant afin de déterminer s'il appartient à cette classe particulière.

Mots clés : Annotation d'image automatique, apprentissage en profondeur, réseau de neurones convolutifs, vecteurs de caractéristiques, encodeur automatique, écart sémantique.

Table of Contents

Acknowledgements II

DEDICATION III

ملخص IV

Abstract IV

Résumé V

Table of Contents VII

List of Figures IX

List of Tables XI

List of Abbreviations XII

Chapter I Introduction 1

1 General Introduction 1

2 Overview 2

3 Problem Background 4

4 Problem Statement 6

5 Research Questions 7

6 Research Objectives 7

7 Research Scope and Limitations 8

8 Conclusion 8

Chapter II State of the Art 9

1 Introduction 9

2 Literature review 9

2.1 Classical Image Retrieval Methods 10

2.1.1 Text Based Image Retrieval 11
2.1.2 Content Based Image Retrieval 12
2.2 Automatic Image Annotation 19
2.2.1 Feature extraction 21
3 Conclusion 26
Chapter III Research methodology 27
1 Introduction 27
2 Research Strategy 27
3 Dataset Used 28
4 Deep learning Model 35
4.1 Feature extraction 35
4.1.1 CNN classification model 35
4.1.2 The Compressing Autoencoder 60
4.1.3 Analysis of features 61
5 Final annotation (Similarity measure) 69
6 The Final Automatic Image Annotation Architecture 69
7 Results and Discussion: 71
8 Conclusion 72
Conclusions and perspectives 73
Bibliography 74

List of Figures

Figure I.1 The evolution of data volume over the years [1].....1
Figure I.2 An example of a common web-based image retrieval using text [3].....4
Figure II.1 a typical TBIR system 11
Figure II.2 a typical CBIR system architecture [10].....14
Figure II.3 An example of a color based CBIR [19].....15
Figure II.4 Examples of images containing common textures.....17
Figure II.5 The difference between boundary based and region-based representations [24].....18
Figure II.6 A general architecture of an Automatic Image Annotation model [34].....20
Figure III.1 The methodology used for research 28
Figure III.2 The different categories of the COCO dataset [62].....29
Figure III.3 A sample of the COCO image Dataset [61] 30
Figure III.4 An example of a COCO image with its respective annotation 31
Figure III.5 The process of transforming the raw dataset images into a useful format.....33
Figure III.6 A sample of the newly created modified COCO Dataset 34
Figure III.7 A typical CNN architecture [64] 36
Figure III.8 An example of a convolution operation between an input matrix and a kernel [65].....38
Figure III.9 An example of an input matrix after the ReLU operation [67] 40
Figure III.10 An example of an input matrix after a Max-Pool operation [69] 41
Figure III.11 A typical Fully Connected layer [70].....42
Figure III.12 A plot of the Cross-Entropy Loss function [72].....44
Figure III.13 The Base single weight layer CNN model 45
Figure III.14 Training graphs for the model which used Batch-Norm and the one that did not.....54
Figure III.15 The final classification CNN model 55
Figure III.16 Learning Rate Step decay scheduling.....57
Figure III.17 How features are extracted 59
Figure III.18 Architecture of Autoencoder used.....61
Figure III.19 A sample of two images from the class 'person'.....62
Figure III.20 An overlayed plot of two feature vectors belonging to two random instances of the class 'person' 62
Figure III.21 The result from a correlation between the two extracted features of the class 'person' 63
Figure III.22 A sample of two images from different the classes, one from ‘umbrella’ and one from ‘train’.....64
Figure III.23 An overlayed plot of two feature vectors belonging to two different classes, one of ‘train’ and one from ‘umbrella’.....64
Figure III.24 The result from a correlation between the two extracted features of two different classes, one from Train and the other from Umbrella 65

Figure III.25 A scatter plot of the PCA reduced feature vectors of size 128 labeled correctly and their automatically clustered version 67
Figure III.26 A scatter plot of the PCA reduced feature vectors of size 50 labeled correctly and their automatically clustered version 68
Figure III.27 The Final Automatic Image Annotation Architecture 70
Figure III.28 Randomly selected images with the smallest Euclidean distance between their feature vectors in the class ‘cat’ 71
Figure III.29 A scatter plot of PCA reduced feature vectors and their respective class identifier drawn in black 72

List of Tables

Table III.1 The subset of COCO classes that were used in our work 32
Table III.2 The experimental results from using the single weight layer CNN models.....46
Table III.3 The experimental results from the 2 weight layers CNN models 48
Table III.4 The set of configurations for a 3 weight layers CNN models.....51
Table III.5 The set of configurations for the 4-9 weight layers CNN models 52

List of Abbreviations

AIA: Automatic Image Annotation

DL: Deep Learning

CNN: Convolutional Neural Network

IDB: Image Database

DBMS: Data Base Management System

CV : Computer Vision

TBIR: Text Based Image Retrieval

CBIR: Content Based Image Retrieval

HOG : Histogram of Oriented Gradients

ANN : Artificial Neural Network

RNN: Recurrent Neural Network

KNN: K-Nearest Neighbors algorithm

MLP : Multi-Layer Perceptron model

RLE: Run-Length Encoding

FC: Fully Connected

FLOPS: Floating Point Operations Per Second

FFT: Fast Fourier Transform

LSB: Least significant Bit

SQL: Structured Query Language

MPEG: Moving Picture Experts Group standard

SIFT: Scale-Invariant Feature Transform

Chapter I Introduction

1 General Introduction

Many experts today agree that data is the new oil, meaning It's considered the most valuable resource in the world because of the way it's influencing today's society by driving businesses, affecting decision-making, and helping us understand more about our world, the reason behind this is the mind-boggling amount of data we create each day. There are 2.5 quintillion bytes of data created each day at our current pace, but that pace is only accelerating with the growth of the Internet of Things (IoT). Over the last two years alone 90 percent of the data in the world was generated, and most of this data is in the form of images/videos, most of it is uploaded to social media, that is every minute, Snapchat users share 527,760 photos, YouTube users watch 4,146,600 videos and Instagram users post 46,740 photos. Figure I-1 shows the evolution of data volume over the last 15 years.

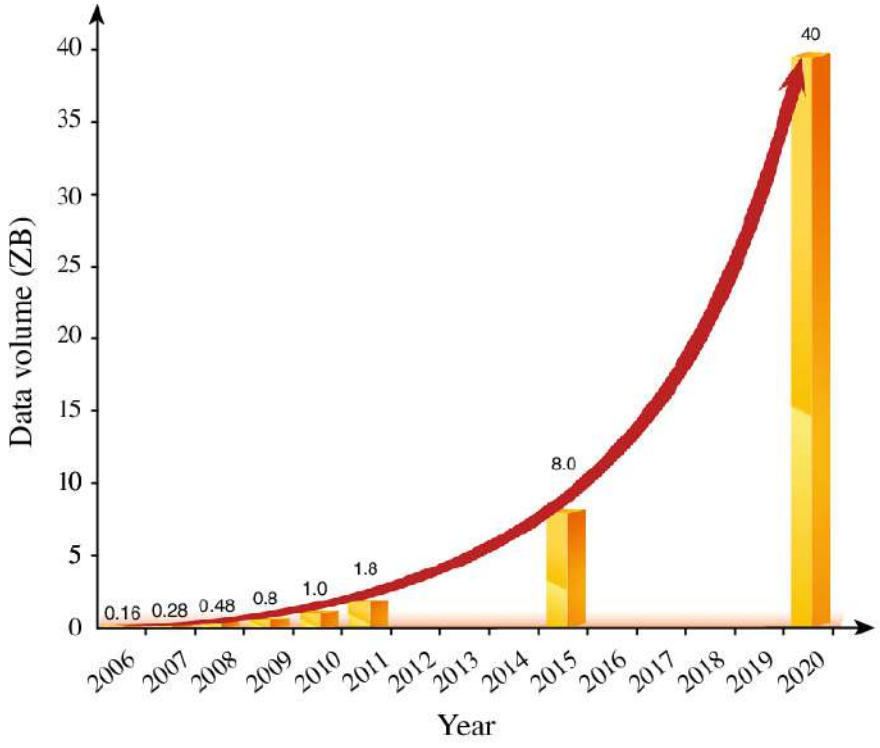


Figure I.1 The evolution of data volume over the years [1]

This recent boom in data is attributed to two main factors, the first being the advancement of the technological processes with the invention of new tools for creating, sharing, and consuming data. It is worth mentioning that the smart phone and social media revolution were the biggest catalysts and the main contributors to this growth, and it is expected that in the future faster networks such as 5G will, with no doubt, increase the amount of data available as the International Data Corporation (IDC) predicted that in 2025 more than 175 zettabytes of new data will be created around the world [2]; these new technologies will enable us to store very large amount of data while also having the ability to access it very fast. The second factor is the development of sophisticated analytical processing techniques to treat this data, which the leading role is attributed to machine learning.

It is also worth mentioning that these two factors are somewhat dependent and interconnected as they each contribute to each other making them into a closed loop system, meaning that without the huge data ,the means of accessing it and the compute power available today, researchers would have not been able to develop new analytical techniques to treat this data, and this is clearly apparent in the history of machine learning as it was already used in the 60s but only gained popularity recently in the 90s during which computers became popular and cheap; add to this the creation of the internet, which led to an exponential rise of data ever since.

This explosion in of rich information poses the challenging problems of browsing, indexing and searching multimedia contents because of the data size and complexity, thus there is a growing demand for new techniques that can efficiently process, model, and manage image/video databases.

2 Overview

The work described in this dissertation is focused on finding a new method for automatic image annotation, particularly one which is based on deep learning. A big part of this work is focused on finding the best feature vectors that represent the most important concepts in a given image from our dataset with respect to the query at hand, the difference between previous work and ours is that we’re not trying to manually handcraft the feature vectors that may represent the most relevant semantics from a given image, but in fact we’re letting the machine learning algorithm find its own optimal features that it see fits, using the accuracy of the deep learning model as the main criterion. It is essentially an attempt at proving that deep learning can in fact

achieve better results than classical methods while also proving that these generated feature vectors are in fact closely related to semantic concepts present in the image making them interpretable.

Our work progresses in the following way, Chapter I outlines the research problem in order to investigate its significance in a contemporary context, by asking related questions and hypotheses while laying out the reasoning behind them and giving some background on it, we basically try to establish a strong foundation for building on top of in the later chapters.

Chapter II serves as a summary of the past work that is related to our own topic, it is essentially a review of the state of the art of automatic image annotation, where we summarize the literature that is relevant to our research project in a historically ordered manner, bringing out where the gaps in the literature are, and how our research will help to fill in one or more of these gaps, we also show the different approaches to research methodology that were followed in previous work; it begins with describing the early years of image retrieval (Section II.2.1) and how the field of image retrieval came into existence with the main actors being the Data Base Management System community (DBMS) that contributed largely during the early years with their main methods being Text-Based Image Retrieval (TBIR) (Section II.2.1.1), and after that how another group of researchers from the Computer vision community (CV) used their own approach to try solve the problem which spawned another field that is based on the content of an image called CBIR (Section II.2.1.2), and from then on, in a marriage of fields Automatic image annotation was born and is described in (Section II.2.2).

In Chapter III, the research methodology is described in sufficient detail that readers come away with a clear understanding of how the study will be conducted, and future researchers would know precisely what procedures should follow if they want to replicate this study.

And finally, the dissertation ends with some highlighted points that concludes and gives perspective to our work.

3 Problem Background

Most of today's image search engines rely on textual information or meta data that appear either close to the image in an article for example or embedded inside to match certain images with the query keywords, as shown in Figure I.2, this is mainly attributed to the very fast retrieval times inherent in text-based search systems; however, this method will exclude a lot of images that either no text is written beside them or the text doesn't have any correlation with the image content, that is why the main focus today is on new methods that exploit the actual pixel level content of the image to map it into a higher semantic concept.



Figure I.2 An example of a common web-based image retrieval using text [3]

This problem can be approached from two different angles, one by assuming that this semantic concept can be represented by an image/video as the query, which is called CBIR, the other is based on textual keywords to represent the semantics and is called image annotation. The first method fell out of researcher's interest because of the inherent difficulty of using low level visual features such as color or textures to adequately model the semantic contents of images, as well as the huge difficulty when it comes to engineering handmade low level visual features,

moreover it is very difficult to find exactly the relevant information from the user's perspective in a complex image query, that is why the latter method is more interesting.

Image annotation is defined as the task of annotating an image with labels, typically involving human-powered work [4], where the aim is to map the relationship between keywords and visual data. however more automatic image annotation methods have been gaining momentum, mainly due to the large labeled datasets widely available nowadays, and to the advances in machine learning techniques.

AIA is a challenging task which has not been solved satisfactory for general real-world applications. Many solutions cover only small specialized domains and they usually work with a very limited vocabulary set [5], what adds to the difficulty is that AIA is a relatively new field that has been getting more attention only recently. AIA is similar to object detection, only minus the localization because we don't particularly care about the exact position of each object in the image.

Assuming semantically labelled image samples are collected and represented with low level features, a machine learning algorithm can then be trained using the feature to semantic label matching; and once trained, the algorithm can be used to annotate new image samples [6] . There are two major aspects of AIA, feature extraction and semantic learning, firstly pixel level content of the image is converted to the feature space, these features can be either global which are computed from the entire image for instance histograms, or they can be local where an image is divided into several regions and then a set of features is computed for each region, giving us a bag of features but they may not be as accurate because they require prior segmentation of the image. And finally, a relationship is learned between these features and the higher semantic concepts.

One of the major challenges in AIA research is the curse of dimensionality and its negative effects on memory and compute time, where the performance degrades dramatically when feature dimensions is too high. This is where research is trying to exploit the fact that some features are more important than others meaning they contribute more to the final result to eliminate the non-important features and reduce dimensionality, this is known as visual feature selection (VFS). Currently all features have certain limitations in describing the information present in the image, so no single feature is powerful enough to be used indiscriminately in all domains.

4 Problem Statement

Human perception of images is a very complex and not well understood task, often attributed to the richness and amount of raw information present in images, which consequently makes AIA also a very complex task that requires a rich vocabulary to bridge the semantic gap between the raw images and the text labels, moreover the system needs to be intelligent enough to use that vocabulary properly and to establish associations and relations between the textual labels and concepts present in the image.

However, if we assume that there is a well-defined bound dictionary of labels to map the images into, the problem of AIA can be reduced to a simple classification problem, where the images are presented as inputs, in the middle we have some sort of feature vectors, and finally in the end we get the correct labels at the output. This is one of the ideas we try to explore their feasibility in this dissertation, in order to reduce human efforts to engineer handmade features which may take allot of time and be constrained to one type of images or a single particular domain.

In recent years, deep learning became the norm when it comes to solving machine learning problems because it achieves far better empirical results than normal ANNs; catalyzed by the ever-increasing compute power and memory of our computer systems, and coupled with the technological leaps in GPUs and hardware accelerators alike has made training large models that used to take months now possible in only a few hours. It's also worth mentioning that widely available software tools such as Keras and Pytorch made it very simple to implement any machine learning algorithm with only a few lines of code, as well as the rise of online severs such as Kaggle or Google Collab has made using very expensive hardware to accelerate training totally free of charge and at the tip of our hands. All this coupled with the huge number of largely available datasets online, practically made deep learning the solver of most machine learning problems and the goto algorithm in a wide variety of domains, especially when it comes to image processing and computer vision.

One of the fields that largely benefited from this DL renaissance, is classification, where learning features at multiple level of abstractions allow the system to learn complex functions mapping from input to output directly from data, without depending completely on human handcrafted features, moreover if we observe these hierarchies' interpretability, we find that each level learns to transform its input data into a slightly more abstract and composite representation.

In an image recognition application, the raw input may be a matrix of pixels; the first representational layer may abstract the pixels and encode edges; the second layer may compose and encode arrangements of edges; the third layer may encode a nose and eyes; and the fourth layer may recognize that the image contains a face. Importantly, a deep learning process can learn which features to optimally place in which level on its own. This does not completely eliminate the need for hand-tuning; for example, varying numbers of layers and layer sizes can provide different degrees of abstraction [7].

The great success of deep learning in many applications is based on the clever idea of constructing sufficiently large nonlinear function spaces throughout the composition of layers of linear and simple nonlinear functions (in the name of activation function). Widely used activation functions include the Sigmoidal function, Rectified Linear Unit (ReLU) and many more, The nonlinearity of activation functions used in deep learning models are crucial for the success of predictive models [8], this large number of layers is what encompasses and models the different patterns present in the data, it is also worth mentioning that larger networks require larger datasets in order to functions properly and avoid underfitting which results in an optimistic and high variance estimation of model performance.

5 Research Questions

In this dissertation, two main investigations are probed:

1. Will a deep learning classification model be able to learn the features necessary for the AIA task itself without the need for handcrafting them ourselves on our particular dataset?
2. Will the feature extraction scheme be able to output similar feature vectors for instances that belong to the same class of labels making it it’s unique identifier?

6 Research Objectives

For this research, the objectives are defined as:

1. To automatically generate object representative feature vectors using a convolutional deep learning model.

2. To improve the generated feature vectors using auto-encoder model in order to generalize it to the whole class.
3. To generate the annotation of images using the improved feature vector

7 Research Scope and Limitations

For this research, the constraints are defined as:

- Only a single object is present in a given dataset image, and this object must be large and clear enough that when resized it is still apparent.
- Dataset images must contain only the pixels of that particular object meaning the background scenery is removed.
- The deep learning model must be small enough that we don't surpass the available memory in our system.

8 Conclusion

In this chapter, we have started by showing the significance of the study in a contemporary context while giving some background on it and how it will be useful for our daily lives; we then formulated the problem at hand by translating the generalist and somewhat vague aims of the study into quantifiable, clear and concise research questions that will help to show our progress later on; and finally we have defined the exact research scope that acts like a boundary box to our work making it more practical.

Chapter II State of the Art

1 Introduction

This chapter serves as a survey for previous works in AIA, with a particular focus on the feature extraction techniques that were used throughout the many different eras of computer vision evolvement, invoking the history behind it along the way and how this particular domain has evolved to become what it is today, while also shedding light on the many technological mile stones this domain has hit, and providing arguments on why some of the methods fell out of interest or became obsolete. It is essential to this dissertation because it establishes the current state of the art of this particular domain while also building a basis for discussing results and interpretations. with a purpose of quantifying our own contribution by placing it in the context of previous works, the implications of the previous studies are summarized, serving later as a comparison measure to contrast it with the study under investigation, it basically summarizes what is known and identifies what is unknown about the topic of the dissertation, another reason for this is that by invoking the history of this domain, our ability to recognize the current trends in this particular field will grow, leading us to predict the possible futures of Automatic Image annotation.

2 Literature review

Man has traditionally outperformed machines for most tasks. One of these tasks in particular is image retrieval, one reason which causes this distinction is that text is man's creation, while typical images are a mere replica of what man has seen since birth, concrete descriptions of which are relatively elusive. Add to this the theory that the human vision system has evolved genetically over many centuries. Naturally, the interpretation of what we see is hard to characterize, and even harder to teach to a machine. Yet, over the past decade, ambitious attempts have been made to make computers learn to understand, index, and annotate pictures representing a wide range of concepts, with much progress [9].

It is important to note that the field of Image retrieval has undergone quite a significant growth along the years since its emergence in the 1970's, researchers coming from various different fields and communities all tried their hand at solving this particular problem, using their own scientific backgrounds they approached the problem from different angles, each having a different formulation of the problem at hand, they successfully invented new techniques along the

way, with different methodologies they managed to drive their own influence to the field, leading to the diversified field of today. Of these communities who contributed largely are the DBMS and CV communities, both having their own perception of the problem at hand and how to solve it, they successfully spawned along the way new closely related fields. The DBMS community saw that the much simpler TBIR was the right solution due to the very poor performance of computer systems at the time, while the Computer Vision folks, motivated by the advancements in technology during later years, wanted to use the actual pixel level content to represent semantics which led to the emergence of Content Based Image Retrieval systems, Amidst such marriages of fields Automatic image annotation was born, inheriting from both text based methods as well as content based methods, it proved to be the most difficult challenge due to the large semantic gap, it is still far from being solved sufficiently to be used in large scale real world applications.

2.1 Classical Image Retrieval Methods

Starting from the 1970's, with the rise of image retrieval as an active research area, researchers having found at their disposal digital images databases of relatively small size at the time and with the fact that these databases were mostly constrained to a particular theme or domain, they opted for the more traditional and most common method of utilizing some form of adding extra textual meta data such as captioning or titles/keywords to the images, making it a manual process that needs human labor, but since digital imagery was far from popular nor accessible at the time, that human labor was not so intense. Most of the research was carried out by the DBMS community, giving rise to the TBIR approach.

An image retrieval system can be used for browsing, searching and retrieving images from a large database of digital images. Most traditional and common methods utilize some method of adding metadata such as captioning, keywords, or descriptions to the images so that retrieval can be performed over the annotation words. As part of this general problem, image indexing and retrieval have been an active research area for more than half a century where the focus of interest is the automatic extraction of semantically-meaningful information from the content of an image. In this work, image retrieval is classified mainly into two types:

2.1.1 Text Based Image Retrieval

Text-based image retrieval depends mostly on annotations that were added manually, these can range from keywords, descriptions, or even collateral text that is accidentally available with an image (captions, subtitles, nearby text), these systems were first developed in the 1960s with databases that contained only textual data, and then in 1970s they were generalized to image datasets by applying traditional text retrieval techniques to image annotations or descriptions [10].

It is worth noting that although text retrieval is an old technique, however, many techniques in image retrieval were either directly taken from this domain by sometimes being reinvented or generalized, or were at least inspired by these old methods [11]. Figure II.1 shows a typical TBIR system.

