



**HAL**  
open science

# Vehicle identity recognition using deep learning

Mohamed Dhia Elhak Besbes

► **To cite this version:**

Mohamed Dhia Elhak Besbes. Vehicle identity recognition using deep learning. Library and information sciences. CY Cergy Paris Université; Université de Sfax (Tunisie), 2021. English. NNT: 2021CYUN1020 . tel-03381928

**HAL Id: tel-03381928**

**<https://theses.hal.science/tel-03381928>**

Submitted on 18 Oct 2021

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# Thèse de doctorat

pour l'obtention du titre de  
Docteur en Sciences et Technologies de l'Information et de la Communication  
délivré par

**l'Université de Cergy-Pontoise**

Ecole doctorale n° 417: SCIENCES ET INGÉNIERIE(SI)

## RECONNAISSANCE DE L'IDENTITE DE VEHICULES PAR APPRENTISSAGE PROFOND

présentée et soutenue publiquement par

**Mohamed Dhia Elhak Besbes**

le 30 06 2021

Directeur de thèse : Monsieur Hedi TABIA

Co-Directeur de thèse: Monsieur Bassem BEN HAMED

### **Jury**

M. Hazem WANNOUS	Professeur, à Université IMT Lille Douai	Président
M. Aladine CHTOUANI	Maitre de conférences, à Université Orléan	Rapporteur
M. Mourad ZAIED	Professeur, à Université de Gabes	Rapporteur
M. Yousri KESSENTINI	Maitre de conférences, à CRNS	Examineur
Mme. Raoudha BEN DJEMAA	Maitre de conférences, à Université de Sousse	Examinatrice

## Declaration

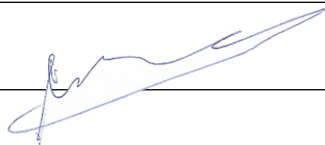
All sentences or passages quoted in this document from other people's work have been specifically acknowledged by clear cross-referencing to author, work and page(s). Any illustrations that are not the work of the author of this report have been used with the explicit permission of the originator and are specifically acknowledged. I understand that failure to do this amounts to plagiarism and will be considered grounds for failure.

Name: **Mohamed Dhia Elhak BESBES**

---

Signature:

---



Date: **12/07/2021**

---

## Acknowledgement

Firstly, I would like to express my sincere gratitude to my advisors Prof. Hedi TABIA and Prof. Bassem BEN HAMED for the continuous support of my Ph.D study and related research, for their patience, motivation, and immense knowledge. Their guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my Ph.D study. Besides my advisor, I would like to thank the rest of my thesis committee: Prof. Mourad ZAIED, Dr. Aladine CHETOUANI, and Dr. Raoudha BEN DJEMAA, for their insightful comments and encouragement, but also for the hard question which incited me to widen my research from various perspectives.

My sincere thanks also goes to Dr. Yousri KESSENTITNI, who provided me an opportunity to join their team as intern, and who gave access to the laboratory and research facilities. Without their precious support it would not be possible to conduct this research. I thank my fellow lab mates for the stimulating discussions, for the sleepless nights we were working together before deadlines, and for all the fun we have had in the last three years. Also I thank my friends in the following institutions CRNS and ENSEA. Last but not the least, I would like to thank my family: my parents and to my brothers and sisters for supporting me spiritually throughout writing this thesis and my life in general.

# Abstract

With the continuous economic growth, the number of motor vehicles is increasing, the role of Intelligent Transportation Systems (ITS) in managing roads logistics and safety is getting less practical. There is a dire need for new methods to handle the ever-growing pressure. Among various promising techniques, we consider in this work deep learning. The vast amount of data at our disposal allow us to better exploit the potential of this technology. Surveillance video data provides a wealth of information with the tremendous amount of videos accumulated over the years. In this thesis, vision-based methods and learning paradigms for vehicle identification are extensively investigated, and improved upon by several novel approaches that we propose, to achieve better results and overcome previous weaknesses.

Traditional machine learning consisted of hand-crafted feature extraction. In contrast, deep neural networks learn hierarchical feature representations from lines and curves to objects inducing more robustness and precision. Deep Learning for visual features has attracted much attention in the last few years. Many applications have seen a leap forward in terms of performance, such as image classification, object detection, image generation, and image segmentation, to name a few. In this thesis, deep models are mainly investigated for vehicle identification.

License Plate Detection and Recognition is a fundamental task in vehicle identification, but it faces many challenges mainly because the license plate occupies a small space of the image, making it sensitive to variation in image quality, illumination, view-points, and occlusions. Moreover, license plates are specific to each country. Chinese license plates, for instant, contain Chinese characters, European license plates have Latin letters. Moreover, the design changes from one plate to another with complex backgrounds and different sizes, further complicating the task. To address these difficulties, a two-stage multi-norm system is proposed. It detects the license plate using both the features inside the license plate and its surroundings. This is because a license plate is placed in the same place for all vehicles. Then, We handle the characters on the license plate as objects instead of characters removing the need for segmentation and allowing our system to be robust in recognizing multi-norm license plates since characters are recognized even when not aligned.