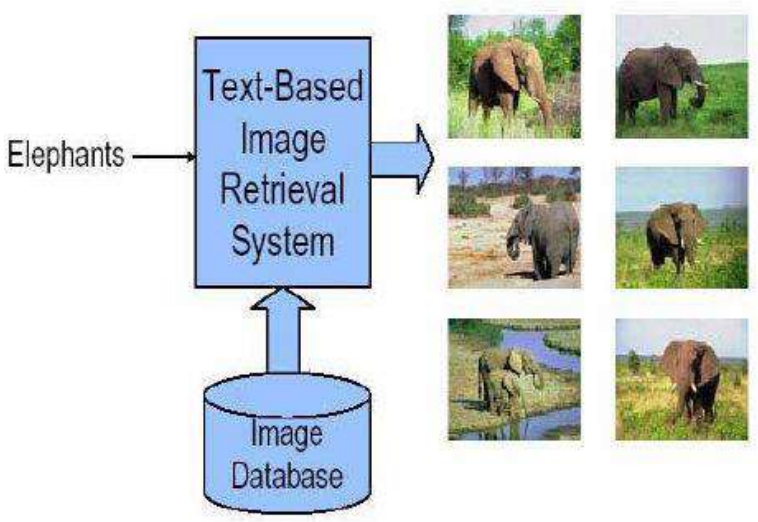


Figure II.1 a typical TBIR system

2.1.1.1 The conventional method

One of the simplest approaches is to consider image data as just being like any other ordinary data type, treating it just like how we would treat textual data or binary data, making the handling of image data as well as conventional coded data unified; one of the first and most famous works of this kind is accredited to the work done by a Japanese IBM research team called Aggregate Data Manager (ADM) [12], This approach enabled the usage of a simple but powerful relational data language called “SYSTEM-R” which was the first implementation of the Structured

Query Language (SQL), which has ever since become the standard relational data query language; it used Boolean operators (AND, OR, NOT…) to combine between simple search terms in a given query and it used no ranking system whatsoever. Another similar work is done by Tang, et al. [13] where they use an extended version of SQL that contains logical I/O to facilitate the specification of an I/O device in a complex system.

2.1.1.2 Examples of image database systems

Some of the image database systems that were created with the sole intention of advancing research have successfully made it later on into a complete software package that was even sometimes commercialized; relying in its core on text based image retrieval techniques, many systems provided a flexible and easy way to adapt an interface to an image database system, an example of an image database (IDB) system developed in the early days is the so-called Graphics-oriented Relational Algebraic Interpreter system (GRAIN) [14]; developed at the University of Illinois at Chicago Circle in 1977. It was initially used as a part of a medical information system, the main goal was to illustrate that by utilizing a database system for the storage of the actual picture description, users can incorporate graphics into their applications very simply without becoming a graphics or programming expert. GRAIN is characterized by the introduction of a concept of logical pictures which are collections of picture objects that can be considered as masks for extracting meaningful parts from an entire image. The language GRAIN, which is a command-oriented language, supports all the commands available in the relational symbolic query language RAIN and some other additional commands for defining, retrieving and displaying both the logical pictures and the physical pictures. Recently, the GRAIN system has been extended by introducing the concept of generalized zooming, its extended system, Distributed Image Management And Projection (DIMAP) [15], is intended to be used as a geographical information system.

2.1.2 Content Based Image Retrieval

During the 90's, and in particular in 1992, a Japanese engineer from the Electrotechnical laboratory of Tsukuba research center named Toshikazu Kato described experiments that delve into the retrieval of images from a database, he used a visual mechanism based on colors and shapes present in the image in order to retrieve similar images in the database [16], this was considered as a new approach to the problem of image retrieval because it used pixel level content present in the image, giving rise to the term of Content Based Image Retrieval, this was a

divergence from the previous approaches that mostly relied on text-based methods carried out by the DBMS community; this work successfully demonstrated that an integration of the field of CV is possible and can contribute to solving the problem of image retrieval using new techniques.

A typical CBIR system, as shown in Figure II.2, consists of two main parts, the first part being the feature extraction which automatically extracts visual attributes such as color, shape, texture and spatial information for each image in the database based on its pixel values, these features can be manually engineered (i.e., pre deep learning era) for better performance, or they could be learned using different machine learning algorithms. The feature data for each image is way smaller in size compared to the original image. Thus, the feature vector contains an abstraction (compact form) of the images, meaning that each image is represented by a compact representation of its contents in the form of a fixed length real-valued n-dimensional feature vectors or signature. The second part is the similarity measure which is responsible for measuring how similarly close are two given images based only on their respective feature vectors; meaning when the users usually formulate a query image and presents it to the system, The system automatically extracts the visual attributes of that query image, and then identifies images in the database whose feature vectors closely resembles those of the query image, and sorts the best similar objects according to their similarity value. During operation, the system processes less compact feature vectors rather than the large size image data thus giving CBIR its cheap, fast and efficient advantages. CBIR systems can be used in one of two ways. First, exact image matching, that is matching the query image to one of the database images. Second is approximate image matching, which is finding most closely match images to a query image [17].

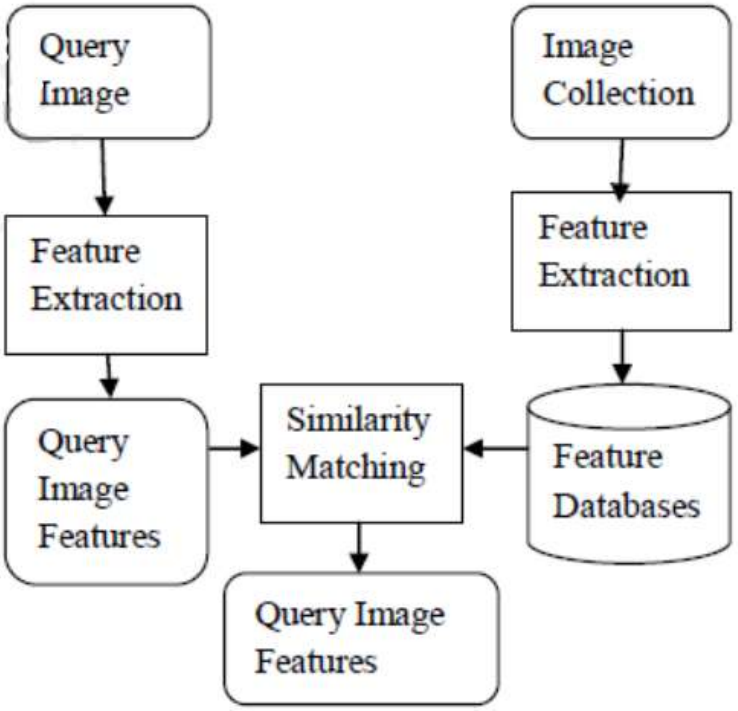


Figure II.2 a typical CBIR system architecture [10]

2.1.2.1 Feature extraction

In computer vision and image processing, a feature is a piece of information about the content of an image; this feature can be either a representation of the whole image and is called a global feature, or it can be focused on properties relating to a certain region of the image making it a local feature. Its main goal is to represent images in an alternative format which is as compact as possible in terms of constituent semantic entities, such a representation downsizes images to their semantic entropy, rather than their statistical entropy. Features may be specific structures in the image such as points, edges, objects or shapes defined in terms of curves or boundaries between different image regions, or they can be more complex [18].

Feature extraction is the first step of content-based image retrieval and is considered a low-level operation. It is the process of extracting image features by examining every pixel to see if a certain feature exists in the image. Once obtained, visual features act as inputs to subsequent image analysis tasks such as similarity estimation, concept detection, or annotation. One type of features

can only represent part of the image properties, there is no single “best” feature that gives accurate results in any general setting. Usually, a combination of features is needed to provide adequate retrieval results .

Color Features

When it comes to extracting visual features from images, color is the most straightforward method, making it one of the most widely used features in content-based image retrieval because of its simplicity, which in turn helps It to reduce the computational cost of the retrieval algorithm making it able to run very fast compared to other methods. Another reason for its popularity is that its relatively robust to background complication and is independent of image size and orientation.

As shown in Figure II.3, where a bunch of pictures representing a sunset scenery were retrieved form a database using color features only; color can be a very good source of information for retrieval purposes in some cases and can hold semantic significance.

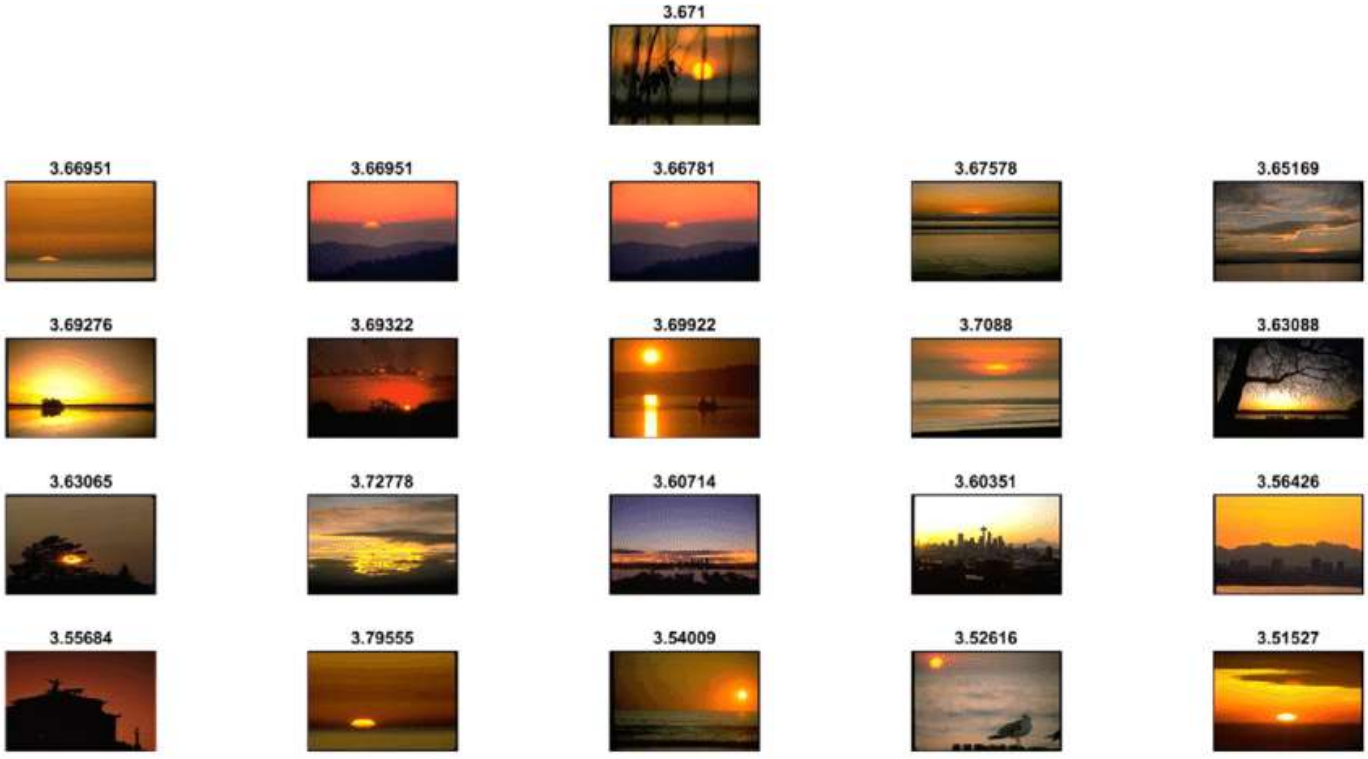


Figure II.3 An example of a color based CBIR [19]

Some of the methods for feature extraction that are color based include the very famous color histograms [20] which essentially is a way to approximate the joint probability of the values of the three color channels creating an approximate statistical model for the image, it's both computationally efficient to compute as well as insensitive to small perturbations in camera position, but it's main drawback is that it does not take into consideration the spatial information of pixels; Another work done by Stricker et al. [21] tries to improve on the previous method by using a cumulative color histogram instead of the conventional histogram which successfully increased the robustness of the retrieval process.

Another very efficient way to estimate the color distributions of images and is an improvement to the previous method are color moments, firstly proposed by Stricker et al. [21] where he used the first three moments (mean, variance, skewness) instead of the whole color distribution which meant that only the dominant features get stored, successfully bringing down memory usage and compute power needed to use this method, but it's obviously also suffers from the same disadvantage of not being able to encode any spatial information whatsoever; this was later fixed by the work of Huang et al. [22] with their method called color correlograms which incorporates spatial data in the encoded color information and therefore it can be used to describe the global distribution of local spatial correlation of colors while still maintaining low compute requirements. After that came the work done by Pass et al. [23] entitled color coherence vector (CCV) which partitions pixels within the image into two types, either coherent or incoherent according to whether it is part of a larger region of uniform color, separate histograms can then be produced for both coherent and incoherent pixels thereby including some spatial information in the feature vector.

Texture features

Texture is considered as the set of visual patterns present in an image; these patterns are characterized by properties of homogeneity that may not necessarily result from the presence of only a single color, It is a natural property of virtually all surfaces, including clouds, trees, bricks, hair, and fabrics. It contains important information about the structural arrangement of surfaces and their relationship to their surrounding environment; Figure II.4 shows a few examples of textures.



Figure II.4 Examples of images containing common textures

Texture representation methods can be classified into three categories [24], the first being statistical techniques that mostly exploits the statistical properties of the gray levels of pixels in order to deduce the periodic occurrence of certain gray levels. Second are structural techniques which starts off with the proposition that any image is composed of a number of small texture elements called texels, and these texels are arranged regularly on a surface according to some specific arrangement rules. And finally, there are spectral techniques which are based on Fourier transform or other similar transforms, which essentially serve as a periodicity detector in an image.

Some of the feature extraction research that is texture based include Tamura Features [25] which were based on some very profound psychological studies research that was focused on the subject of visual perception of textures by humans; they successfully figured out six important visual properties in textures, these properties are coarseness, contrast, directionality, linelikeness, regularity and roughness. After that came the work of R.M.Haralick with his features given later on the name of co-occurrence matrices [26], these features were constructed on a basis of orientation and distance between the pixels, this work was later enhanced by Gotlieb and Kreyszig who found out that the most important properties are contrast, inverse diference moment, and entropy [27].

Another very important work which influenced many contemporary computer vision techniques and is said to be similar to how the human visual system perceives textures are Gabor filters [28], named after Dennis Gabor, it basically analyzes whether there is any frequency content

in the image in a specific direction in a localized region around the point or region of analysis; these filters have been successfully employed in many other applications including image coding, retina identification, document analysis, edge detection, and others.

And finally, another very popular and much used technique in many other image processing fields are Wavelet transforms [29], but unfortunately, they do not achieve high level of retrieval accuracy on their own, however if these wavelet features were assigned different weights, it drastically increases their effectiveness in CBIR [30].

Shape features

Another seemingly simple and primitive type of features are shape features, these are supposed to describe the geometrical information of an object in an image even when orientation or location of the object are changed, these were motivated by the fact that many objects can be very easy to identify when looking at just the shape of them, for example airplanes, roses, cups and many other objects have a very distinguishable and reorganizable shape.

Shape representations can be divided into two categories, either boundary-based which uses only the outer boundary of the shape, or region-based which uses the entire shape region as shown in Figure II.5.

The most successful representations for these two categories are Fourier descriptors and moment variants; The main idea of Fourier Descriptor is to use the Fourier transformed boundary as the shape feature [31]; whereas moment invariants is to use region-based moments, which are invariant to transformations as the shape feature. Hu et al. [32] identified seven such moments.

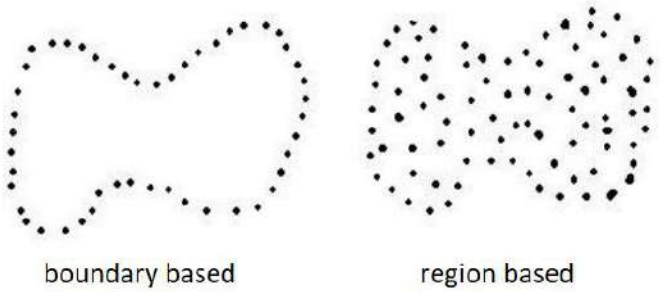


Figure II.5 The difference between boundary based and region-based representations [24]

2.1.2.2 Similarity Measure

After the feature extraction part where both the query image and the database images are transformed into their alternative representation which constitutes only the essential information (feature vectors), comes the task of similarity measures to measure the distance between these feature vectors in order to evaluate the similarity between a query image and a certain database image according to their features only, it is the final step in order to retrieve images from a database that are similar to the query image specified by a user which is the objective of CBIR systems.

The choice for a particular measure can affect significantly the retrieval performance depending on their characteristics and the particular needs of the retrieval application. Some of the commonly used measures include the most basic and most common Euclidean distance measure which is appropriate to use when the feature vector elements are equally important and are independent of one another, it is very simple to implement with a low level of compute complexity; it's uses in commercial systems include the Multimedia Analysis and Retrieval System (MARS) [33] which uses this metric to compare between texture features in a CBIR system. Another popular metric is the Manhattan distance which unlike the Euclidean distance that represents the length of the straight line between two points, this latter one represents the length of the two sides of a square approach. It is this that gives the technique its name since Manhattan is laid out in city blocks forcing you to walk 2 sides of a square in order to get anywhere.

2.2 Automatic Image Annotation

A very important problem that computer vision folks are investing a lot of interest in over the last two decades is AIA, this particular field came as a consequence of the growing need to annotate very large image databases as it is very time consuming and impractical to annotate them manually, this is being mostly done in order to use it later on as textual metadata for the task of image retrieval, which makes the two fields closely related in principle as they both share the use of pixel level content of the image and complement each other nicely, add to this that many techniques were taken from CBIR and then improved on especially when it comes to the feature extraction part; However, many researchers believe that there is a semantic gap between CBIR and image semantics understandable by humans. As a result, research in this area has shifted to bridge the semantic gap between low level image features and high-level semantics, The typical method of bridging the semantic gap is through the AIA process.

AIA aims to assign several tags in a limited vocabulary to describe an unlabeled image. This is done by automatically learning semantic concept models from a large number of image samples, and then using the learned models to label new images; basically, the process starts by extracting features from the training and testing images and then generating the annotation model according to training data, and finally, we can generate annotations according to the features of testing images, the detailed process is indicated in Figure II.6.

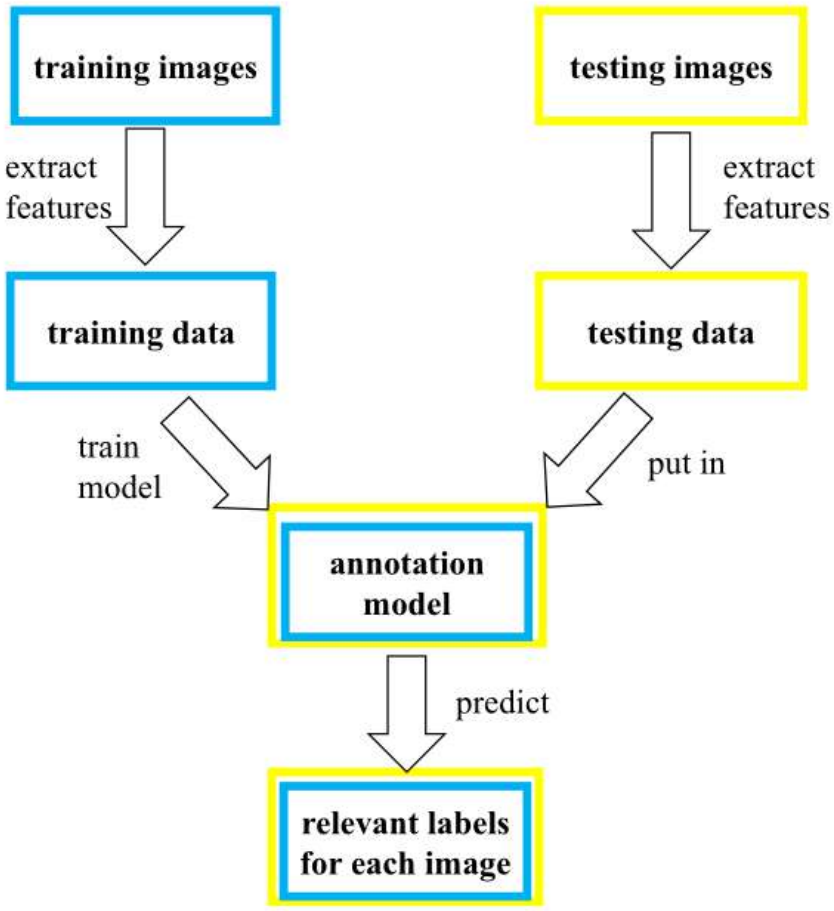


Figure II.6 A general architecture of an Automatic Image Annotation model [34]

At the beginning, the research direction of image annotation was mostly directed towards image classification, which is equivalent to labeling each image just by one tag, hence image classification can be deemed as single-label annotation, making the problem at hand way simpler and easier to solve. Because images usually contain multiple objects, a single label cannot describe the picture in a comprehensive and complete way, however it's a good starting point for research.

Consequently, numerous studies have been conducted on AIA, particularly on the classification-based and probabilistic modeling techniques.

In this paper, we focus on this latest development in AIA. We analyze key aspects of the various AIA methods, including both feature extraction and semantic learning methods. Major methods are discussed and illustrated in detail.

2.2.1 Feature extraction

Just like in CBIR, images need to be firstly converted into a low-level compact representation that englobes all the semantics of the image in order to use it later on for semantic learning which is an essential step for the task of AIA, hence, many features that were firstly used with CBIR were re-used for AIA since they brought good results nonetheless and were a good starting point for researchers; however, we will focus only on new feature extraction techniques that were exclusive or firstly employed for the task of AIA to avoid repetitiveness.