Vehicle Make and Model Classification complement license plate recognition since

the latter might not be visible for recognition. It is a challenging problem due to the large number of classes and the minor inter-class variations. In other words, Vehicle make and model recognition is both a coarse and fine-grained classification problem: On one hand, vehicles can have unconstrained poses when taken under multiple view-points. Classification under such conditions can be seen as a coarse-grained problem. On the other hand, the unique hierarchical structure starting from make, model, to year of manufacture produces very similar vehicles with a subtle inter-class variation. These challenges were addressed with a multi-stream robust architecture to extract and combine both local and global features representations: First, a pre-trained detector crops out vehicle parts where the number of detected parts varies from one image to another, hence the robustness of the system. Second, a selection process filters out the vehicle image with some detected parts to optimize performance. Then, every part goes through a different stream into a specialized feature extractor allowing the system to detect subtle inter-class variations. Finally, all extracted features are aggregated through a novel fusion technique.

When the license plate is occluded, and the task is to re-identify a vehicle, make and model classification falls short of the task. Thus we address the Vehicle re-identification (V-Reid) problem without LP recognition. V-Reid aims to automatically find vehicle identity from a large number of vehicle images captured from multiple cameras. Most existing V-Reid approaches rely on fully supervised learning, where large amounts of annotated training data are required. In practice, massive data annotation is an expensive task and may be impossible with real-time learning and identification, in which semi or unsupervised learning is needed. We focus our interest on semi-supervised V-Reid, where each identity has a single labeled and multiple unlabeled samples in training. We propose a framework that gradually labels vehicle images taken from surveillance cameras. Our framework is based on a deep Convolutional Neural Network (CNN), which is progressively learned using a feature anchoring regularization process. The experiments conducted on various publicly available datasets demonstrate the efficiency of our framework in V-Reid tasks. Our approach with only 20% labeled data shows interesting performance compared to the state-of-the-art supervised methods trained on fully labeled data.

With new technologies like drones and self-driving vehicles becoming popular, more challenging datasets are becoming available. Hence, we propose a cross camera V-Reid, a system that performs re-identification from different cameras. Usually, vehicle images captured from different cameras exhibit large intra-class variability and do not follow the same distribution. To alleviate this problem, our system models the cross cameras re-identification as a domain adaptation problem. It uses optimal transport to transfer knowledge from different cameras and project vehicle features onto a shared space in which the same vehicle identities are close. The similarities are computed using the euclidean distance in the joint space. We conducted a set of experiments on the two publicly available datasets VeRi776 and VehicleID.

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Vision-based Intelligent Transportation Systems . . . . .	1
1.1.1	Deep Visual Feature Learning . . . . .	2
1.1.2	Learning Paradigms . . . . .	3
1.2	Research Motivation . . . . .	4
1.3	Problem Restatement . . . . .	4
1.4	Licence Plate Recognition . . . . .	5
1.5	Vehicle Make/Model Classification . . . . .	6
1.6	Vehicle Re-Identification . . . . .	7
1.7	Outline . . . . .	8
<b>2</b>	<b>Research Preliminary</b>	<b>10</b>
2.1	Visual Feature Representation . . . . .	10
2.2	Deep Visual Feature Learning . . . . .	10
2.2.1	Convolutional Neural Network . . . . .	10
2.2.2	Recurrent Neural Network . . . . .	14
2.3	Learning Paradigms . . . . .	16
2.3.1	Machine Learning Paradigms . . . . .	16
2.3.2	Progressive & Curriculum Learning . . . . .	22
<b>3</b>	<b>LP Detection and Recognition</b>	<b>24</b>
3.1	Literature Survey . . . . .	24
3.1.1	Object Detection . . . . .	24
3.1.2	Optical Character Recognition . . . . .	26
3.2	Datasets and Evaluation Protocols . . . . .	29
3.2.1	GAP-LP Dataset . . . . .	29
3.2.2	RADAR Dataset . . . . .	30
3.3	Material and Methodology . . . . .	30
3.4	Results and Discussion . . . . .	38
3.4.1	Comparison with state-of-the-arts ALPR systems . . . . .	42
3.4.2	Runtime Evaluation . . . . .	42
3.5	Conclusion . . . . .	43