2.2.1.1 Local Features (features based on image segmentation)

During the early days and up until now, the trend was directed towards region-based feature extraction techniques, which means that images needed prior segmentation before being passed to the feature extraction part as pointed out by Zhang et al. [6], this meant that unlike global features which are directly calculated from the whole image, local features are instead calculated from different components of the image, which are the product of dividing the image into partitions based on feature homogeneity, however, it's important to note that the biggest disadvantage of these methods is that the unsupervised segmentation associated with local features may affect their accuracy.

We start off with one of the simplest and most straight forward type of approaches which is grid-based methods, these essentially aim to roughly segment images into blocks; its advantage is that it requires very little compute but its naïve simplicity fails to describe the semantic components in images very well; of the many works that used this method is the work by Cusano et al. [35] where they used local features to annotate images based on seven scenery related classes; another one is the work by Feng et al. [36] where they used a Bernoulli probabilistic model to map the feature vectors to the annotations keywords.

Another segmentation algorithm is by Wang et al. [37] where the image is firstly portioned into 4x4 pixels blocks, and then color and/or texture features are extracted for each block, and finally a clustering algorithm such as k-means is used to cluster the pixels into different groups, where these groups go on to form regions based on the pixels belonging to blocks of the same cluster. The obvious disadvantage of this method is that k is hyperparameter that requires predefining, which means that inappropriate choice of k may yield poor results, the other issue is that the algorithm assumes data are in spherical clusters which isn't always the case.

2.2.1.2 Color features

Some feature representations seem to have a magic like behavior that makes it quite universal and very flexible, making it appliable in many fields of computer vision be it image retrieval, image annotation, and even compression and encryption; of these color-based feature extraction methods that showed great success for the task of AIA is the MPEG-7 representation; this method does not contain only a single type of feature, but it describes and standardizes a whole set of color-based features, these include the scalable color descriptor (SCD) [38], which is basically a histograms but in the Hue-Saturation-Value (HSV) color space, it's advantage is its scalability where the number of color bins are reduced using Haar transform, as well as the LSB bits of the quantized integer representation of bin values are removed; but since this method closely resembles a conventional histogram, it lacks the special information of objects in an image.

Another feature described in the MPEG-7 standard is the Color structure descriptor (CSD) [39] which is also a histogram-based feature, its idea is that instead of using single pixel to calculate the frequency of a certain color, a moving structuring element which could be a square or another

shape is swept across the whole image, and in doing so every time the presence of a certain color checked if present at least in a single pixel inside of the structuring element and the color bins are filled accordingly, it's important to note that when the size of the square is one pixel, this method reduces to a simple histogram.

And finally, is The Dominant color descriptor (DCD) [40] which is also a histogram-based feature, it's basically a compact histogram where only the highest bins (colors of highest frequency in an image) are taken with the help of thresholding; this number of bins is recommended by the MPEG-7 standard as 1–8 colors are sufficient to represent a region, however, the biggest disadvantage is how to measure the distance between two DCD features.

2.2.1.3 Texture features

Of the texture-based feature extraction methods that were used for AIA, the methods of most importance are probabilistic model-based techniques, where texture is interpreted as a stochastic generative model, making the model parameters characterize the underlying textual properties of the image; popular texture models include the Markov random field (MRF)[41], which essentially describes the overall macrotextures such as grass, sand, cork, ceiling tile, and paper which all obeyed the MRF model quite beautifully, they showed how the parameters of the model controls the strength and direction of the clustering in the image. and also showed how even the parameters of the model themselves were sufficient to generate new images that contain a certain texture; another work is the simultaneous auto-regressive model (SAR) [42], which is a kind of two dimensional noncausal random field model, where In its general form, it resembles a conditional autoregressive (CAR) model and can be made equivalent but is often parameterized differently, this model can be used for processes with directional graphical properties making texture feature extraction a great candidate for this method, its advantage is that it possess strong discriminability power for texture, but the main problem affecting the application of the SAR model is the process of parameter estimation, most commonly SAR models have been used in applications involving ecology, epidemiology, sociology, and environmental sciences.

Another very important method is the work by Chaudhuri et al. [43], where they used the technique of fractal dimensions in order to segment images based on texture; This choice was

motivated by the observation that the fractal dimension (FD) is relatively insensitive to image scaling, and also shows a strong correlation with human judgment of surface roughness, add to this that an FD is related to the linear log power spectrum making it into is an approximative spectral estimation method. their features were based on the fractal geometry of images, where they used six FD features, the above average/high gray level image, the below average/low gray level image, the horizontally smoothed image, the vertically smoothed image, and the multi-fractal dimension of order two; this was all estimated by using modified box-counting approach.

And finally, as an improvement of classical spectral methods such as Gabor filters and wavelets, research on multi-resolution analysis have shown that curvelet features [44] may have a significant advantage, because curvelet features are more effective in capturing curvilinear properties such as lines and edges; Curvelet transform was originally proposed for image denoising but since it has been shown promising performance, it was then used for texture feature extraction; the main disadvantage with this method as with all spectral methods is that they can only be applied to square regions due to the use of FFT, but recently Islam et al. [45] proposed a texture padding method to transform an irregular texture region to a square texture region solving this problem.

2.2.1.4 Shape features

Most of the shape feature extraction methods was carried out from previous CBIR methods and reused for AIA, these methods were mostly constrained to a certain domain because shape features are not very good when it comes to generalizing, for example [39] have used MPEG-7's contour shape descriptor for animal ontology, where the visual shape features are mapped to semi-concepts via their mapping algorithm. Another work is by Liu et al. [46] where they used a shape ontology framework for the task of bird classification.

2.2.1.5 Deep learning for Automatic image annotation

During this decade, and inspired by the cranial nerve theory, Deep Learning has started to become widely used in the area of computer vision. This enabled a variety of deep models for large-scale image annotation. Where feature hierarchies are learned so that features from the higher hierarchy are formed by compiling features from the subordinate hierarchy, this will help the system directly map the input to the output without depending on the human-crafted features that

may not be the best considering the time-consuming nature of it and the much effort needed to invent a new type of feature, add to this the fact that these features may only work in a particular domain. This is all solved with using deep learning.

The deep-learning based AIA can be summarized in two aspects. Firstly, robust visual features are generated by using convolution neural network (CNN). Secondly, side information (such as semantic label relationships) is fully extracted through deep learning techniques for AIA [47].

For the feature generation part, Gong et al. [48] made use of five convolutional layers and three densely connected layers in a CNN architecture for the task of multi label annotation. they investigated several different ranking-based loss functions for training the CNN model and successfully demonstrated the effectiveness of using top-k ranking which is essentially a weighted approximate ranking loss function, and tested it on the widely known NUS-WIDE dataset [49]. They also employed a set of 9 different visual features (GIST, D-SIFT, D-CSIFT, D-RGBSIFT, H-SIFT, H-CSIFT, H-RGBSIFT, HOG and Color feature) combining them in order to serve as a feature for comparison with their CNN based method, and their experiments show that the deep learning method had better results that all the 9 handcrafted features combined.

Similarly, Mayhew et al [50] trained two different image annotation algorithms which are Tagprop [51] and 2PKNN [52] using features that are derived from two widely known CNN architectures (AlexNet [53] and VGG-16 [54]), and they also found that by using deep learning-based features instead of handcrafted ones, we can achieve far better results. Moreover, their study proved the idea that using both the deep and handcrafted features could be jointly used to enhance the predictive performance.

Another work is by Murthy et al. [55] where they propose features extracted from a CNN and word embedding vectors to represent their associated tags, these are then mapped using three different types of models, which are all variants of the Canonical Correlation Analysis (CCA) model, namely linear CCA, kernel CCA and CCA with k-nearest neighbor (CCA-KNN) clustering. They have found that CCA-KNN outperforms previous methods on the Corel-5k [56] and the ESP-Game [57] datasets and achieves better accuracy overall.

And for the side information part, Wang et al. [58] utilizes recurrent neural networks (RNN) to capture high-order label relationships at a moderate level of computational complexity, this establishes a hierarchical system in which the correlation between the adjacent labels is found automatically which eliminates label imbalances, this is all being done in a joint CNN-RNN framework where the orders of the labels are determined according to their occurrence frequencies in the training data.

Another work that also uses the CNN-RNN framework is the RIA model [59] which uses CNN to extract image visual features, and RNN to generate the tag sequence from the visual features one by one, the advantages of RNN in AIA lie in two aspects. On the one hand, RNN can generate output with different length which is a huge advantage compared to previous methods which evaluate their performance using top-k retrieval performance, where k is fixed, where instead the number of annotated tags should depend on image contents and not on a fixed number. On the other hand, RNN is able to refer to previous inputs in predicting the current time step output [47] .

And finally, [60] used a structured inference neural network (SINN) which mainly is used to deal with layered label predictions. The main idea of SINN is that an image with various objects and abundant attributes can be assigned with fine-grained labels that describe major components and rough-grained label that depict high level abstraction. This is being exploited using a state-of-art deep learning framework, where they propose a generic structured model that leverages diverse label relations in order to improve image classification performance.

3 Conclusion

We have in this chapter, discussed briefly about published matter that technically relates to our proposed work; we have mostly ordered the related works in a historical manner while also mentioning the developments that led to the emergence of our domain of interest (AIA) in order to know exactly how this domain started and how it grew over the years, this was done by firstly talking about the classical image retrieval methods such as TBIR and CBIR, with a particular focus on feature extraction, and then the AIA methods were discussed and compared with their previous counterparts.

Chapter III Research methodology

1 Introduction

As mentioned during the first chapter, the main objective of this dissertation is the application of Deep Learning techniques to an Automatic Image Annotation problem. The main research hypothesis of this work is:

Deep Learning classification methods are able to automatically produce class specific feature vectors that will help in the AIA task.

In order to test this hypothesis, three different steps are used. First, a typical Deep learning algorithm such as Convolutional Neural Networks are used in an image classification task. Then, the feature vectors will be extracted from the trained CNN model and then compressed using an Autoencoder model, and finally, these feature vectors will be used for the task of tagging new test images using a similarity measure method.

2 Research Strategy

In Figure III.1, the research methodology used in this dissertation is shown. After the first three steps, there is an iterative process including researching feasible approaches, implementing said approach, and evaluating the implementation to conclude the performance of the model and if the approach was suitable for the problem.

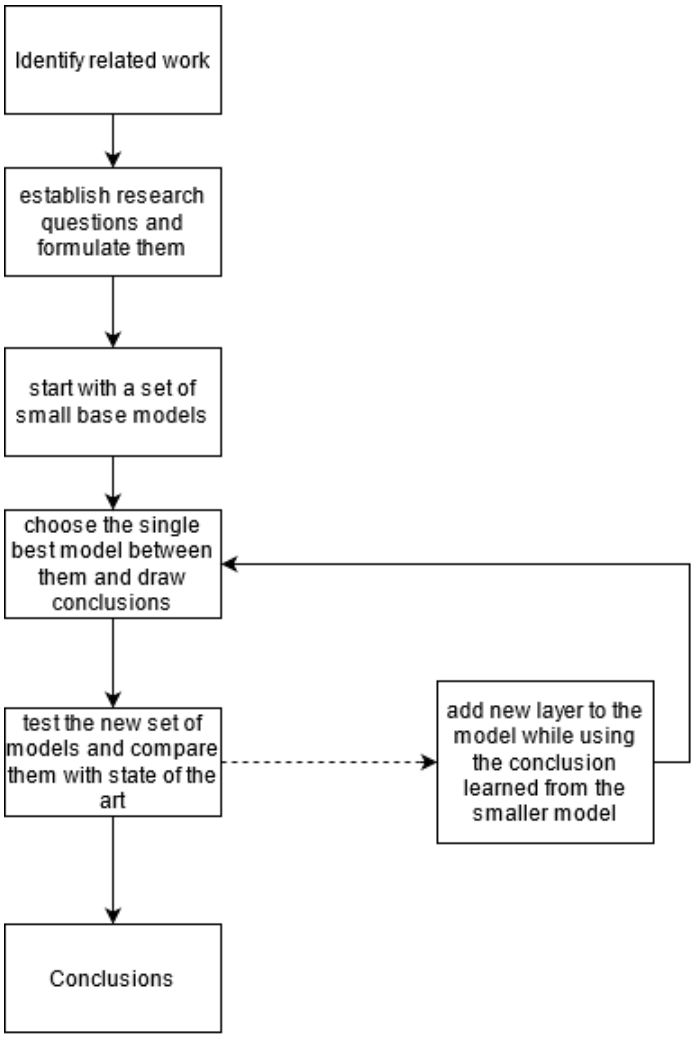


Figure III.1 The methodology used for research

3 Dataset Used

As explained in Chapter I, the main purpose of our work is to automatically annotate images based on objects present there. In order to achieve this task, we opted for a deep learning-based method, and since we're using deep learning, it is imperative that any machine learning task and especially deep learning should be based on choosing the right dataset to achieve the required results. The size of the dataset should follow the size and architecture of the deep learning mode, since the bigger the model the more data is needed to achieve good results.

The motivation for using Microsoft's common object in context (MS-COCO) dataset [61] is that it's a very large image dataset that contains images from everyday scenes that contain

common objects in their natural context, as well as the fact that it's widely used as a benchmark for the task of AIA which helps us evaluate our own contribution. it contains photos of 80 objects types that would be easily recognizable by a 4-year-old, all of these are also grouped into 11 super categories that group related categories together. With a total of 2.5 million labeled instances in 328k images. These object types are illustrated in Figure III.2.

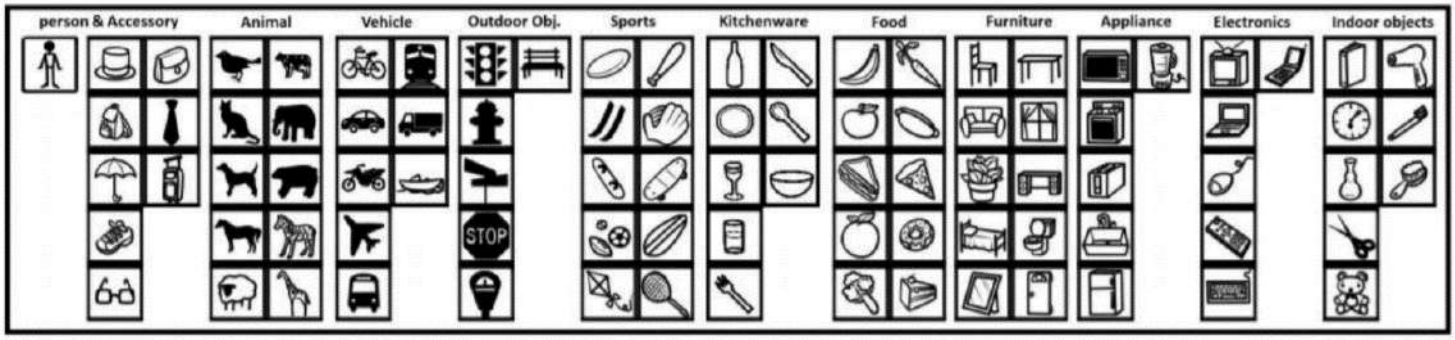


Figure III.2 The different categories of the COCO dataset [62]

The ground truth data generated by humans is available for all of the images in the training and validation dataset in the form of an annotation file, this annotation file groups several information that can be used for the task of object detection, keypoint detection, stuff segmentation, panoptic segmentation, densepose, and image captioning. The annotations are stored using the widely used JSON file format (JavaScript Object Notation), it contains five sections which are info, licenses, images, annotations, and categories. But the most relevant sections which we'll be using are annotations, and categories.

The annotations section for each image is provided in a per-object basis, and each object has an annotated data structure which provides the following information:

1. Which image that object belongs to in the dataset.

2. The object's class.

3. A segmentation mask for that particular object, encoded into a custom RLE

4. The bounding box for the object.

5. Several other types of information irrelevant to the present study.

An example of some sample images from the dataset with their respective categories and their class mask overlayed on the objects is presented in Figure III.3.

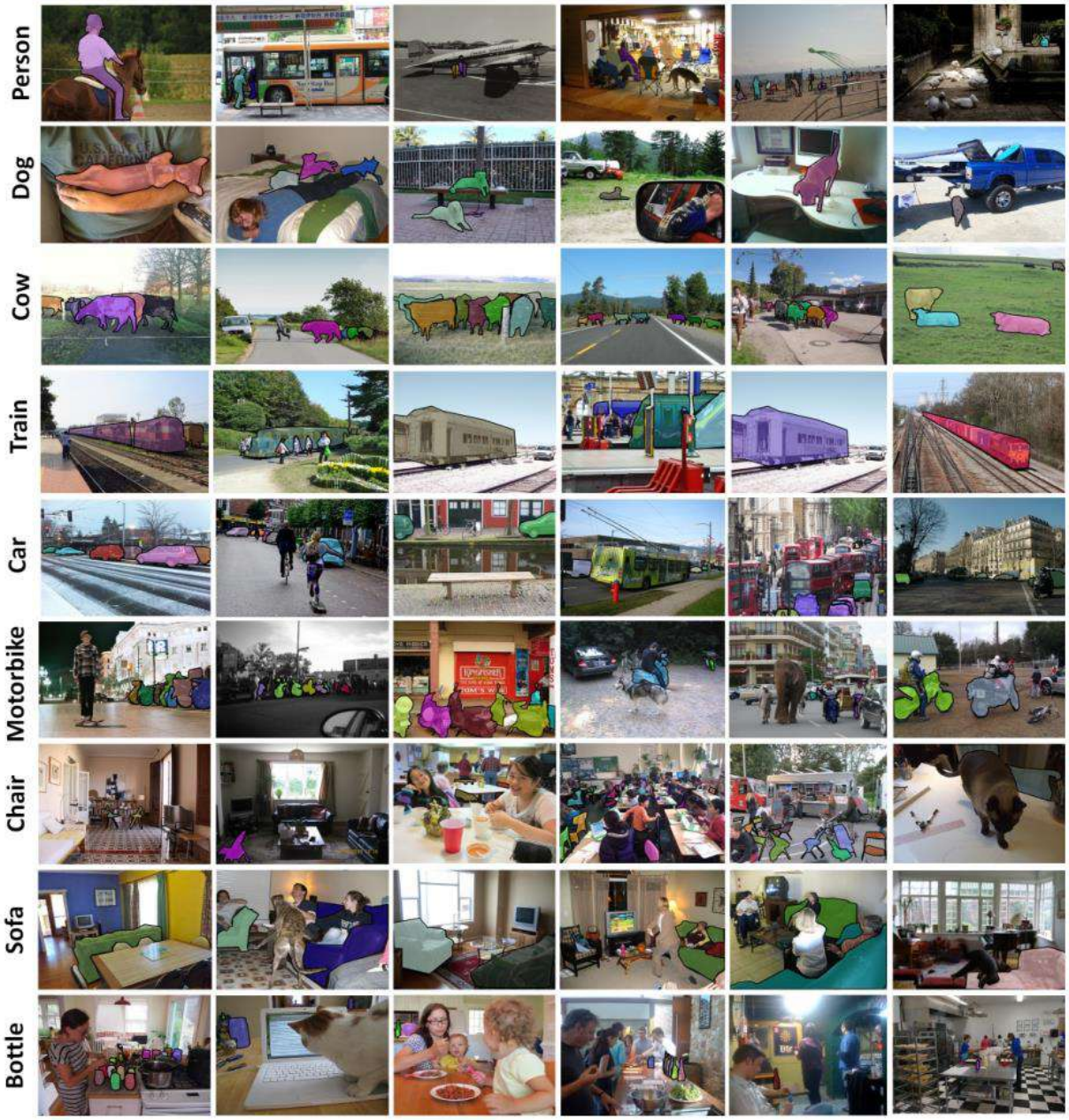


Figure III.3 A sample of the COCO image Dataset [61]

The team behind the MS-COCO dataset also provided us with a very helpful tool that makes it easier to extract the required information from the annotation file in order to use in any of the computer vision tasks, this tool is called the COCO API [63], it's essentially a software library available for Python, MATLAB, and Lua programming languages, it contains all of the functions that enable us to extract the segmentation masks as well as the bounding boxes which we'll be using later on, as well as other functionality which assists in loading, parsing, and visualizing the annotations in COCO. For example, a picture like in Figure III.4 containing the segmentation mask for a specific class object (human) and their respective pose data can be produced using the COCO API with only a minimal of lines of code, compared if we had to load in these annotations manually.



Figure III.4 An example of a COCO image with its respective annotation

The COCO database is huge with its 1.5 million labeled object instances and 80 categories, so I had to choose a small subset of it for this study, a subset of 30 classes was chosen based on the number of images available after the operation of cropping, segmenting, and masking; at the end we ended up with a total number of 69122 images for training and 28574 for validation/testing, making the ratio of train/test be around 70% train and 30% test.

These 30 classes are presented in Table III.1.

Table III.1 The subset of COCO classes that were used in our work

Categories	Number of training examples	Categories	Number of training examples
person	2500	giraffe	2500
dining table	2500	cake	2500
chair	2500	elephant	2500
bowl	2500	motorcycle	2500
car	2500	toilet	1977
cat	2500	laptop	1956
bus	2500	airplane	1893
cup	2500	broccoli	1858
pizza	2500	sandwich	1841
dog	2500	zebra	1821
truck	2500	banana	1791
couch	2500	horse	1783
tv	2500	teddy bear	1765
bed	2500	umbrella	1686
train	2500	cow	1566

The way this subset was chosen was done by firstly filtering objects using the flags and information available in the annotation file, where the following requirements must apply to them:

1- The bounding box surrounding the object should be bigger than 64x64 pixel.
2- The total area of the bounding box should surpass 3000.
3- The objects should not be a crowd object but a singular object (iscrowd flag must equal 0).
4- The object should not be split into two or more parts but a single mask should cover it (segmentation RLE vector should be a single vector and not a matrix).