<b>4</b>	<b>Vehicle Make and Model Classification</b>	<b>44</b>
4.1	Literature Survey . . . . .	44
4.2	Datasets and Evaluation Protocols . . . . .	45
4.2.1	The Comprehensive Cars (CompCars) Dataset . . . . .	45
4.2.2	Other datasets . . . . .	46
4.3	Material and Methodology . . . . .	48
4.3.1	Parts qualitative study . . . . .	53
4.4	Results and Discussion . . . . .	56
4.4.1	Results from Individual Parts . . . . .	56
4.4.2	Multi-Stream Dynamic Fusion . . . . .	57
<b>5</b>	<b>Vehicle Re-Identification</b>	<b>59</b>
5.1	Literature Survey . . . . .	59
5.1.1	Object Re-identification . . . . .	59
5.2	Datasets and Evaluation Protocols . . . . .	60
5.2.1	PKU VehicleID Dataset . . . . .	61
5.2.2	VeRi-776 Dataset . . . . .	61
5.2.3	VeRi-Wild Dataset . . . . .	61
5.3	Material and Methodology . . . . .	63
5.3.1	Progressive One-Shot/Few-Shots Vehicle Re-Identification . . . . .	63
5.3.2	Cross Camera Vehicle Re-Identification . . . . .	65
5.4	Results and Discussion . . . . .	69
5.4.1	Progressive One-Shot/Few-Shots Vehicle Re-Identification . . . . .	69
5.4.2	Cross Camera Vehicle Re-Identification . . . . .	76
<b>6</b>	<b>Conclusions and Future Work</b>	<b>80</b>
6.1	Conclusions . . . . .	80
6.2	Future Work . . . . .	81
6.2.1	The Lottery Ticket Hypothesis . . . . .	81
6.2.2	Zero-Shot Learning . . . . .	81
6.2.3	Meta-Learning . . . . .	81
6.2.4	Self-Driving cars . . . . .	82
	<b>Appendices</b>	<b>97</b>
<b>A</b>	<b>Deep Learning Models</b>	<b>98</b>
A.1	AlexNet . . . . .	98
A.2	VGG-16 . . . . .	99
A.3	GoogLeNet (Inceptionv1) . . . . .	99
A.4	ResNet . . . . .	101



<b>B</b>	<b>Deep Learning ToolBox</b>	<b>103</b>
B.1	Caffe . . . . .	103
B.2	TensorFlow . . . . .	103
B.3	Keras . . . . .	104
B.4	PyTorch . . . . .	105

# List of Figures

1.1	CNN’s hierarchical features representation of human faces from edges to facial structure. . . . .	2
1.2	Transfer learning from SVHN to MNIST. . . . .	3
1.3	Illustrations of the coarse-to-fine concept of vehicle recognition at different granularity. . . . .	5
1.4	License Plate Recognition task is composed of three consecutive sub-tasks: (1) Vehicle Detection => (2) LP Detection => (3) LP Recognition. . . . .	6
1.5	Make and Model Classification three most researched approaches: (1) Global predictor (2) Part-Based predictor (3) Attention-Based predictor. . . . .	7
1.6	V-reID can be addressed as an image retrieval problem. Given one probe image from an arbitrary camera viewpoint (1), the V-Reid consists of seeking the most relevant (based on a similarity metric) image from the gallery (2). The gallery is generally formed from a multi-camera network. . . . .	8
2.1	Filters are convolved over the image or a feature map in a sliding window fashion. . . . .	11
2.2	Average pooling and Max pooling methods. . . . .	12
2.3	Recurrent neural network architecture. . . . .	14
2.4	Long Short-term Memory architecture. . . . .	15
2.5	Distribution obeying both smoothness and low-density assumptions. . . . .	17
2.6	Distribution obeying the manifold assumption. . . . .	17
2.7	Example of Exclusive Clustering. . . . .	18
2.8	Example of Overlapping Clustering. . . . .	19
2.9	Example of Hierarchical Clustering. . . . .	19
2.10	Transfer learning using fine-tuning. . . . .	20
2.11	The two methods for fine-tuning a deep convolutional network. . . . .	21
2.12	The MNIST dataset (left) and SVHN dataset (right) share the same labels (0, 1, 2, 3, 4, 5, 6, 7, 8, 9) but the samples are different. . . . .	21
3.1	Differences between text and scene text recognition. . . . .	27
3.2	GAP-LP sample images and their respective plates . . . . .	29
3.3	Radar image samples, LPs blurred for privacy . . . . .	30