This was done in order to obtain the clearest looking objects from the original dataset, meaning the objects should not be too small to the point where they could not be recognizable anymore.

A little pre-processing was required on these objects before it could be trained because of the way the training model works. Firstly, the segmentation mask was used to retain only the pixels which belong to the object itself, the rest was given a black color (RGB= [0,0,0]), and after that the images were cropped using their respective bounding boxes. And then finally, all the images

were resized to 64x64 pixels in order to make all the objects uniform in size. Figure III.5 shows the procedure in detail.

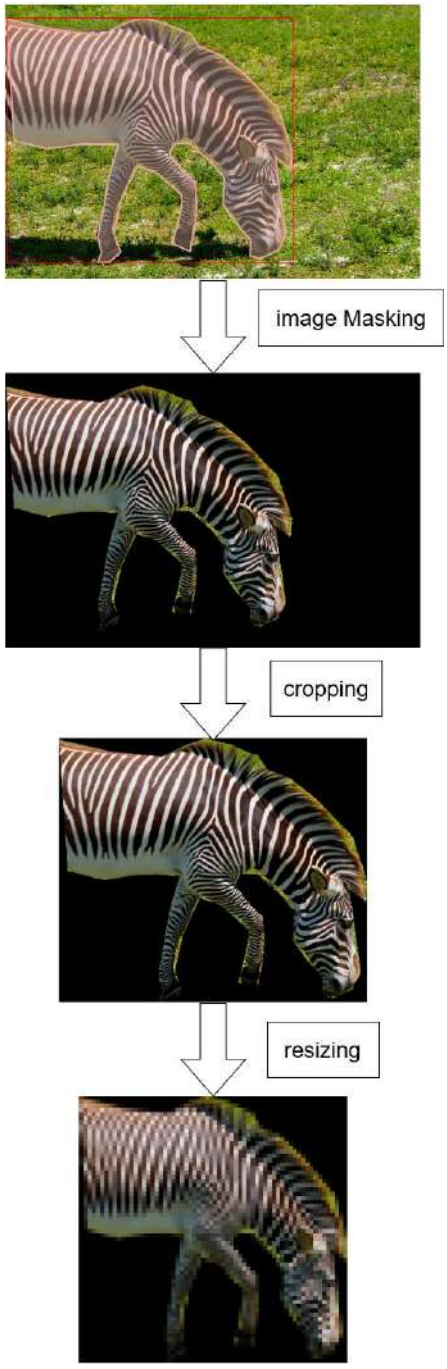


Figure III.5 The process of transforming the raw dataset images into a useful format

A sample of the newly created dataset which contain only a singular object of interest in one image is presented in Figure III.6.

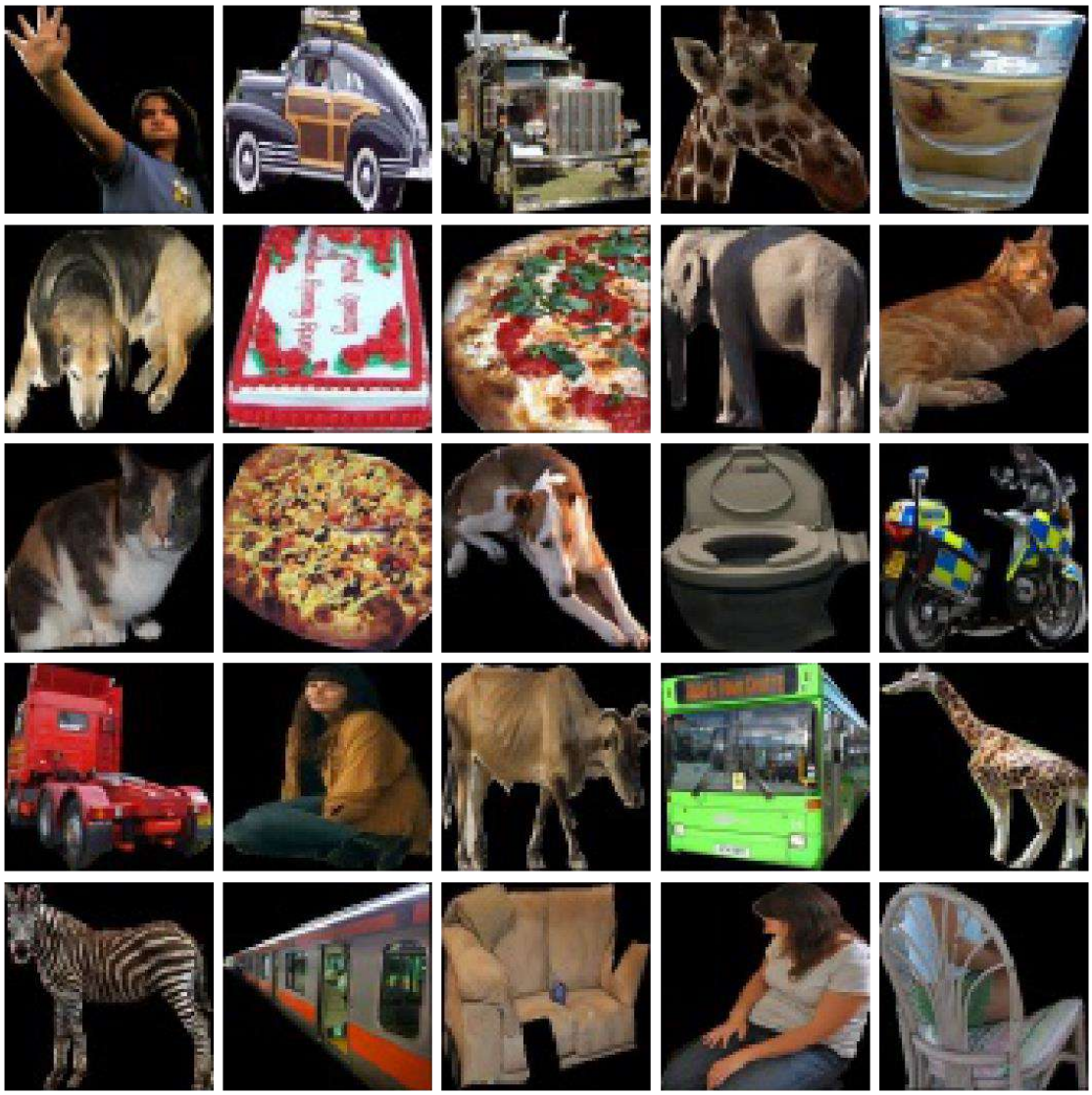


Figure III.6 A sample of the newly created modified COCO Dataset

4 Deep learning Model

Any AIA task requires firstly extracting the feature vectors from an image, which in turn are then used by the machine learning algorithm which tries to link them to the training annotation in order to automatically apply annotation to a test image; our method firstly extracts the feature vectors using a CNN based classification algorithm, and then these vectors are compressed using an autoencoder, and finally, they are used for the annotation task.

4.1 Feature extraction

4.1.1 CNN classification model

Convolutional neural networks are very efficient when it comes to extracting meaningful features from images with hardly using any pre-processing compared to other image classification algorithms, this means that the network learns to optimize its kernels through automated learning, whereas in traditional algorithms these filters are hand-engineered; we use this characteristic of CNN models to our advantage by firstly giving the model a normal image classification task, and then the model will find the best features which best discriminate (separate) between object types all by it itself, and then we will extract these features from one of the hidden layers of our trained CNN model.

In order to achieve this task, we need to firstly learn how a typical CNN model works and how its architecture is built.

4.1.1.1 Prerequisites

A convolutional neural network (CNN), is a deep neural network architecture that is best suitable for visual imagery; similar to regular neural networks, they consist of an input layer, multiple hidden layers, and an output layer. These hidden layers however are different from typical MLP layers in that they contain what is called convolution kernels instead of a normal linear weights layer which only does a simple linear combination of the layer weight matrix with the input matrix, on the other hand these conv kernels are essentially a stack of filters that slide along input features in order to produce what is called feature maps (analogous to activations in a normal MLP network).

These CNNs are known to take advantage of the hierarchical patterns present in data and assemble patterns of increasing complexity as we go deeper into the network, using smaller and simpler patterns embossed in their filters. Therefore, on a scale of connectivity and complexity, CNNs are on the lower extreme making them one of the most efficient and universal machine learning algorithms [64].

Figure III.7 shows a typical CNN architecture where the hidden layers consist of convolutional layers, pooling layers (sub-sampling), and fully-connected layers.

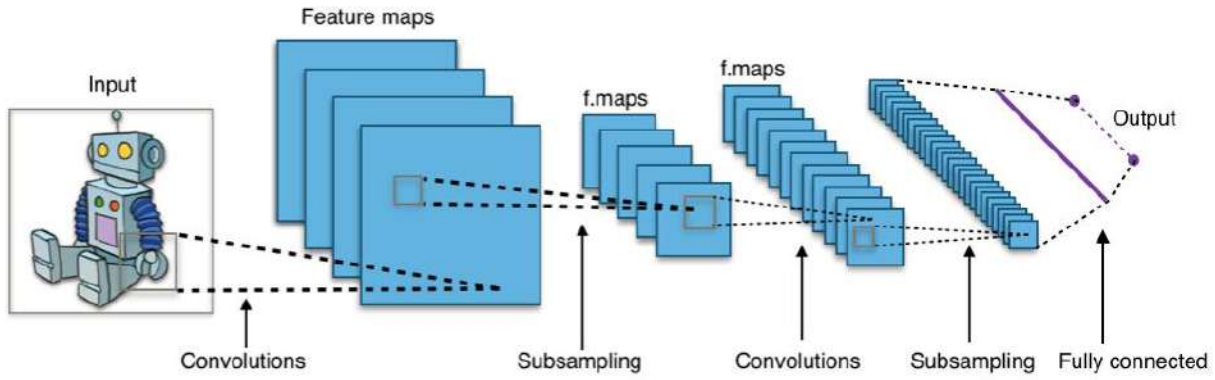


Figure III.7 A typical CNN architecture [64]

In the next section we will explore the purpose of each of these layers.

4.1.1.1.1 Convolutional layers

The convolution layer is the core building block of a CNN. It consists of a set of filters. A filter can be seen as a smaller window that convolves (slides) across the input image and computes dot products between the values of the filter and the pixel values of the input image. This is similar to the response of a neuron in the visual cortex to a specific stimulus. The output of this action is a 2-dimensional activation map that shows the responses of the filter at each position. Each convolutional layer has a set of filters, and each filter will produce a separate 2-dimensional map. These separate activation maps are stacked to produce the output of the convolutional layer which is called a feature map.

Its main advantages is that it reduces drastically the number of free parameters in a network compared to a fully connected layer for example, allowing the network to be way deeper and more computationally efficient.

the width and height of the filters are hyper-parameters, while the depth of the filters must be equal to the depth of the input image. Other hyper parameters that are to be considered, which all will affect the size of the output image, are:

- **Number of filters**: each learning to look for something different in the input. then different neurons along the depth dimension may activate in presence of various oriented edges, or blobs of color. This will also determine the depth of output feature map.
- **Stride**: The stride parameter specifies how many steps at the time the filter will move across the image. When the stride is set to 2 or more, the filter would jump 2 or more pixels at a time which would produce an output with smaller width and height than the input.
- **Zero-padding**: pads the input image with zeros around its border. Most commonly, zero-padding is used to preserve the size of the input size to the output.

Figure III.8 shows an example of a convolution operation between an input matrix and a kernel.

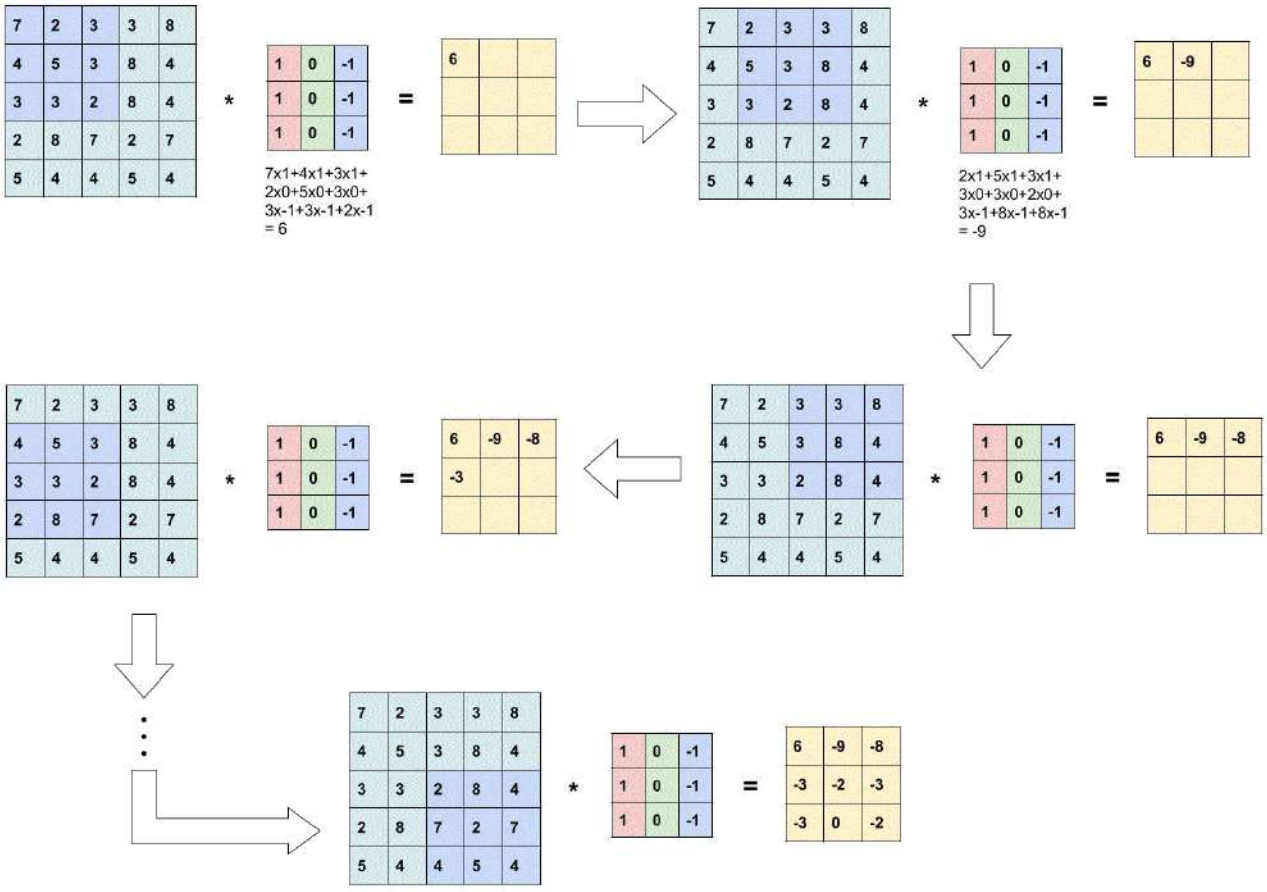


Figure III.8 An example of a convolution operation between an input matrix and a kernel [65]

The following formula shows how to calculate a single element with indices [i, j] of the matrix G which is a resultant of the convolution between two matrices F and W.

$$G[i,j] = \sum_{k=-\infty}^{\infty} \sum_{k=-\infty}^{\infty} F[i-k, j-l]\, W[k,l]$$

Considering the input feature map shape to be (**h** x **w** x **c**), after the convolution operation with a kernel of shape ($\mathbf{k_h}$ x $\mathbf{k_w}$) and stride of ($\mathbf{S_h}$ x $\mathbf{S_w}$) with padding **P**, the dimensions of the output feature map become:

$$\text{Output height} = \frac{\mathbf{h} + 2\mathbf{P} - \boldsymbol{k_h}}{\boldsymbol{s_h}} + 1$$

$$\text{Output height} = \frac{\mathbf{w} + 2\mathbf{P} - \boldsymbol{k_w}}{\boldsymbol{s_w}} + 1$$

while the number of channels stays the same.

4.1.1.1.2 Activation layers

The activation function is a node that is put at the end of or in between Neural Networks; its name comes from biological equivalent "action potential" which is called sometimes activation protentional, because just like the biological version that represents a stimulation threshold that when reached it entices a neuron response, most of the artificial activation functions used also fire when a certain threshold is reached, making their response nonlinear, which is a huge advantage since it allows the network to model the behavior of a complex nonlinear function as is proven in the universal approximation theorem [66].

The main characteristics of an activation layer is that it produces an output that has the same dimensions as its input feature map since the activation function is applied in an element-wise manner, and also no parameters/weights are learned inside of it which helps bring down memory usage and the number of parameters to optimize for the machine learning task.

There are numerous activation functions commonly used today, the most popular of them are the logistic sigmoid activation function (sigmoid), the SoftMax function, as well as the ReLU activation function. The simplest of them is the ReLU function, which is a special implementation that combines both non-linearity and rectification in convolutional neural networks.

A rectified linear unit (ReLU) (i.e., thresholding at zero) is a piecewise linear function defined as:

$$f(x) = \max(0, x)$$

where *x* is the input to a neuron. This is also known as a ramp function and is analogous to half-wave rectification in electrical engineering. Its main advantages include having better gradient

propagation compared to sigmoidal activation functions that saturate in both directions, and it's very efficient to compute: Only comparison, addition and multiplication.

Figure III.9 shows an example of an input feature map going through a ReLU activation.

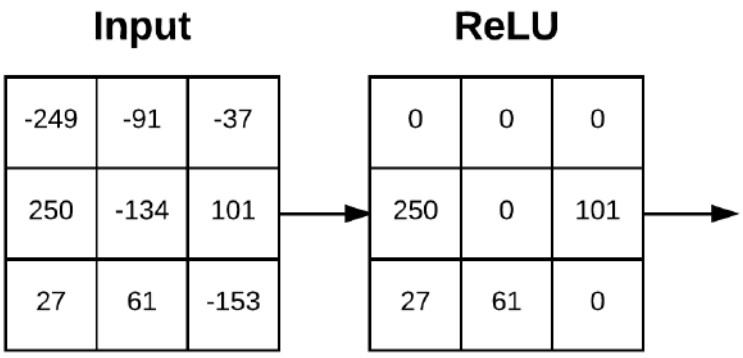


Figure III.9 An example of an input matrix after the ReLU operation [67]

4.1.1.1.3 Pooling layers

The purpose of the pooling layer is to progressively reduce the number the dimensions of the feature maps and thus reducing the number of parameters to learn which in turn reduces the amount of computation in the network, and hence also controls overfitting. it's one of the best approaches for down sampling the feature maps efficiently. In all cases, pooling helps to make the representation become approximately invariant to small translations of the input. Invariance to translation means that if we translate the input by a small amount, the values of most of the pooled outputs do not change [68, p. 342].

The most commonly applied type of pooling is max-pooling. For example, a max-pooling layer with a filter size of 2 x 2 and stride 2 applied to the input feature map will discard of 75% of the activations taking only the max over the 4 numbers of the input making only the feature with highest values (most important or sharpest features) go through the rest of network. In most cases a pooling layer of size = 2 x 2 and stride = 2 is used, and in some cases a size = 3 x 3 with stride = 2 is also used, this latter one holds the special name of overlapping pooling. Another common type of pooling is average-pooling, this type instead of taking the max between the input feature map values, it takes the average value. Figure III.10 shows a typical feature map after the operation of max-pooling.

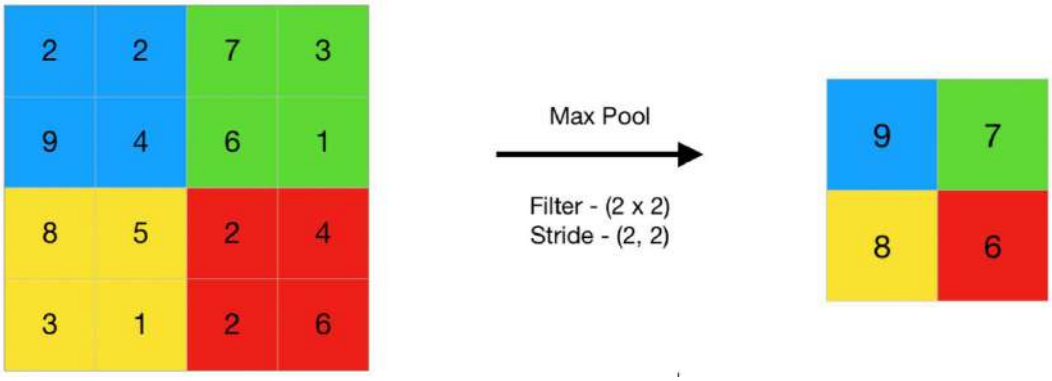


Figure III.10 An example of an input matrix after a Max-Pool operation [69]

It's worth noting that for an input feature map having dimensions **h** × **w** × **c**, the dimensions of output obtained after a pooling layer is:

$$\frac{(\mathbf{h}-\mathbf{f}+\mathbf{1})}{\mathbf{s}} \times \frac{(\mathbf{w}-\mathbf{f}+\mathbf{1})}{\mathbf{s}} \times \mathbf{C}$$

Where,

- -h is the height of the input feature map
- -w is the width of the input feature map
- -c is the number of channels in the input feature map
- -f is the size of pooling filter
- -s is the stride length

4.1.1.1.4 Normalization layers

A normalization layer is used to make artificial neural networks training converge faster and more stable through normalization of the layers inputs by re-centering and re-scaling. The most famously used type of normalizations is Batch Normalization (Batch Norm).