3.4	Illustration of our LP detection/recognition system based on YOLOv2 and CNN. The output of the YOLOv2 detection module becomes the input of both the CRNN recognition module and the YOLOv2 detection/recognition module. . . . .	31
3.5	Architecture of the segmentation-free LP recognition engine . . . . .	34
3.6	LP recognition pipeline using YOLO network . . . . .	34
3.7	Background changes in LPs. . . . .	38
3.8	Background changes in LPs. . . . .	38
3.9	Synthesized images of LPs . . . . .	40
3.10	Qualitative results obtained by the proposed ALPR system in the GAP-LP and Radar datasets. Green text refers to correctly recognized LP, while miss-recognized LP are written in red. . . . .	41
4.1	Web (left) and Surveillance (right) datasets. . . . .	46
4.2	The Cars dataset. . . . .	47
4.3	The VMRRdb dataset. . . . .	47
4.4	The BVMMR Dataset. . . . .	48
4.5	system's main steps. YOLO first processes the image to detect the vehicle parts. Global and local features are then extracted from the full vehicle image and the selected parts using VGG. The global and the local representations are then fed to the dynamic fusion layer to perform the final classification. . . . .	49
4.6	Examples of subtle differences between vehicle models. From a global perspective the vehicles seem similar, yet they belong to different vehicle models. This is due to very subtle variations. . . . .	50
4.7	Model architecture. Initially, parts are introduced to the model sequentially from a single input. The batch $B$ is composed of $n$ mini-batches $\{b_1, b_2, \dots, b_n\}$ , $n$ being the number of parts, each mini-batch contains similar parts $\{p_1, p_2, \dots, p_k\}$ for $k \leq n$ so the batch size is $k \times n$ . The images are passed through the first four convolutional block then the batch $B$ is split into $n$ groups $\{\{p_{11}, p_{12}, \dots, p_{1n}\}, \{p_{21}, p_{22}, \dots, p_{2n}\}, \dots, \{p_{k1}, p_{k2}, \dots, p_{kn}\}\}$ each image in the group is passed to a part-specific fifth convolutional block. Finally, the image and its parts features are aggregated with the dynamic fully-connected layer. . . . .	52
4.8	Models yearly production in the US in 2000-2020 . . . . .	53
4.9	Logo of the vehicle's mark on the grilles. . . . .	54
4.10	Logo of the vehicle's mark and the model the bumper. . . . .	54
4.11	Air flux smoothly going over a vehicle. . . . .	54
4.12	Logo of the vehicle's mark and the model the bumper. . . . .	55
4.13	Front and Rear lights with little room for design variation. . . . .	55
4.14	Same design proportions across different makes and models. . . . .	56
5.1	Samples from the VehicleID dataset. . . . .	61

5.2	Samples from the VeRi-776 dataset. . . . .	61
5.3	Samples from the VeRi-Wild dataset. . . . .	62
5.4	Method Overview: First, the training stage starts by learning the vehicle identities from labeled data and a cross-entropy loss function (1). Second, unlabeled data are projected to the feature space induced by the CNN penultimate layer. A selected number of unlabeled data close to the originally labeled samples is given pseudo-labels (2). A mean squared loss is further computed between pseudo labeled and the originally labeled data. This loss is backpropagated until narrowing the gap between labeled and pseudo-labeled data (3). . . . .	64
5.5	Different vehicles with same model. . . . .	65
5.6	The input fed into our model, in the training phase, is a batch composed of a pair of vehicle images captured using different cameras with the same identity. We use Resnet-50 as a feature extractor. A loss function based on the optimal transport is then used to minimize the discrepancy between extracted features. In the V-reID phase, we use the trained model to extract features from each vehicle image. The Euclidean distance is used to compute the similarities between extracted features. . . . .	66
5.7	Transporting source images to a representation that reduces the euclidean distance between their respective features. . . . .	67
5.8	Latent space distribution: each training step is composed of two planes (points alignment) each representing features of images from same camera. $d^*$ are distances to a correct V-reID while $d$ are distances to incorrect V-reID. Naturally, V-reID from the same camera (same plane) has $d^* < d$ however, this is not the case from one plane to another where $d^* > d$ . The optimal transport aims to project the two planes on a latent space where $d^* < d$ . . . . .	68
5.9	The CMC comparisons on VehicleID-2400 test set. . . . .	71
5.10	The CMC comparisons on VeRi-776 test set. . . . .	76
A.1	AlexNet Architecture. . . . .	98
A.2	VGG16 Architecture. . . . .	99
A.3	GoogleLeNet Architecture. . . . .	100
A.4	GoogleLeNet Inception Module. . . . .	100
A.5	Residual Block. . . . .	101
A.6	ResNet50 Architecture. . . . .	102
B.1	Tensorflow as an engine to Keras. . . . .	104
B.2	PyTorch Vs Tensorflow in terms of paper publications. . . . .	105

# List of Tables

3.1	The network configuration of the convolutional recurrent neural network k : filter size, s : stride, p: padding . . . . .	35
3.2	Number per iteration and percentage (the total of all iterations) of annotated LP images on GAP-LP dataset . . . . .	37
3.3	Number per iteration and percentage (the total of all iterations) of annotated LP images on Radar dataset . . . . .	37
3.4	LP Detection: precision and recall rates for IoU threshold of 0.5 . . . . .	39
3.5	LP Detection: recall rates for IoUs ranging from 0.3 to 0.8 . . . . .	39
3.6	LP and character recognition rates obtained by CRNN on GAP-LP dataset	40
3.7	LP and character recognition rates obtained by CRNN on Radar dataset	40
3.8	LP recognition rates obtained by CRNN using generated data . . . . .	40
3.9	Comparison of plate detection rates on three subsets of the AOLP dataset	42
3.10	Comparison of plate Recognition rates on three subsets of the AOLP dataset . . . . .	43
3.11	The computational time required in each ALPR stage . . . . .	43
4.1	Quantity distribution in different view ports. . . . .	46
4.2	Individual front parts of CompCars’s web data. . . . .	57
4.3	Individual rear parts of CompCars’s web data. . . . .	57
4.4	Fusion using different combinations on the CompCars’s web data. FB: Front Bumper, FH: Front Hood, FG: Front Grilles, FLL: Front Left Light, FRL: Front Right Light, RB: Rear Bumper, RT: Rear Trunk, RLL: Rear Left Light, RRL: Rear Right Light . . . . .	58
4.5	Comparison with our approach. . . . .	58
5.1	Comparisons between the VehicleID [74], the VeRi-776[79] and the VERI-WILD[81] . . . . .	62
5.2	Performance on VehicleID dataset gallery size = 1600. . . . .	70
5.3	Performance on VehicleID gallery size = 2400 dataset. . . . .	72
5.4	Performance on VeRi-Wild gallery size = 3000 (small gallery) . . . . .	73
5.5	Performance on VeRi-Wild gallery size = 5000 (medium gallery) . . . . .	73
5.6	Performance on VeRi-Wild gallery size = 10000 (large gallery). . . . .	74
5.7	Performance on VeRi-776 dataset . . . . .	75