Batch normalization applies a transformation that maintains the output mean close to 0 and the output standard deviation close to 1 which makes the input data follow a standard Gaussian distribution; Ideally, the normalization would be conducted over the entire training set, but it is impractical to use the global information. Thus, normalization is usually restrained to each mini-batch in the training process. For an input vector x of d dimensions, and for every k from 1 to d, we calculate the new vector $\hat{x}$ using the following formula:

$$\widehat{\boldsymbol{x}}^{(\boldsymbol{k})}{}_{\boldsymbol{i}} = \frac{\boldsymbol{x}^{(\boldsymbol{k})}{}_{\boldsymbol{i}} - \boldsymbol{\mu}^{(\boldsymbol{k})}}{\sqrt{\boldsymbol{\sigma}^{(\boldsymbol{k})2} + \boldsymbol{\varepsilon}}}$$

Where, μ is the variance of the input vector, and *σ is its variance.*

4.1.1.1.5 Fully connected layers

Much like a traditional multi-layer perceptron (MLP) architecture, the neurons in the fully connected layers have complete connections to all activations in the previous layer, their activations can hence be computed with a matrix multiplication followed by a bias offset. Usually, this layer is put in the last phase of a CNN.

A fully connected layer usually doesn't need a nonlinear activation layer in between its hidden layers because the non-linearity is already built within the neurons of fully connected layer. Note that a convolutional layer can be replaced by a FC layer with the distinction that these neurons must be connected to a local region in the input image.

One of the worst disadvantages of these layers is that they are responsible for most of the memory usage and the extra compute time required by the network, this is because of the huge number of parameters contained in them, that is why the research direction is going after omitting these layers from a CNN, It was even proven that it is possible to remove the fully connected layers from a CNN while still maintaining good results, by using only the output from the convolutional and pooling layers.

Figure III.11 shows a typical fully connected layer.

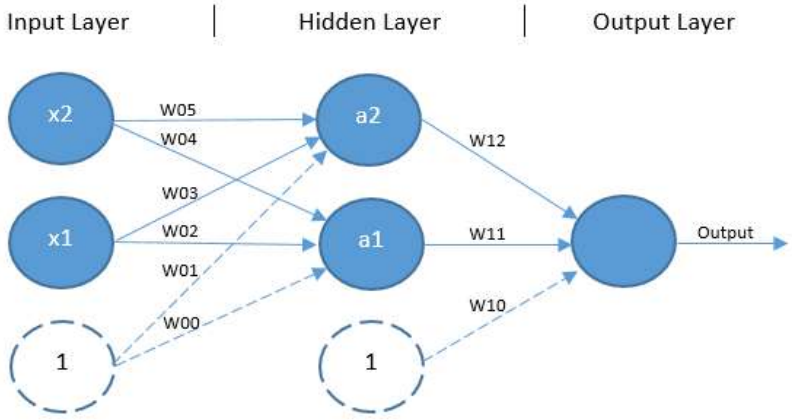


Figure III.11 A typical Fully Connected layer [70]

4.1.1.1.6 Regularization layer (Dropout)

Large neural networks that are trained on a small dataset are prone to overfitting. This has the effect of the model learning the statistical noise in the training data instead of the important features from the data, which results in poor performance when the model is evaluated on new data because of its poor generalization ability.

This can be partly solved by using a regularization layer, and the most famously used of them in accordance with CNNs is the so-called Dropout layer. This layer randomly sets its input units to 0 "*dropped out*" with a frequency set by the probability hyperparameter (called rate) that specifies the probability at which outputs of the layer are dropped out at each step during training time, which helps prevent overfitting. Inputs not set to 0 are scaled up by 1/(1 - rate) such that the sum over all inputs is unchanged. Note that the Dropout layer only applies at training time only, and no values are dropped during inference.

Dropout has the effect of making the training process noisy, forcing nodes within a layer to probabilistically take on more or less responsibility for the inputs. and offers a very computationally cheap and remarkably effective regularization method to reduce overfitting and improve generalization error in deep neural networks of all kinds [71].

4.1.1.1.7 Loss Layer

The loss layer is usually the last layer of the neural network. The purpose of this layer is to specify the penalty that is attributed to the error between the prediction and the true label (target) during training. This layer contains a function that is often referred to as a cost function or a loss function and the value calculated by this function is referred to as simply "*loss*."

For classification, usually the SoftMax function is used. The SoftMax layer takes the previous layer as input, and produces a probability distribution of K probabilities as output, where K is the number of classes during training. The following formula shows how the probability of the j'th class is calculated given a sample vector x and a weighting vector w:

$$\mathcal{P}(y = j \mid x) = \frac{e^{x^T w_j}}{\sum_{k=1}^{K} e^{x^T w_k}}$$

Another famous Loss function is the Cross-Entropy Loss, also called log loss; this function is used mainly for multi-label classification, where it penalizes both types of errors, but especially

those predictions that are confident and wrong; Cross-entropy loss increases as the predicted probability diverges from the actual label as is shown in the following Figure III.12.

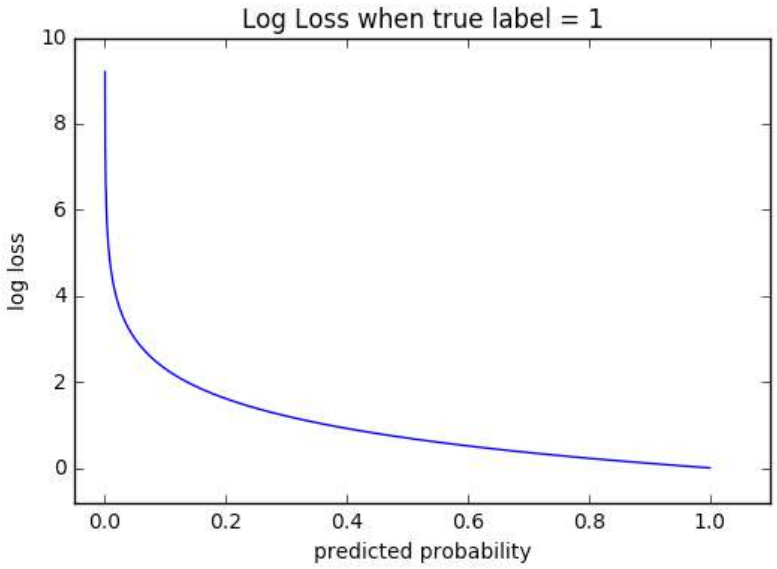


Figure III.12 A plot of the Cross-Entropy Loss function [72]

4.1.1.2 Our CNN architecture

In order to choose the best architecture for the classification task, a method of fixing some hyper parameter of the network such as filter size in a convolutional layer or number of filters, while varying the others was chosen; this also includes trying out the different combinations and ordering of layers in the limit of the architecture permitting it (sometimes certain combinations cannot work because of the inherent non-compatibility of certain layer's output shape dimensions with the following layer's input dimensions, and sometimes they're just too expensive computationally or they heavily increase the number of parameters). until we find the best solution for a particular configuration, this is continued on by adding extra layers and following the conclusions that were drawn from the previous smaller architectures, until we find the optimal CNN model based on accuracy and efficiency of the network (compute), but also based on how the model generalization performance.

4.1.1.2.1 Single weight layer architecture

At first a very simple base CNN architecture was chosen as a start point in order to determine which hyper-parameters work best for our particular task. This architecture is presented in the following Figure III.13.

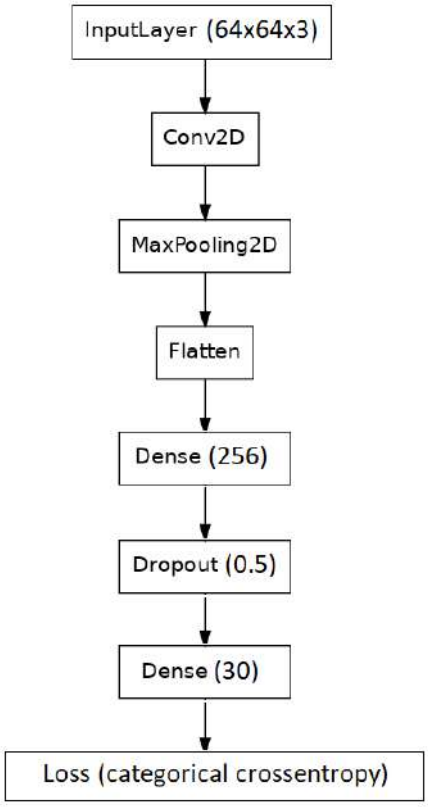


Figure III.13 The Base single weight layer CNN model

Note that the flatten layer is used to convert the 3-dimensional feature map coming out from the last max-pool layer into a vector. This enables output processing with the fully connected layers, known as dense. All networks that will be presented later use the same fully connected layers configuration and only the top part of the network is going to be modified.

Table III.2 shows the results from most of the possible combinations when using a single parametrizable weight layer (not fixed).

Table III.2 The experimental results from using the single weight layer CNN models

Weight Layer 1					
filter size	Nbr of filters	Max pooling after conv	Test accuracy	FLOPS (G) (during inference)	Nbr of parameters
	16		0.4372	8.31257728	63251678
	64	None	0.4381	33.1464164	252193310
	16		0.5034	4.31940736	32024798
	64		0.502	17.0728768	126496286
	128	2x2	0.4821	34.0775027	252458270
	16		0.5185	1.09367219	6825182
	64		0.5304	7.6598592	52964894
	128		0.5289	15.2514675	105395486
3x3	256	3x3	0.5101	30.4346842	210256670
	16		0.4171	15.7243917	118500318
	64	None	0.4192	17.3439009	118109214
	16		0.491	4.40345728	30026718
	64		0.4622	17.4090765	118503966
	128	2x2	0.4768	34.7499021	236473630
	16		0.52	1.43382451	6825950
	64		0.5129	9.02046848	52967966
	128		0.5107	17.9726861	105401630
5x5	256	3x3	0.5098	35.8771213	210268958
	16		0.3983	15.1940867	110768222
	64	None	0.4768	34.7499021	236473630
	16		0.4777	4.6153024	28094558
	64		0.4532	18.256457	110775326
	128	2x2	0.4515	36.444663	221016350
	16		0.5052	2.60191872	12365918
	64		0.4836	10.2029222	47860766
	128		0.4929	20.3375936	95187230
7x7	256	3x3	0.4643	40.6069363	189840158

What we conclude from this table is:

- -a filter size of (3x3) is sufficient to capture the local features required to achieve a good accuracy while still maintaining a low compute time and small number of parameters, meaning the features that govern our data are very small and most of them are most likely edge detectors or blob detectors.
- -using subsampling (Max-Pool) layer after the convolution layer in order to reduce the dimension of the output feature map before entering the fully connected layer drastically increases the performance of the network.

4.1.1.2.2 Two weight layer architecture

After that, we used the same approach but this time with two parametrizable weight layers instead of a single one; for the first layer, only a filter size of (3x3) was used since it proved its effectiveness in the first architecture; Table III.3 shows the results.

Table III.3 The experimental results from the 2 weight layers CNN models

Weight Layer 1			Weight Layer 2					
filter size	Nbr of filters	max pooling after	filter size	Nbr of filters	max pooling after	Test accuracy	FLOPS (G)	Nbr parameters
3x3	16	None	3x3	16	2x2	0.5107	5.12867456	30028270
				32		0.5259	9.97260416	59521790
				64		0.5059	19.6604634	118508830
				16	3x3	0.5393	3.03992755	13710062
				64		0.5707	11.2802604	53038622
				80		0.5302	24.504393	148002350
			5x5					
				16	2x2	0.4791	6.5748329	28099054
				32		0.5056	11.2923988	48772282
				64		0.5323	25.4535017	110857758
				16	3x3	0.5241	4.56144922	12370414
				32		0.541	8.84655846	24271870
				64		0.5331	17.399967	47943198
			7x7					
				16	2x2	0.4605	8.61476378	26237422
				32		0.4902	16.9447826	51940094
				64		0.4791	33.6048202	103345438
				16	3x3	0.4962	6.68515866	11164142
				32		0.5015	13.0855724	21793534
				80		0.495	32.2868134	53681710
	16	2x2	3x3	16	None	0.491	4.06497306	28094958
				32		0.5168	7.84966989	55720958
				64		0.5015	15.3938486	110775582
				16	2x2	0.5472	1.36885478	7025390

				32		0.5489	2.43221837	13384446
				64		0.536	4.58416051	26299934
				16		0.5513	0.67455923	1601006
				64	3x3	0.5638	2.6461504	11160862
				80		0.5718	3.24392269	13883182
				16		0.4968	3.94427213	24429038
				32	None	0.5109	7.60826803	48389118
				64		0.4802	14.9110449	96111902
				16		0.5337	1.6045591	6144750
			5x5	32	2x2	0.5323	2.90362701	11623166
				64		0.5363	5.52697779	22777374
				16		0.577	0.8625984	346606
				64	3x3	0.5744	4.05079987	11243038
				80		0.5641	4.9808233	13837870

From this table we conclude that:

- -a filter size of (3x3) for both layers is sufficient for the classification task. And it seems to be the best choice for a filter size for our particular case.
- -taking a small number filters in the first layer and then gradually increasing it in the second layer brings better results.

4.1.1.2.3 The rest of CNN configurations

Following on from the single weight configuration and two weight configurations, we continued using the same approach by adding more layers into the best configuration chosen from the previous smaller architecture; the (7x7) filter size was removed from the pool since it proved to have a negative effect on the accuracy while also increasing the overall computational cost.

Table III.4 shows the configurations for the 3 weight layer models. Wherein the model labeled H, which is the model containing only (3x3) size filters once again proved its effectiveness, proving even more our hypothesis from the first two configurations, consequently, from now on, we will only work with (3x3) exclusively, and the (5x5) filter will be removed from the pool.

Table III.4 The set of configurations for a 3 weight layers CNN models

CNN Configuration							
A	B	C	D	E	F	G	H
3 weight							
Conv(5X5) Conv(5X5) Conv(5X5)	Conv(5X5) Conv(5X5) Conv(3X3)	Conv (5X5) Conv (3X3) Conv (5X5)	Conv(5X5) Conv(3X3) Conv(3X3)	Conv (3X3) Conv (5X5) Conv (5X5)	Conv (3X3) Conv (5X5) Conv (3X3)	Conv (3X3) Conv (3X3) Conv (5X5)	Conv (3X3) Conv (3X3) Conv (3X3)
Max Pooling (2X2)							
Flatten							
Dense							
Dropout (0.5)							
Dense							
SoftMax							

From our experiments testing the configurations presented in this table we conclude that:

- -using sub sampling (Max-Pool) after each conv layer is the best choice for both the performance and the computational cost, so, we will be following that particular architecture.
- -fixing the size of the conv filter to be (3x3) while varying the number of filters by gradually increasing their number from layer to the next one brought good results.

Table III.5 shows the rest of the configurations starting from the 4 weight layers model all the way into 9 weight layers model.

Table III.5 The set of configurations for the 4-9 weight layers CNN models

CNN Configuration					
I	J	K	L	M	N
4 weight	5 weight	6 weight	7 weight	8 weight	9 weight
Conv (3X3) Conv (3X3) Conv (3X3) Conv (3X3)	Conv (3X3) Conv (3X3) Conv (3X3) Conv (3X3) Conv (3X3)	Conv (3X3) Conv (3X3) Conv (3X3) Conv (3X3) Conv (3X3) Conv (3X3)	Conv (3X3) Conv (3X3) Conv (3X3) Conv (3X3) Conv (3X3) Conv (3X3) Conv (3X3)	Conv (3X3) Conv (3X3) Conv (3X3) Conv (3X3) Conv (3X3) Conv (3X3) Conv (3X3) Conv (3X3)	Conv (3X3) Conv (3X3) Conv (3X3) Conv (3X3) Conv (3X3) Conv (3X3) Conv (3X3) Conv (3X3) Conv (3X3)
Max Pooling (2X2)					
Flatten					
Dense					
Dropout (0.5)					
Dense					
SoftMax					

From our experiments testing the configurations presented in this table we conclude that:

- The J model with the 5 weight layers configuration has the best performance to efficiency ratio, wherein it achieved very good results and it was the least computationally expensive of the other models, which means there is no need to increase the number of weight layers more than 5 since we will not be getting any significant performance increase that the price to pay computationally will be worth it. consequently, from now on, we will focus only on this particular model and how to improve it.

4.1.1.2.4 Improving the 5 weight layers model

Carrying on from our previous experiments regarding the architectural choices which seemed to work best for our classification task, we have found that much like the AlexNet [53] architecture, the choice of using five convolutional layers, which are followed by max-pooling layers each, and three fully-connected layers at the end seems to obtain the best results. We wanted to go a step further in order to improve the performance of this particular model; consequently, and inspired by the technique of using only small filters of size (3x3) employed in the VGG-Net [54] architecture and the batch normalization layer employed in ResNet [73], we created an architecture that is a hybrid between the three models.

A method of adding a batch normalization layer after each convolutional layer was chosen, this way the feature maps coming out of the conv layers got re-scaled and re-centered before entering the next conv layer; this in fact improved the generalization of the model quite a bit which made the network attain the 5% of the final test accuracy around the 15'th epoch compared to the model which did not use the Batch-Norm where it converged almost immediately in the first 5 epochs making it clear that an overfitting is most likely present; this increased the overall required training time but it also improved the test accuracy which means the models generalization capability grew quite a bit. Figure III.14 shows the difference between the training graphs of the models with and without the Batch-Norm layers.

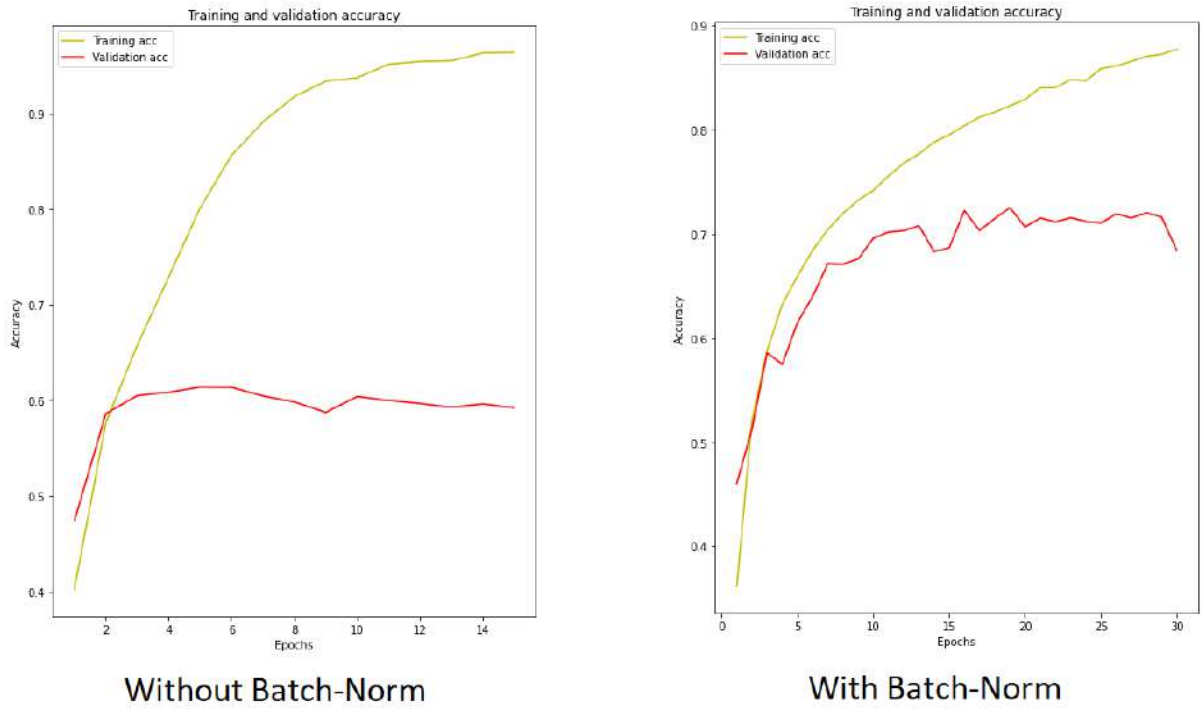


Figure III.14 Training graphs for the model which used Batch-Norm and the one that did not

We noticed an increase of about 13% in test accuracy after adding Batch-Norm, as a result, we have successfully come up with significantly more accurate ConvNet architectures.

Note that for the second plot in Figure III.14, we did not continue the training process up util the convergence point, because our goal is to only show the effects of using batch-norm, which are apparent without the need to add more epochs.

For the final architecture, it was chosen among almost a hundred different models based on their accuracy and computational complexity, it achieved a test accuracy of 78% and number of parameters of only 745,118, and the required FLOPS for inferencing of 26.79 G; making it vastly superior to the state-of-the-art ResNet architecture in terms of performance for our particular task where it achieved a test accuracy of only 63% had 25.6 million parameters.

Figure III.15 shows the final CNN classification model we have chosen.