5.8	Ablation Study on VeRi-776 dataset . . . . .	77
5.9	Ablation Study on VehicleID dataset gallery size = 1600. . . . .	77
5.10	Ablation Study on VehicleID dataset gallery size = 2400. . . . .	77
5.11	Performance on VeRi-776 dataset . . . . .	78
5.12	Performance on VehicleID dataset gallery size = 1600. . . . .	79
5.13	Performance on VehicleID gallery size = 2400 . . . . .	79

# Chapter 1

## Introduction

The use of vehicles in our life is increasing exponentially due to rapid economic development. Hence, maintaining the ease of transport and road security is essential for many reasons: companies move their goods on the road, public transport is expected to arrive on time, and vehicle owners can be the target of hijacking. Hence there is a dire need for high-performing Intelligent Transportation Systems (ITS).

ITS is where information and communication technologies are applied in the field of road transport, including infrastructure, vehicles, and users, and in traffic management.

ITS uses a range of cellular data, such as the mobile triangulation method, where phones periodically transmit their presence information to the mobile phone network. The data is then converted into traffic flow information. Bluetooth and RFID detectors are also used by mounting them on the side poles.

Another more current source of information is smartphones. They offer a rich set of sensory data from GPS to accelerometers to track traffic speed and density. Finally, radar is a very common sensor to monitor roads, and it's primarily used in highways but not exclusively. In this thesis, however, we focus on vision-based ITS.

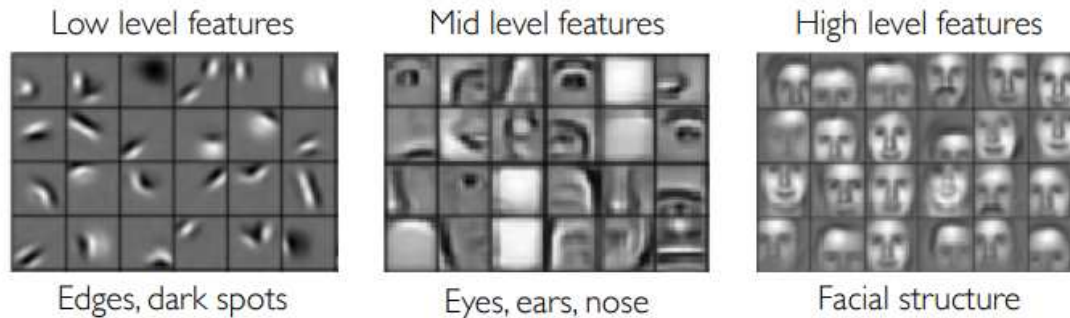
### 1.1 Vision-based Intelligent Transportation Systems

Video surveillance cameras are the most relied-on source of data since they convey an accurate depiction of reality. However, the size of recorded videos makes analyzing it a tedious and sometimes impossible task for humans. Hence, automatic video analysis acquired a significant interest from researchers and engineers. Surveillance tasks such as, incident detection Oskarbski et al. [95] and vehicle counting [5, 96, 2] are either addressed by sensor-based or vision-based algorithms. Furthermore, visual data contains an abundance of patterns that can be analyzed and interpreted for more advanced tasks like vehicle detection and license plate recognition. In contrast, radar sensor-based techniques can only detect vehicles in a minimal area, while video-based

techniques can detect multiple vehicles simultaneously. However, no video analysis algorithm came close to the human level of vehicle detection, classification, or any other naturally easy task for humans. All of that radically changed with the emergence of CNN. Inspired by biological processes, the connectivity pattern between neurons resembles the organization of the animal visual cortex. CNN can mimic the brain's capacity to analyze visual data and, in some cases, surpass it.

### 1.1.1 Deep Visual Feature Learning

Deep Features refers to the hierarchical nature of extracted features. CNN's are composed of layers stacked one after the other. The first layers extract fundamental features like lines and curves while layers at the end recognize shapes and objects, as is shown in Figure 1.1.



**Figure 1.1:** *CNN's hierarchical features representation of human faces from edges to facial structure.*

Deep CNN layers usually consist of several convolutional layers, pooling layers, fully-connected layers, and normalization layers. Each is designed to learn a different level of representation in the hierarchy. A drawback of this architecture is that the more a CNN has layered, the more it is computationally expensive. Residual Neural Networks (ResNets) [45] are a class of CNN wherein training it learns to skip or shortcut layers. A CNN learning process requires data, images, or videos in computer vision and ground-truth. For example, given an image of a vehicle, the ground truth is the make and model in the context of make and model classification. The next section introduces learning paradigms that categorize the relationship between training and test sets and labeled and unlabeled samples.



### 1.1.2 Learning Paradigms

Machine Learning paradigms can be coarsely categorized into supervised learning, unsupervised learning, and more recently, one-shot or few-shots learning with other related concepts such as transfer learning and domain adaptation. The most common is supervised learning, where all the samples in the training set are labeled with ground truth. In contrast, unsupervised learning has no labeled samples, so a CNN learns how to label the training set, unsupervised learning can be further grouped into clustering as shown by Greene et al. [40] and association problems by Frawley et al. [29]. In between, the few-shot and one-shot setups are when a CNN is given only a few images or only one to learn the features and accomplish the task.

Deep unsupervised learning adopts layer-wise training to learn statistical structure or dependencies of the unlabeled data. For example, deep-stacked denoising auto-encoder by Pascal et al. [121], deep belief nets by Hinton, Osindero, and Teh [48] and sparse coding proposed by Jenatton et al. [58].

Domain adaptation can be viewed as a special case of transfer learning (Patel, Li, and Chellappa [97]). Transfer learning is defined as the learning scenario where a model is trained on a source domain or task and evaluated on a different but related target domain or task, where either the tasks differ. For example, learning a model on a handwritten digit dataset MNIST [67] to use it to recognize street numbers SVHN [38] or the opposite as shown in Figure 1.2.



**Figure 1.2:** *Transfer learning from SVHN to MNIST.*

These types of learning are often referred to as Machine learning paradigms, not to be confused with learning paradigms that refer to learning in general, i.e., for machines and humans. Learning paradigms found their way to machine learning by projecting theories that worked on humans and animals onto deep nets concepts. For instant, curriculum learning presented by Bengio, Louradour, Collobert, and Weston et al. [8] consists of the premise that humans and animals learn much better when the examples are not randomly presented but organized in a meaningful order which illustrates gradually more concepts, and gradually more complex ones. Works such as [42, 8]

demonstrates the influence of curriculum learning on deep neural networks. A second example is the progressive learning paradigm, where instead of a labeled training set, the learning algorithm trains to progressively label the training set by assigning pseudo-labels then train on the pseudo-labeled set. this approach minimizes the annotation afford and actually enhances results.

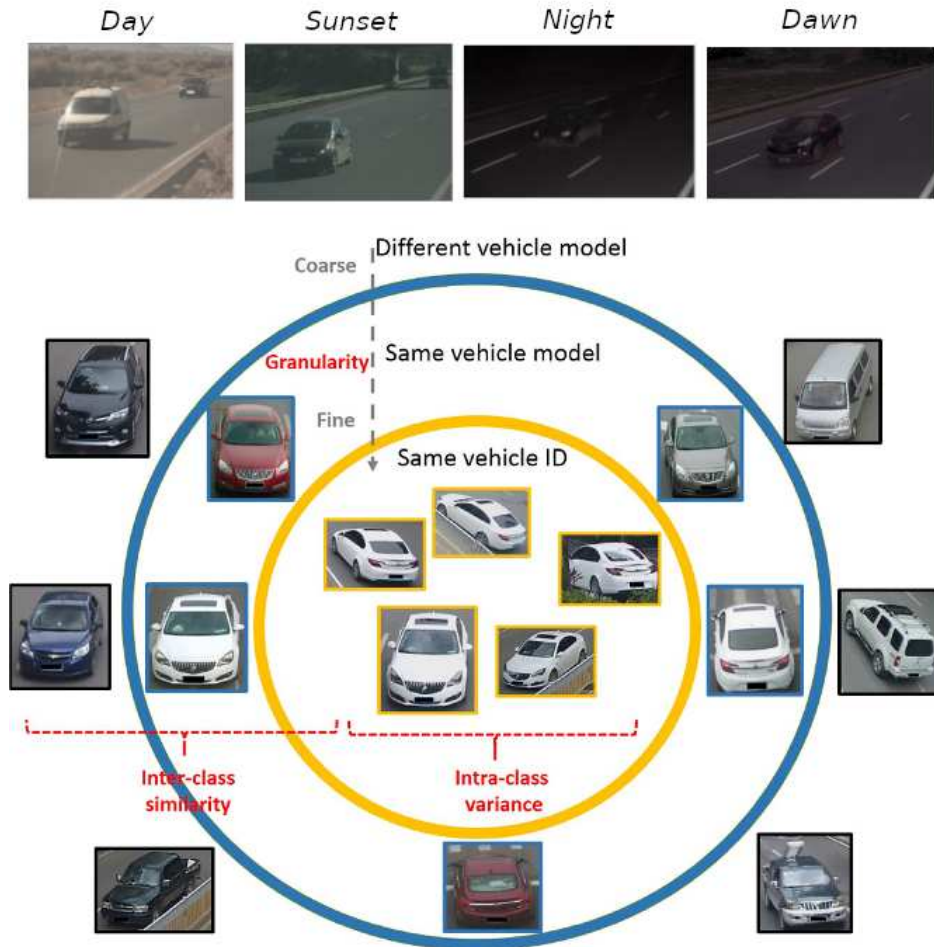
Therefore, in this thesis, several computer vision models are extensively studied to achieve better results and overcome previous weaknesses such as learning from fewer examples, generalized solution cross countries norms, and robustness to challenging conditions.