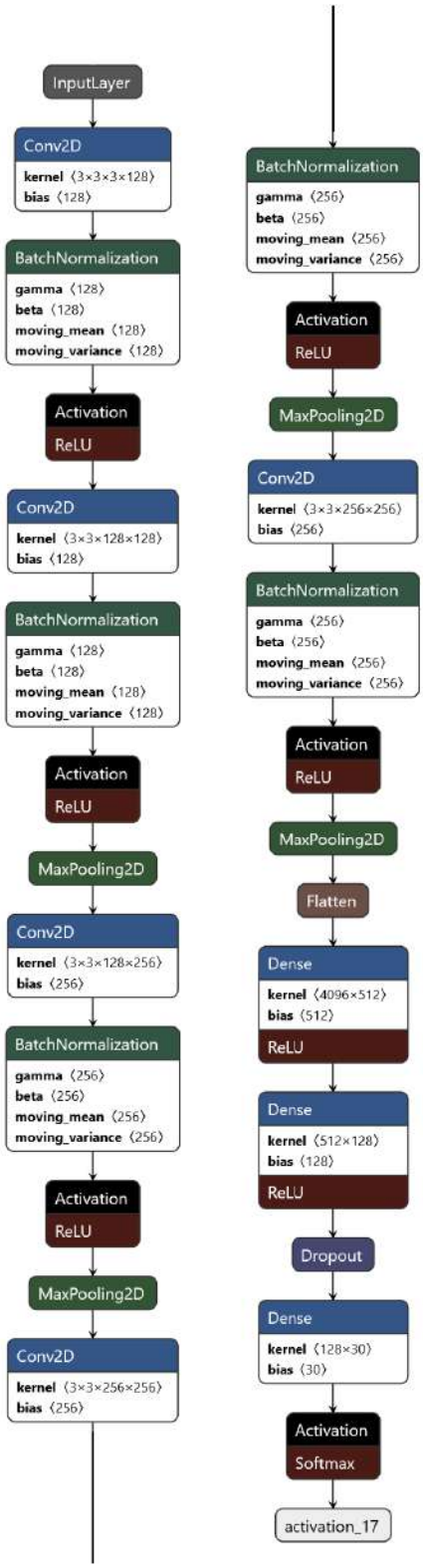


Figure III.15 The final classification CNN model

4.1.1.2.5 Training The Network

In order to speed up training time to be able to try out all the different Deep Learning models architecture configurations and make changes accordingly, we have used the publicly available GPU equipped platforms of both Google Collaboratory [74], and Kaggle [75], their servers permitted us to run Python code that is optimized for their Virtual Machines with the help of the Keras Library [76]; with their very fast GPU accelerators, it made the training time decrease by factors of 20 when compared to using our own local machine.

This however came with a downside which is the low amount of RAM size available, this in turn brought a constraint on the model size and amount of image data that can be stored in RAM at a given time, furthermore it limited the amount of image data that can be loaded at once into the models input; in order to overcome this problem, a method of batch training was used, wherein each iteration the data is loaded by small chunks of 128 into the models input, each time updating the model parameters according to the data, this is repeated on until the model goes through all the dataset images and learns all the necessary features from them.

A number of optimization algorithms were used including Stochastic Gradient Descent (SGD) and Adam [77], but the final choice was to the Adam optimizer for performing better and faster; it is worth mentioning that the SGD algorithm also brought similar results and even had smoother accuracy graphs which meant that the transitions of accuracy values from epoch to epoch were monotonic, but Adam was faster in converging.

The Adam optimization algorithm was used to train the model, with the learning rate at first being fixed to 0.1 and then gradually decreasing it exponentially every 10 epochs by using a step decay scheduler with the Keras callbacks function. The Number of epochs for which we trained our model is a 100, and the training lasted 4 hours approximatively. Figure III.16 shows how the learning rate is decayed thought the training process.

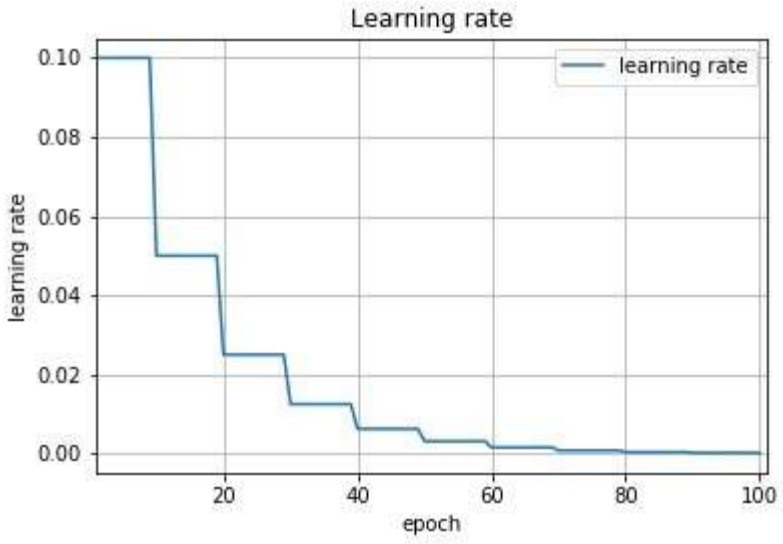


Figure III.16 Learning Rate Step decay scheduling

4.1.1.2.6 CNN model analysis

This is done in order to verify that the generalization performance of the network that its learning actual useful features and not just overfitting the statistical noise. From Table III.7, we notice that the zebra class sits on top when it comes to precision, this is in fact very logical because most likely the features required to distinguish this class is a simple low level texture feature, which verify our hypothesis that the CNN model is actually learning only the useful features; another classes with a very high precision are the giraffe, umbrella, and airplane classes, the learned features that distinguish these three classes are most likely low level shape features, because if we look at an image that contains one of these, only the contour shape will be required in order to classify these.

Table III.6 Classification report on the testing dataset for the 30 classes

Categories	precision	recall	f1-score	Support
person	0.65	0.79	0.71	1000
dining table	0.68	0.59	0.63	1000
chair	0.46	0.56	0.51	1000
bowl	0.67	0.69	0.68	1000
car	0.62	0.68	0.65	1000
cat	0.48	0.74	0.58	1000
bus	0.84	0.67	0.74	1000
cup	0.62	0.71	0.67	1000
pizza	0.79	0.81	0.80	1000
dog	0.37	0.65	0.47	1000
truck	0.64	0.50	0.56	1000
couch	0.59	0.34	0.43	1000
tv	0.85	0.81	0.83	1000
bed	0.56	0.48	0.52	1000
train	0.86	0.71	0.78	1000
giraffe	0.93	0.96	0.94	1000
cake	0.60	0.42	0.49	984
elephant	0.84	0.68	0.75	987
motorcycle	0.80	0.84	0.82	1000
toilet	0.93	0.69	0.79	1000
laptop	0.86	0.70	0.77	1000
airplane	0.93	0.73	0.82	648
broccoli	0.74	0.95	0.83	1000
sandwich	0.56	0.56	0.56	947
zebra	0.95	0.96	0.95	987
banana	0.45	0.89	0.60	747
horse	0.73	0.62	0.67	830
teddy bear	0.75	0.62	0.68	840
umbrella	0.89	0.63	0.74	886
cow	0.75	0.51	0.61	718

4.1.1.2.7 Extracted Features

As we have mentioned in Chapter I, a large part of this work is dedicated to finding the best feature vectors in order to achieve the task of AIA; we let a CNN classification model train itself repeatedly in order to find the best class discriminative feature representations that assure a good overall accuracy score, and then these features are extracted from the trained CNN model.

We extracted the feature vectors by inputting the various class images and then retrieving the layer activations from the last Fully Connected layer that has a size of 128, right before the final size 30 FC layer that outputs the final classification results, these activation vectors are then used as our feature representations. Figure III.17 shows the process.

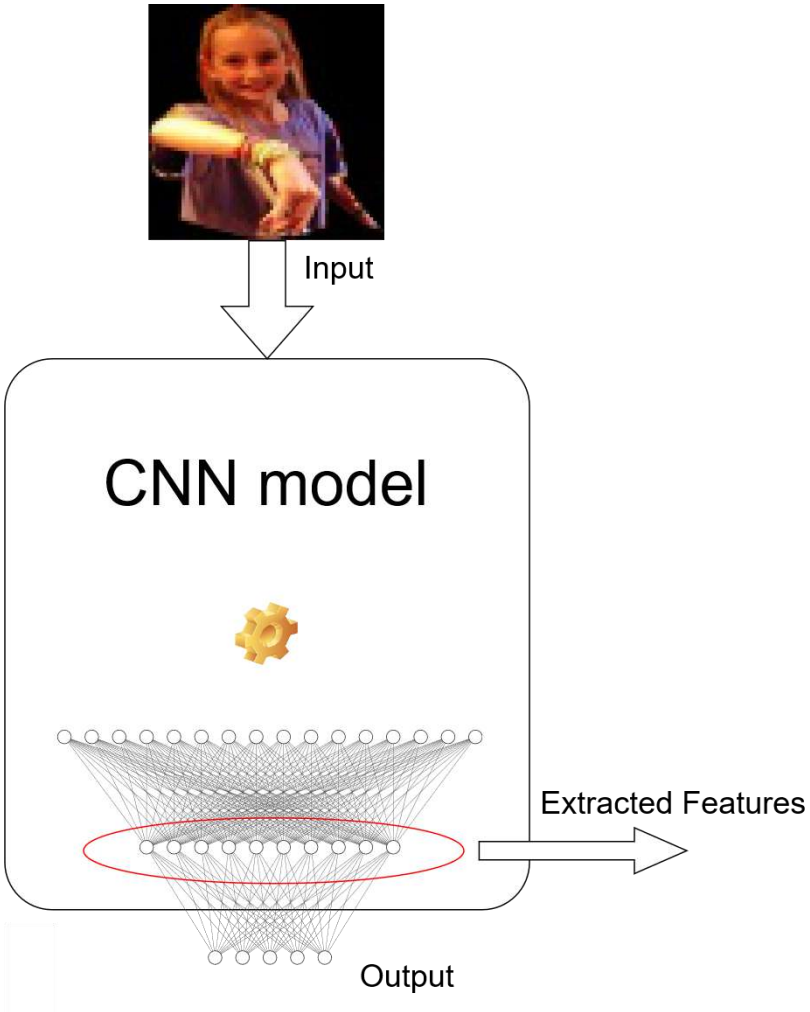


Figure III.17 How features are extracted

4.1.2 The Compressing Autoencoder

In order to reduce the dimensionality of the extracted feature vectors as well as to only keep the relevant data, an Autoencoder was trained to ignore any insignificant data present in the 128-dimensional feature vectors, and produce a compact vector of size 50, aiming to force the learned representations to assume only useful properties. Turning the latter vectors into an efficient coding of the feature representations.

We have tried a number of different architectures for the Autoencoder model, including Deep Learning based ones using 1D Convolutions with Sub-Sampling and Up-Sampling, but it brought no good results that would make the extra computational cost worth it, instead a simple MLP model (Fully Connected) was implemented. This model gradually decreases the size of the Dense (FC) layers until the bottleneck of size 50; where the compressed features will be extracted from its activations, and then it gradually increases again until it reaches back the input size of 128. It has a number of parameters of only 65,140 and an inference FLOPS of only 8.29 M, making it computationally negligible compared with the CNN model. Figure III.18 shows the diagram of the AE architecture.

For training the AE, a simple loss function of Mean Squared Error was used in combination with the Stochastic Gradient Descent with momentum algorithm to try to match the input to the output; A learning rate of 0.005 and momentum of 0.8 values were fixed for training the network, this resulted in an almost instant training time of about 5 mins because of how small and simple the network is. it took about 120 epochs for this model to converge to its final accuracy value of 70% and had a good generalization ability where the train and test accuracies differed by only about 2%.

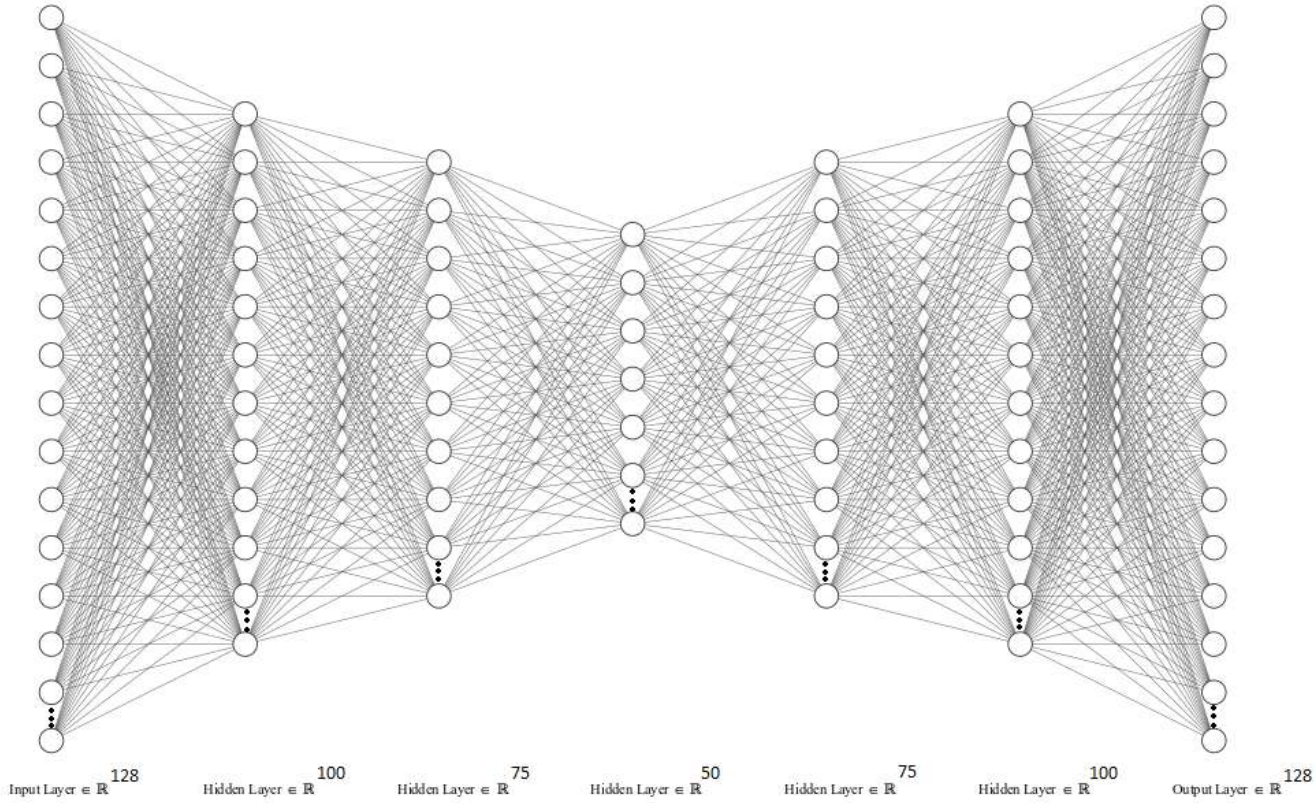


Figure III.18 Architecture of Autoencoder used

4.1.3 Analysis of features

In order to test out if the class specific properties were carried over from the CNN classification model into our feature representations, we had to verify this either analytically or visually; we opted for the more appealing visual method. Two random images from the class 'person', as shown in Figure III.19, were loaded into the DL model's input, and then their representative feature vectors were recovered and overlayed into each other in Figure III.20.



Figure III.19 A sample of two images from the class 'person'

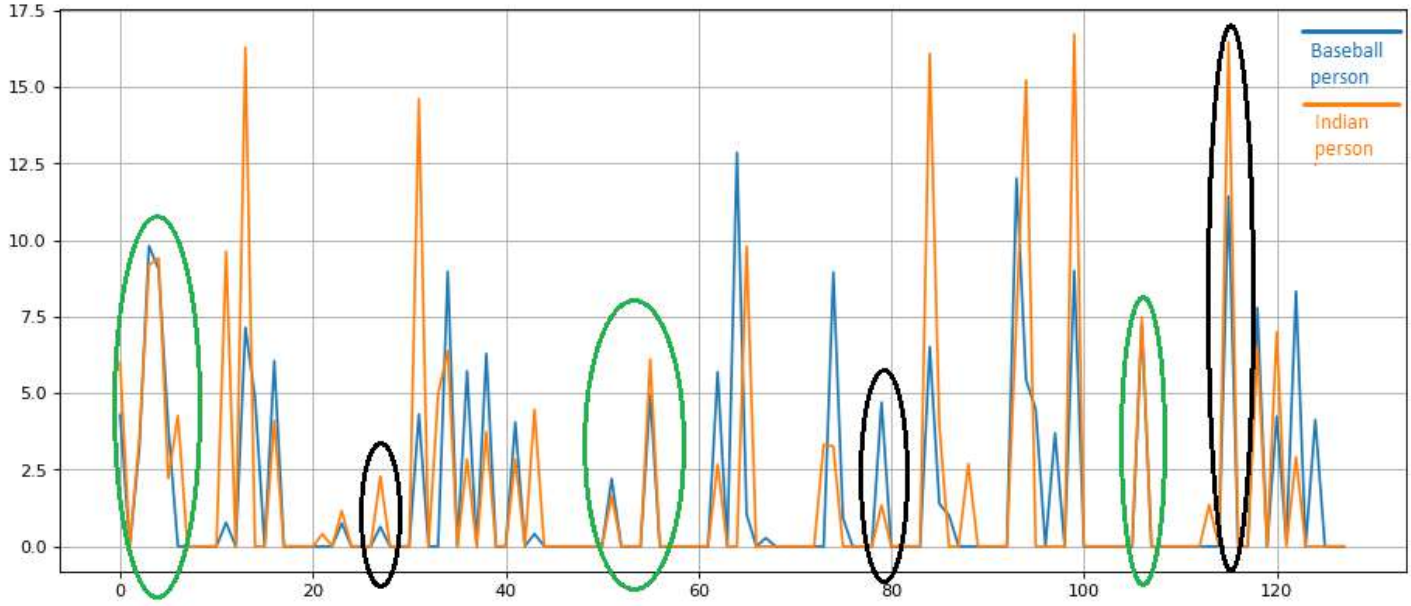


Figure III.20 An overlayed plot of two feature vectors belonging to two random instances of the class 'person'

We notice immediately is the amount of similarity between the respective peaks from the two features, where for example, the peaks encircled by green are almost identical in amplitude and shape, these may be the features that distinguish the class ‘person’, perhaps they represent the different human body components, such as hands or feet; another interesting observation is that there are some peaks that have the same overall shape and placement, but their peaks are less pointy (they have lower amplitudes), these are encircled by black; this could mean that some features in a certain image are less or more apparent, such a hand being half covered or a shadow

covered face. And then there are peaks only present in one of the two features, these could relate to the orientation or the posture or the clothing type.

Figure III.21 shows the correlation between the two feature representations of the images from the class ‘person’, it is apparent that there is a high correlation between them as pointed out by the single peak in the middle and the low noise level everywhere else, approximating a Dirac impulse.

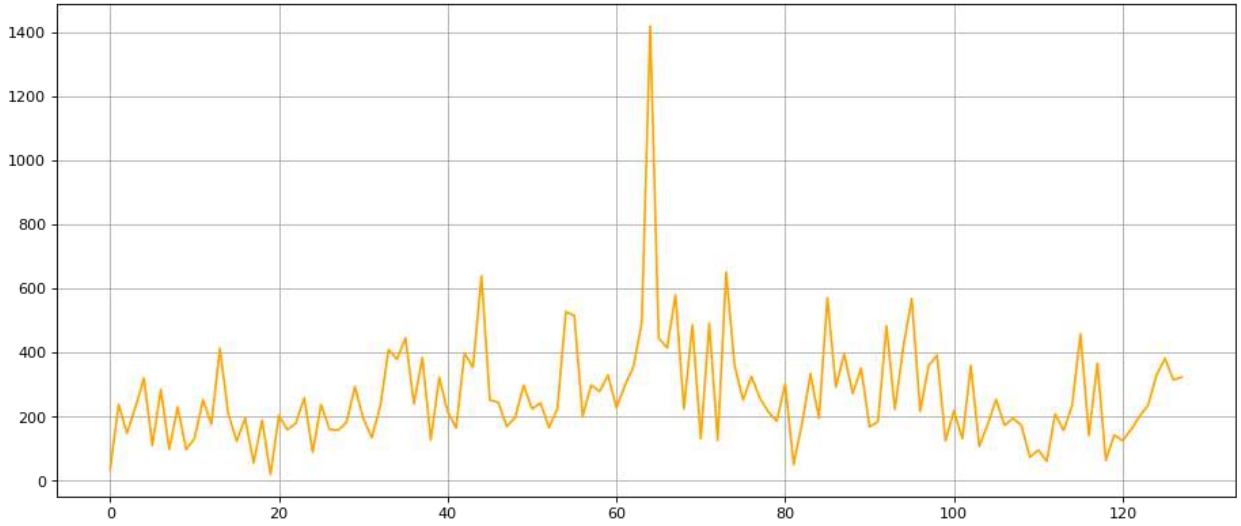


Figure III.21 The result from a correlation between the two extracted features of the class 'person'

To test out the hypothesis that the peaks from the feature vectors are in fact related to each object class and its particular components, we used two different images from two different classes, as shown in Figure III.22 and repeated the same experiment.