## 1.2 Research Motivation

Our brain's capability to analyze visual data is impressive; however it remains limited by the visual input, we can only analyze what we see and only at a certain speed. Computers, on the other hand, can analyze data at an astonishing speed. Combining the best of two worlds will enable us to improve our technologies at an unprecedented pace. Vehicle Identification represents a key challenge in improving computer vision in general for even humans struggle in the vehicle identification task; thus, working on this problem will have a security and economic impact on our society and push the body of knowledge forward.

## 1.3 Problem Restatement

The vehicle identification problem presents some unique challenges. Figure 1.3 shows, on the one hand, how vehicles with the same models can be very different, and the reverse is also true. Vehicles of different models can look similar. On the other hand, the same identity vehicles look radically different from multiple viewpoints. Furthermore, the license plate (The unique identifier of a vehicle) occupies a very small space of the images making recognizing the text a very hard task. Plus, vehicle re-ID can be between images at different times of the day and even different weathers. Hence, robustness is a vital criterion for an effective vehicle V-Reid system.



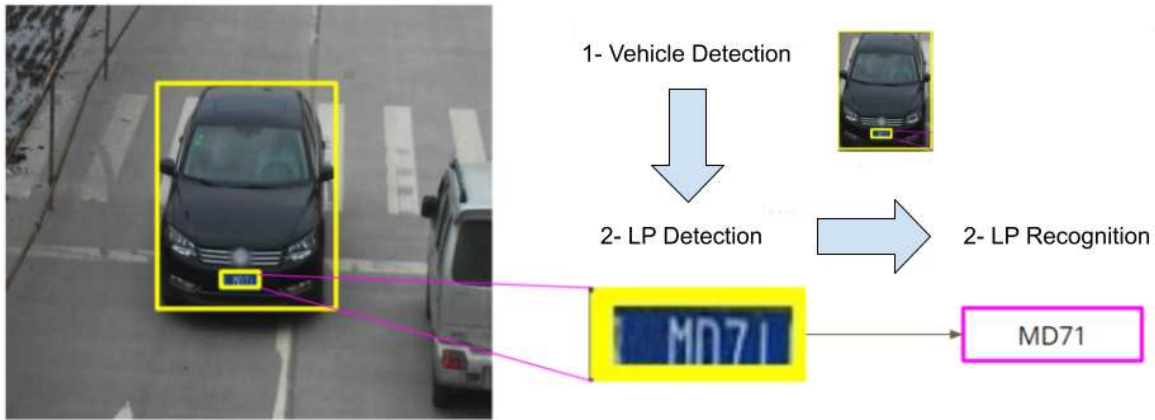
**Figure 1.3:** Illustrations of the coarse-to-fine concept of vehicle recognition at different granularity.

The purpose of this thesis is to develop a system for vehicle identity recognition based on both the recognition of the number of the license plate (LP) and the brand (manufacturer, model) of the vehicle. The main objective is to propose solutions to improve the mobility and the safety of road traffic by, for instant, the surveillance of the border crossings and the search of suspicious vehicles.

## 1.4 Licence Plate Recognition

Automatically identifying vehicles through their LPs proves to be a solution of a significant role in this active world. The Automatic License Plate Recognition (ALPR) is a field of research that gained a lot of interest during the last decade with many applications in ITS along with the improvement of digital cameras and the increase

in computational capacity. These systems aim to identify vehicles through their LPs. They provide automatic detection and recognition of the vehicle's LP within a real view camera scene. Once the camera takes an image of the front of the vehicle, this image is given as an input to processing algorithms to analyze it, locate and extract the plate regions from the background. The recognition phase consists of the segmentation of the characters within the detected region and then recognition as shown in figure 1.4.