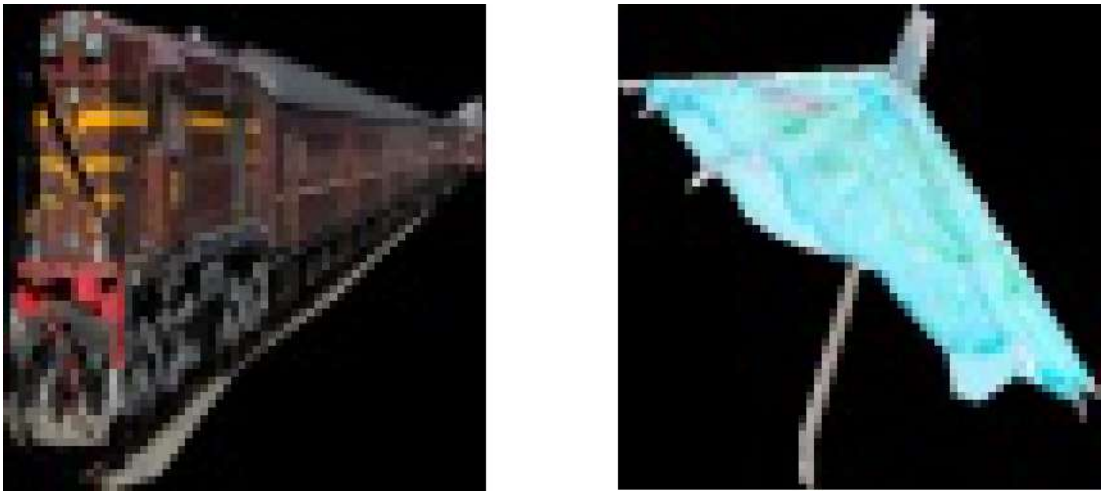


Figure III.22 A sample of two images from different the classes, one from 'umbrella' and one from 'train'

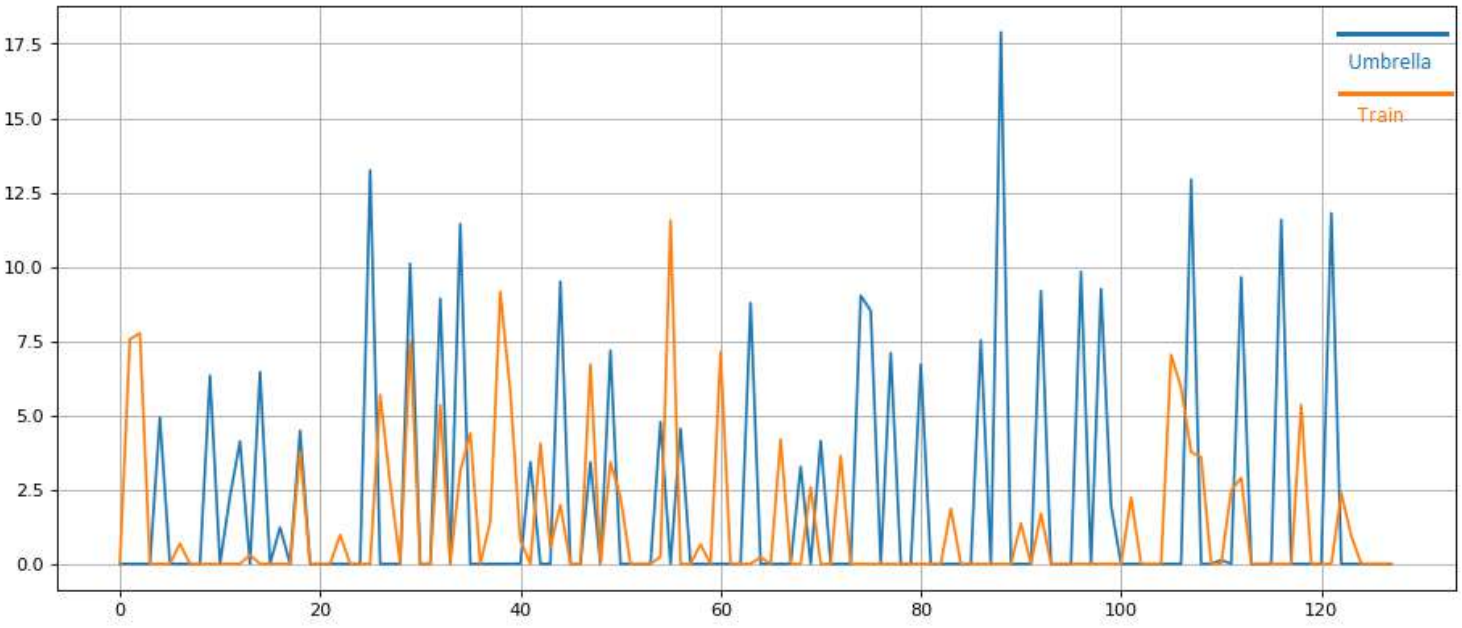


Figure III.23 An overlayed plot of two feature vectors belonging to two different classes, one of 'train' and one from 'umbrella'

From Figure III.23, we observe that the peaks from the feature representations are uncorrelated, in both shape, placement and amplitudes, except for about 4 peaks which have the same placement but different amplitudes. This is in fact confirmed by Figure III.25 where the correlation between their two feature representations resulted in a noise, which means there is no correlation between these two classes.

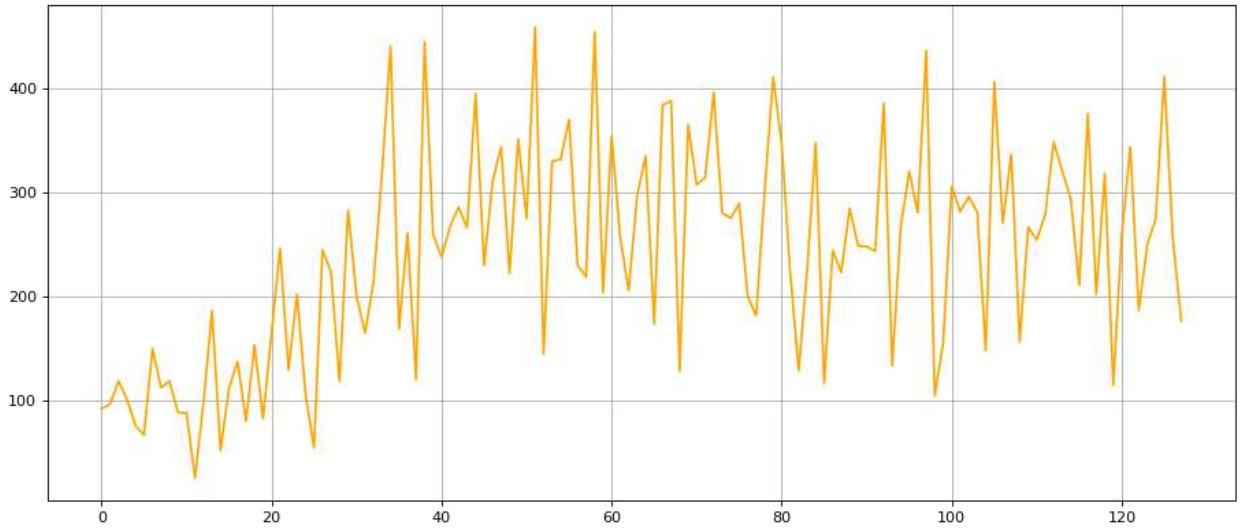


Figure III.24 The result from a correlation between the two extracted features of two different classes, one from Train and the other from Umbrella

Note that from these set of experiments, one can hypothesize that perhaps these feature vectors can be exploited for future works such as pose estimation, front facing or back facing detection, clothes type classification and so one; since they seem to contain the necessary information for many other tasks other than just classification.

In order to plot the feature vectors in a more informative way, we have used a Principal Component Analysis (PCA) method in order to reduce the dimensionality from 128 into 2D, in order to plot them, each dot (x, y) represents now a reduced feature vector of a particular input image, Figure III.29 and Figure III.30 shows the resulting plots; we notice right away that the overall structure between the 128 feature vectors and the compressed 50 sized ones stayed the same, we also notice that not only the features that are supposed to be similar such as car and bus are grouped side by side, but also, the classes which achieved the best accuracy results such as zebra are somewhat far from the rest of the classes, making the classification algorithm easily distinguish them.

To test out the effectiveness of these features, an unsupervised KNN clustering algorithm was used to get a rough idea on whether or not a class can be found using only its feature representation; only the number of classes of 30 was given to the KNN algorithm and was asked to find the hidden structure using the patterns present in the features without explicitly giving any information about the location or size of each class cluster, Figure III.29 and Figure III.30 clearly show that it is in fact possible and the resultant clusters are very close to the original, especially some classes such as zebra are almost identical.

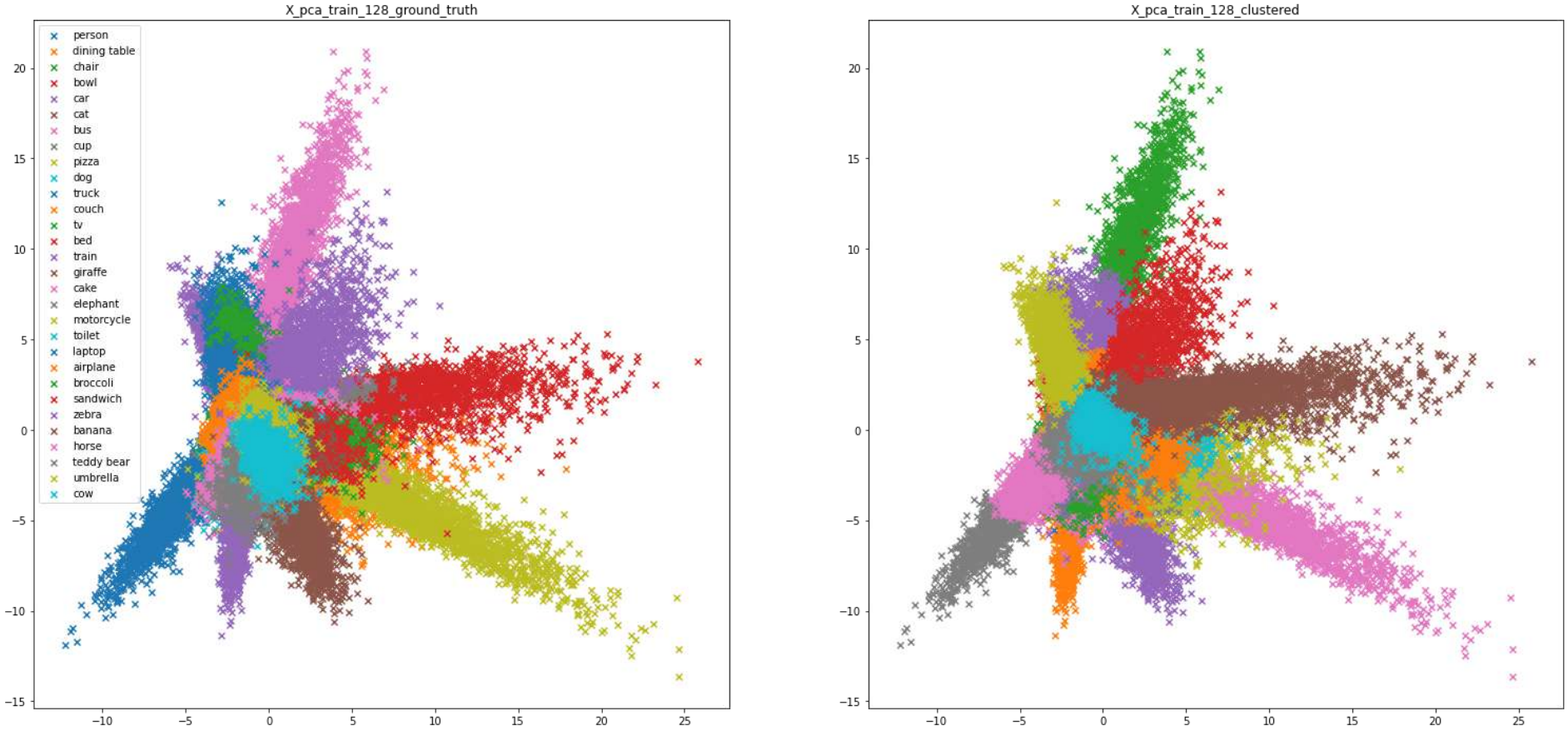


Figure III.25 A scatter plot of the PCA reduced feature vectors of size 128 labeled correctly and their automatically clustered version

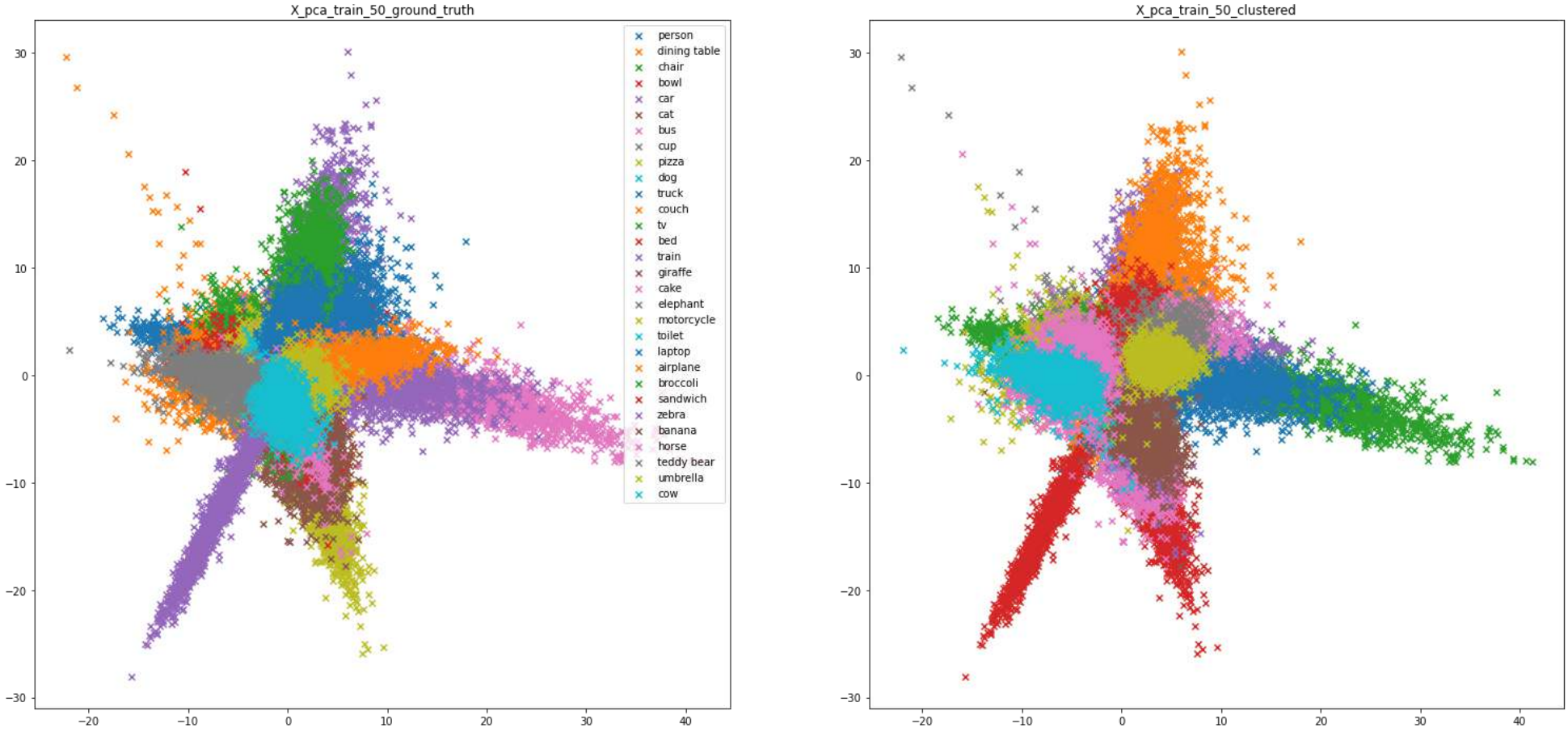


Figure III.26 A scatter plot of the PCA reduced feature vectors of size 50 labeled correctly and their automatically clustered version

5 Final annotation (Similarity measure)

Finally, for generating the annotations for each image, a similarity measure method was used; we firstly assumed that there is a single englobing feature vector for each class, this vector then becomes an identifier for that particular class, to measure the distance between a test image's feature vector and the class vector; we chose the simplest method of measuring the Euclidean distance. The following formula shows how to calculate the distance between two vectors x and y:

$$d(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

The overall procedure that is used is resumed in these steps:

1. Take a test image and extract the features from it using the CNN model.
2. obtain the compressed feature vectors by passing them through the Autoencoder.
3. measure the similarity between the class specific identifier feature vector with the test image's feature vector.

6 The Final Automatic Image Annotation Architecture

This Section provides a high-level overview and explains the final architecture of our Automatic Image Annotation algorithm; there are three main parts to this architecture. Firstly, a trained classification CNN model is used to extract the feature vectors from each image, and then the extracted features get compressed using an MLP-based Autoencoder model, and finally, a similarity measure method is used to determine the distance between a given test image's feature vector and the class specific feature vector and output the final annotation. Figure III.30 shows the final architecture.

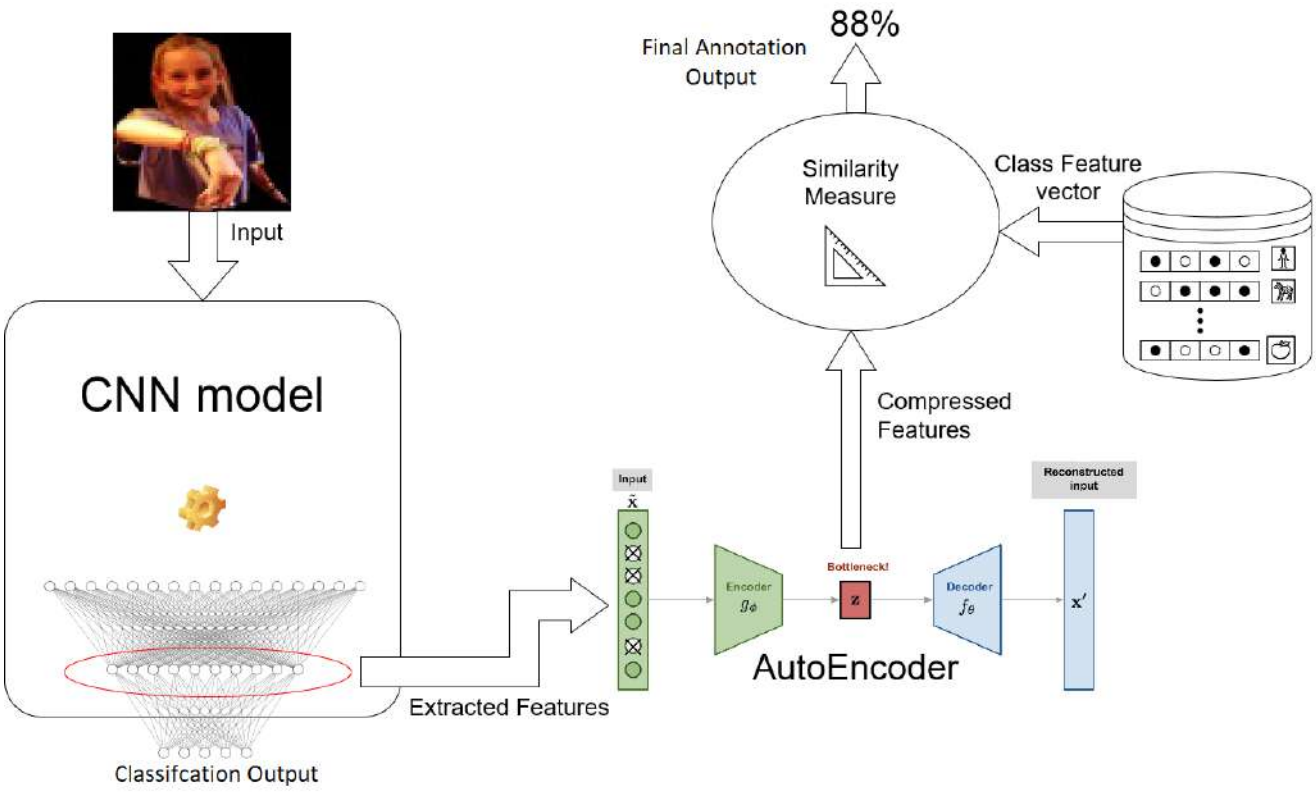


Figure III.27 The Final Automatic Image Annotation Architecture

7 Results and Discussion:

Our results show that a large, deep convolutional neural network is capable of achieving good results on a highly challenging dataset using purely supervised learning. Figure III.31 shows the annotated images of class ‘cat’ that their feature representation are closest in distance using our similarity measure method, we observe that most of the cat pictures in fact look similar and are most likely of the most breed apart from some images that seem to be somewhat different (most likely attributed to the non-perfect accuracy of our models), no matter their orientation or shape, our algorithm was able to successfully find the similarity between them.



Figure III.28 Randomly selected images with the smallest Euclidean distance between their feature vectors in the class ‘cat’

It is however noteworthy that the method of using a single feature vector as a class identifier is not optimal, Figure III.31 shows how the identifier achieved by averaging the class vectors for some classes is not centered and sometimes does not even belong to the same blob of its class feature vectors, this could be a subject of improvement for the future. Perhaps even try out a combination of features as an identifier and not a single one, sort of like bag of features model.

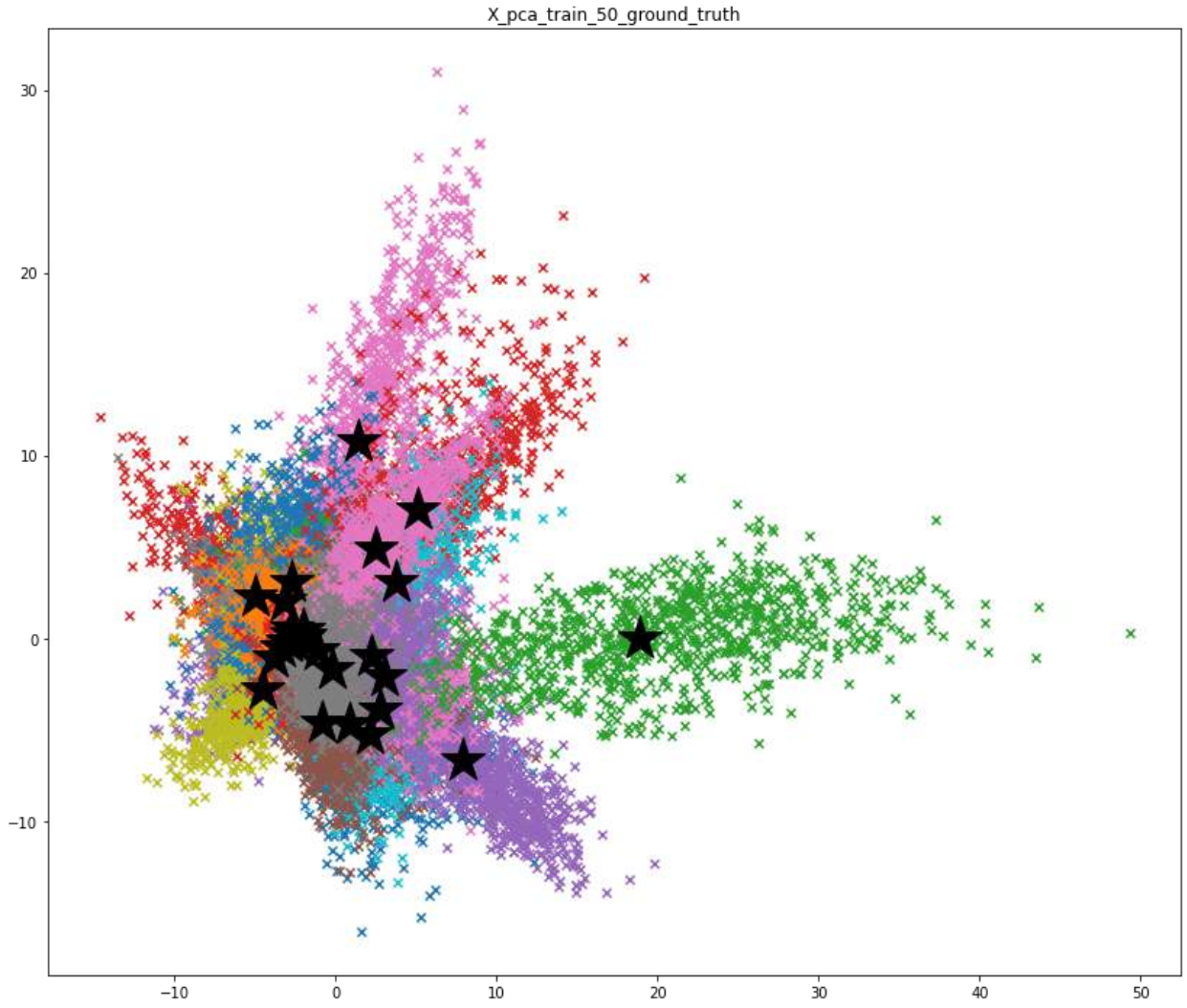


Figure III.29 A scatter plot of PCA reduced feature vectors and their respective class identifier drawn in black

8 Conclusion

We have in this chapter, started by laying out the research strategy that was used throughout our work; we then showcased the dataset we used in order to train the Deep Learning models; we continued on afterwards describing the methodology we followed for finding the best CNN classification model, we then described how we used this model In order to generate the feature vectors necessary for the task image annotation; after that we described how an Autoencoder model was used to compress the extracted feature vectors where we also provided an extensive analysis into the properties of these feature representation; we then finally provided the full architecture to our AIA system and how we evaluated our results and then discussed them.

Conclusions and perspectives

The objective of this research was to develop an Automatic Image Annotation algorithm based on Deep Learning techniques, in order to do so, a new technique for feature extraction was developed and tested; The proposed approach is based on extracting a compact representation of the images from a trained Convolutional Neural Network model's activations, from the Fully Connected layer of size 128 to be exact; these feature representations are then compressed to size 50 using an Autoencoder model; this step did not come without any loss of information of course, but as we showed in the analysis section, most of the class specific structure and patterns inherent in the features remained the same after compression, which showed that there is a strong correlation between the class types and their feature representations. Finally, in order to reduce the semantic gap between the features and their respective class annotations, a similarity measure method was used to measure the distance between a particular class and the feature vector outputted by an unlabeled test image, this was done by assuming that a single englobing feature vector for each class is sufficient to characterize the whole class. In conclusion, our objectives were met.

In the future, as shown in the feature analysis section, these CNN based feature vectors show great potential to be used for other tasks, such as human pose estimation, dog and cat breed classification, human clothes classification and even CBIR, provided that the similarity measure will be improved without assuming the overly simplistic idea of a having a single vector that is sufficient to characterize the whole class; perhaps even employ a Deep learning technique in the similarity measure part, in order to automatically find the best distance metric that reduces the semantic gap between features and annotation tags and measure the correlation between them. Also, a pre-processing can be added to the CNN model in order to improve its accuracy and make the class separation even higher.

Bibliography

[1] H. Guo, L. Wang, F. Chen, and D. Liang, "Scientific big data and Digital Earth," *Chinese Science Bulletin*, vol. 59, no. 35, pp. 5066–5073, Dec. 2014, doi: 10.1007/S11434-014-0645-3.

[2] "175 Zettabytes By 2025." https://www.forbes.com/sites/tomcoughlin/2018/11/27/175-zettabytes-by-2025/?sh=32d579075459 (accessed Sep. 18, 2021).

[3] "A Guide to Better Google Searching Techniques - Hongkiat." https://www.hongkiat.com/blog/better-google-searching/ (accessed Jul. 15, 2021).

[4] "Introduction to Image Annotation for Machine Learning and AI." https://labelbox.com/image-annotation-overview (accessed Jun. 17, 2021).

[5] S. Lindstaedt, R. Mörzinger, R. Sorschag, V. Pammer, and G. Thallinger, "Automatic image annotation using visual content and folksonomies," *Multimedia Tools and Applications*, vol. 42, no. 1, pp. 97–113, Mar. 2009, doi: 10.1007/s11042-008-0247-7.

[6] D. Zhang, M. M. Islam, and G. Lu, "A review on automatic image annotation techniques," *Pattern Recognition*, vol. 45, no. 1, pp. 346–362, Jan. 2012, doi: 10.1016/j.patcog.2011.05.013.

[7] "Deep learning - Wikipedia." https://en.wikipedia.org/wiki/Deep_learning (accessed Jun. 21, 2021).

[8] N. Kulathunga, N. R. Ranasinghe, D. Vrinceanu, Z. Kinsman, L. Huang, and Y. Wang, "Effects of the Nonlinearity in Activation Functions on the Performance of Deep Learning Models," Oct. 2020, [Online]. Available: http://arxiv.org/abs/2010.07359

[9] R. Datta, D. Joshi, J. Li, and J. Z. Wang, "Image retrieval: Ideas, influences, and trends of the new age," *ACM Computing Surveys*, vol. 40, no. 2, Apr. 2008, doi: 10.1145/1348246.1348248.

[10] T.Karthikeyan, P.Manikandaprabhu, and S.Nithya, "A Survey on Text and Content Based Image Retrieval System for Image Mining," *International Journal of Engineering Research*

& Technology, vol. 3, no. 3, Mar. 2014, Accessed: Jul. 27, 2021. [Online]. Available: www.ijert.org

[11] "(PDF) Image Retrieval: A Literature Review." https://www.researchgate.net/publication/264276256_Image_Retrieval_A_Literature_Review (accessed Jul. 16, 2021).

[12] Y. Takao, S. Itoh, and J. Iisaka, "An image-oriented database system," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 81 LNCS, pp. 527–538, 1980, doi: 10.1007/3-540-09763-5_28.

[13] G. Y. Tang, "A management system for an integrated database of pictures and alphanumerical data," *Computer Graphics and Image Processing*, vol. 16, no. 3, pp. 270–286, Jul. 1981, doi: 10.1016/0146-664X(81)90040-X.

[14] D. G. Johnson, "An image storage system using a relational database An image storage system using a relational database management system to facilitate picture data handling management system to facilitate picture data handling." [Online]. Available: https://scholarworks.rit.edu/theses

[15] S. K. Chang, B. S. Lin, and R. Walser, "A generalized zooming technique for pictorial database system," *Pictorial information ssystems*, pp. 257–287, 1980, doi: 10.1007/3-540-09757-0_10.

[16] T. Kato, "Database architecture for content-based image retrieval," *Image Storage and Retrieval Systems*, vol. 1662, pp. 112–123, Apr. 1992, doi: 10.1117/12.58497.

[17] N. Seth and S. Jindal, "A REVIEW ON CONTENT BASED IMAGE RETRIEVAL," *INTERNATIONAL JOURNAL OF COMPUTERS & TECHNOLOGY*, vol. 15, no. 14, pp. 7498–7503, Jan. 2016, doi: 10.24297/IJCT.V15I14.5640.

[18] "Feature (computer vision) - Wikipedia." https://en.wikipedia.org/wiki/Feature_(computer_vision) (accessed Sep. 19, 2021).

[19] F. Baig *et al.*, "Boosting the Performance of the BoVW Model Using SURF–CoHOG-Based Sparse Features with Relevance Feedback for CBIR," *Iranian Journal of Science and*

Technology - Transactions of Electrical Engineering, vol. 44, no. 1, pp. 99–118, Mar. 2020, doi: 10.1007/S40998-019-00237-Z.

[20] M. J. Swain and D. H. Ballard, "Color indexing," *International Journal of Computer Vision 1991 7:1*, vol. 7, no. 1, pp. 11–32, 1991, doi: 10.1007/BF00130487.

[21] M. A. Stricker and M. Orengo, "Similarity of color images," *Storage and Retrieval for Image and Video Databases III*, vol. 2420, p. 381, Mar. 1995, doi: 10.1117/12.205308.

[22] J. Huang, S. R. Kumar, M. Mitra, W. J. Zhu, and R. Zabih, "Image indexing using color correlograms," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 762–768, 1997, doi: 10.1109/CVPR.1997.609412.

[23] G. Pass and R. Zabih, "Histogram refinement for content-based image retrieval," *IEEE Workshop on Applications of Computer Vision - Proceedings*, pp. 96–102, 1996, doi: 10.1109/ACV.1996.572008.

[24] A. Khokher and R. Talwar, "Content-based Image Retrieval: Feature Extraction Techniques and Applications," *undefined*, 2012.

[25] H. Tamura, S. Mori, and T. Yamawaki, "Textural Features Corresponding to Visual Perception," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 8, no. 6, pp. 460–473, 1978, doi: 10.1109/TSMC.1978.4309999.

[26] R. M. Haralick, I. Dinstein, and K. Shanmugam, "Textural Features for Image Classification," *IEEE Transactions on Systems, Man and Cybernetics*, vol. SMC-3, no. 6, pp. 610–621, 1973, doi: 10.1109/TSMC.1973.4309314.

[27] C. C. Gotlieb and H. E. Kreyszig, "Texture descriptors based on co-occurrence matrices," *Computer Vision, Graphics, and Image Processing*, vol. 51, no. 1, pp. 70–86, Jul. 1990, doi: 10.1016/S0734-189X(05)80063-5.

[28] J. G. Daugman, "Complete Discrete 2-D Gabor Transforms by Neural Networks for Image Analysis and Compression," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 36, no. 7, pp. 1169–1179, 1988, doi: 10.1109/29.1644.

[29] N. Suematsu, Y. Ishida, A. Hayashi, and T. Kanbara, “Region-Based Image Retrieval using Wavelet Transform,” 2002.

[30] M. J. Swain and D. H. Ballard, “Color indexing,” *International Journal of Computer Vision 1991 7:1*, vol. 7, no. 1, pp. 11–32, 1991, doi: 10.1007/BF00130487.

[31] Y. Rui, Y. Rui, A. C. She, and T. S. Huang, “Modified Fourier descriptors for shape representation -- a practical approach,” *PROC OF FIRST INTERNATIONAL WORKSHOP ON IMAGE DATABASES AND MULTI MEDIA SEARCH*, 1996, Accessed: Aug. 03, 2021. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.36.6197

[32] M. K. Hu, “Visual Pattern Recognition by Moment Invariants,” *IRE Transactions on Information Theory*, vol. 8, no. 2, pp. 179–187, 1962, doi: 10.1109/TIT.1962.1057692.

[33] Y. Rui, T. S. Huang, and S. Mehrotra, “Content-based image retrieval with relevance feedback in MARS,” *IEEE International Conference on Image Processing*, vol. 2, pp. 815–818, 1997, doi: 10.1109/ICIP.1997.638621.

[34] Y. Chen, X. Zeng, X. Chen, and W. Guo, “A survey on automatic image annotation,” *Applied Intelligence 2020 50:10*, vol. 50, no. 10, pp. 3412–3428, Jun. 2020, doi: 10.1007/S10489-020-01696-2.

[35] C. Cusano, G. Ciocca, and R. Schettini, “Image annotation using SVM,” *Internet Imaging V*, vol. 5304, pp. 330–338, Dec. 2003, doi: 10.1117/12.526746.

[36] S. L. Feng, R. Manmatha, and V. Lavrenko, “Multiple Bernoulli relevance models for image and video annotation,” *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, 2004, doi: 10.1109/CVPR.2004.1315274.

[37] J. Z. Wang, J. Li, and G. Wiederhold, “SIMPLIcity: Semantics-sensitive integrated Matching for Picture Libraries,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 9, pp. 947–963, Sep. 2001, doi: 10.1109/34.955109.

[38] X. Qi and Y. Han, “Incorporating multiple SVMs for automatic image annotation,” *Pattern Recognition*, vol. 40, no. 2, pp. 728–741, Feb. 2007, doi: 10.1016/j.patcog.2006.04.042.

[39] K.-W. Park, J.-W. Jeong, and D.-H. Lee, "OLYBIA: Ontology-Based Automatic Image Annotation System Using Semantic Inference Rules," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 4443 LNCS, pp. 485–496, 2007, doi: 10.1007/978-3-540-71703-4_42.

[40] J. Fan, Y. Gao, H. Luo, and G. Xu, "Automatic image annotation by using concept-sensitive salient objects for image content representation," *Proceedings of Sheffield SIGIR - Twenty-Seventh Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 361–368, 2004, doi: 10.1145/1008992.1009055.

[41] G. R. Cross and A. K. Jain, "Markov Random Field Texture Models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-5, no. 1, pp. 25–39, 1983, doi: 10.1109/TPAMI.1983.4767341.

[42] M. Liao, J. Qin, and Y. Tan, "Texture classification and segmentation using simultaneous autoregressive random model," *Proceedings - IEEE Symposium on Computer-Based Medical Systems*, vol. 1992-June, pp. 398–401, 1992, doi: 10.1109/CBMS.1992.244923.

[43] B. B. Chaudhuri and N. Sarkar, "Texture Segmentation Using Fractal Dimension," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, no. 1, pp. 72–77, 1995, doi: 10.1109/34.368149.

[44] I. J. Sumana, M. M. Islam, D. Zhang, and G. Lu, "Content based image retrieval using curvelet transform," *Proceedings of the 2008 IEEE 10th Workshop on Multimedia Signal Processing, MMSP 2008*, pp. 11–16, 2008, doi: 10.1109/MMSP.2008.4665041.

[45] Md. M. Islam, D. Zhang, and G. Lu, "Region Based Color Image Retrieval Using Curvelet Transform," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 5995 LNCS, no. PART 2, pp. 448–457, 2009, doi: 10.1007/978-3-642-12304-7_42.

[46] Y. Liu, D. Tjondronegoro, J. Zhang, and S. Geva, "A shape ontology framework for bird classification," *Proceedings - Digital Image Computing Techniques and Applications: 9th Biennial Conference of the Australian Pattern Recognition Society, DICTA 2007*, pp. 478–484, 2007, doi: 10.1109/DICTA.2007.4426835.

[47] Q. Cheng, Q. Zhang, P. Fu, C. Tu, and S. Li, "A survey and analysis on automatic image annotation," *Pattern Recognition*, vol. 79, pp. 242–259, Jul. 2018, doi: 10.1016/J.PATCOG.2018.02.017.

[48] Y. Gong, Y. Jia, T. Leung, A. Toshev, and S. Ioffe, "Deep Convolutional Ranking for Multilabel Image Annotation," *2nd International Conference on Learning Representations, ICLR 2014 - Conference Track Proceedings*, Dec. 2013, Accessed: Aug. 29, 2021. [Online]. Available: https://arxiv.org/abs/1312.4894v2

[49] "NUS-WIDE Homepage." https://lms.comp.nus.edu.sg/wp-content/uploads/2019/research/nuswide/NUS-WIDE.html (accessed Sep. 19, 2021).

[50] M. B. Mayhew, B. Chen, and K. S. Ni, "Assessing semantic information in convolutional neural network representations of images via image annotation," *Proceedings - International Conference on Image Processing, ICIP*, vol. 2016-August, pp. 2266–2270, Aug. 2016, doi: 10.1109/ICIP.2016.7532762.

[51] M. Guillaumin, T. Mensink, J. Verbeek, and C. Schmid, "TagProp: Discriminative metric learning in nearest neighbor models for image auto-annotation," *Proceedings of the IEEE International Conference on Computer Vision*, pp. 309–316, 2009, doi: 10.1109/ICCV.2009.5459266.

[52] Y. Verma and C. v. Jawahar, "Image Annotation Using Metric Learning in Semantic Neighbourhoods," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 7574 LNCS, no. PART 3, pp. 836–849, 2012, doi: 10.1007/978-3-642-33712-3_60.

[53] KrizhevskyAlex, SutskeverIlya, and H. E., "ImageNet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, May 2017, doi: 10.1145/3065386.

[54] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, Sep. 2014, Accessed: Aug. 29, 2021. [Online]. Available: https://arxiv.org/abs/1409.1556v6

[55] V. N. Murthy, S. Maji, and R. Manmatha, "Automatic image annotation using deep learning representations," *ICMR 2015 - Proceedings of the 2015 ACM International Conference on Multimedia Retrieval*, pp. 603–606, Jun. 2015, doi: 10.1145/2671188.2749391.

[56] "The COREL Database for Content based Image Retrieval - dct-research." https://sites.google.com/site/dctresearch/Home/content-based-image-retrieval (accessed Sep. 19, 2021).

[57] "ESP game - Wikipedia." https://en.wikipedia.org/wiki/ESP_game (accessed Sep. 19, 2021).

[58] J. Wang, Y. Yang, J. Mao, Z. Huang, C. Huang, and W. Xu, "CNN-RNN: A Unified Framework for Multi-label Image Classification," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2016-December, pp. 2285–2294, Apr. 2016, Accessed: Aug. 30, 2021. [Online]. Available: https://arxiv.org/abs/1604.04573v1

[59] J. Jin and H. Nakayama, "Annotation order matters: Recurrent Image Annotator for arbitrary length image tagging," *Proceedings - International Conference on Pattern Recognition*, vol. 0, pp. 2452–2457, Jan. 2016, doi: 10.1109/ICPR.2016.7900004.

[60] H. Hu, G. T. Zhou, Z. Deng, Z. Liao, and G. Mori, "Learning Structured Inference Neural Networks with Label Relations," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2016-December, pp. 2960–2968, Dec. 2016, doi: 10.1109/CVPR.2016.323.

[61] T.-Y. Lin *et al.*, "Microsoft COCO: Common Objects in Context," May 2014, Accessed: Aug. 31, 2021. [Online]. Available: https://arxiv.org/abs/1405.0312v3

[62] "COCO - Common Objects in Context." https://cocodataset.org/#explore (accessed Aug. 31, 2021).

[63] "GitHub - cocodataset/cocoapi: COCO API - Dataset @ http://cocodataset.org/." https://github.com/cocodataset/cocoapi (accessed Sep. 02, 2021).

[64] "Convolutional neural network - Wikipedia." https://en.wikipedia.org/wiki/Convolutional_neural_network (accessed Sep. 05, 2021).

[65] "Convolutional Neural Networks. The motive of this blog is to explain… | by Rijul Vohra | DataDrivenInvestor." https://medium.datadriveninvestor.com/convolutional-neural-networks-3b241a5da51e (accessed Sep. 07, 2021).

[66] "Universal approximation theorem - Wikipedia." https://en.wikipedia.org/wiki/Universal_approximation_theorem (accessed Sep. 05, 2021).

[67] "Convolutional Neural Networks (CNNs) and Layer Types - PyImageSearch." https://www.pyimagesearch.com/2021/05/14/convolutional-neural-networks-cnns-and-layer-types/ (accessed Sep. 06, 2021).

[68] I. Goodfellow, Y. Bengio, and A. Courville, "Deep Learning - whole book," *Nature*, vol. 521, no. 7553, p. 800, 2016, Accessed: Sep. 05, 2021. [Online]. Available: http://goodfeli.github.io/dlbook/%0Ahttp://dx.doi.org/10.1038/nature14539

[69] "CNN | Introduction to Pooling Layer - GeeksforGeeks." https://www.geeksforgeeks.org/cnn-introduction-to-pooling-layer/ (accessed Sep. 05, 2021).

[70] "Artificial Neural Networks – Part 2: MLP Implementation for XOr – Machine Learning and Optimisation." http://www.mlopt.com/?p=207 (accessed Sep. 06, 2021).

[71] "A Gentle Introduction to Dropout for Regularizing Deep Neural Networks." https://machinelearningmastery.com/dropout-for-regularizing-deep-neural-networks/ (accessed Sep. 07, 2021).

[72] "Loss Functions — ML Glossary documentation." https://ml-cheatsheet.readthedocs.io/en/latest/loss_functions.html#cross-entropy (accessed Sep. 06, 2021).

[73] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2016-December, pp. 770–778, Dec. 2015, Accessed: Sep. 09, 2021. [Online]. Available: https://arxiv.org/abs/1512.03385v1

[74] "Welcome to Colaboratory - Colaboratory." https://colab.research.google.com/notebooks/intro.ipynb (accessed Sep. 10, 2021).

[75] “Kaggle: Your Home for Data Science.” https://www.kaggle.com/ (accessed Sep. 10, 2021).

[76] “Chollet, F. & others, 2015. Keras. Available at: https://github.com/fchollet/keras.”.

[77] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, Dec. 2014, Accessed: Sep. 10, 2021. [Online]. Available: https://arxiv.org/abs/1412.6980v9