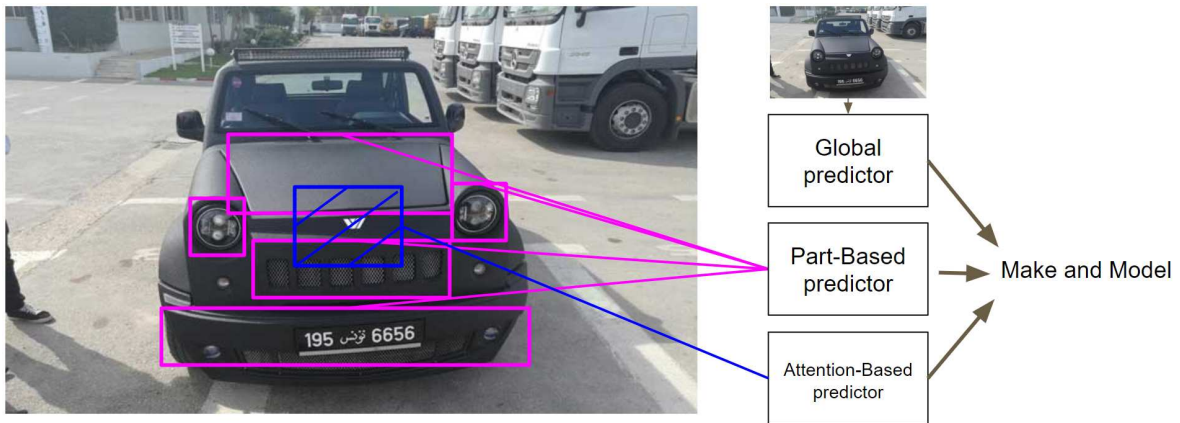


**Figure 1.4:** License Plate Recognition task is composed of three consecutive sub-tasks: (1) Vehicle Detection => (2) LP Detection => (3) LP Recognition.

## 1.5 Vehicle Make/Model Classification

There are a wide variety of vehicle models available. Every vehicle manufacturer is striving to release new vehicle models every year with improved styling in the competitive environment. Classifying vehicle models has become a topic of interest in ITS. The problem with identifying vehicle models is the appearance in different views. Many of the models carry some of the old features from the previous models.

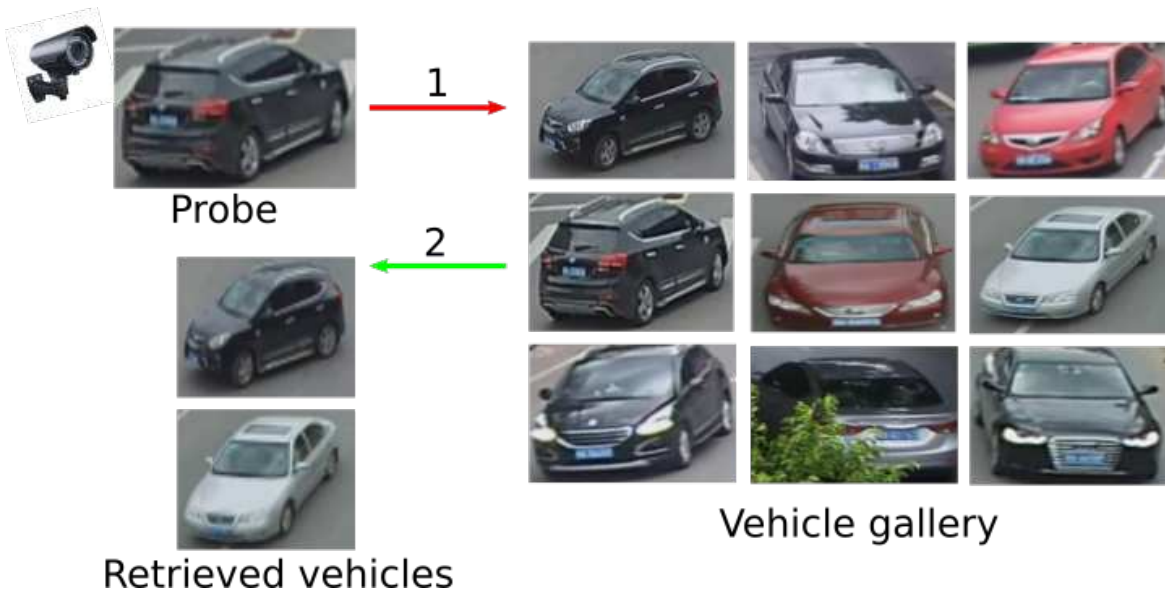
Most mainstream approaches are at least in one of these three categories shown in Figure 1.5: (1) Global predictors where the entire image is fed to the system to be analyzed and create a global representation (2) Part-Based predictors detects parts of the vehicle then analyze and create a representation of each part, and an aggregation method is used for a final prediction. (3) Attention-Based predictors learn to find a Region Of Interest (ROI) then predict by analyzing that region.



**Figure 1.5:** *Make and Model Classification three most researched approaches: (1) Global predictor (2) Part-Based predictor (3) Attention-Based predictor.*

## 1.6 Vehicle Re-Identification

V-Reid is the task of recognizing a vehicle, captured by one or more cameras, over a range of candidates targets. Figure 1.6 shows the process of V-Reid where a prob (the search for) image is given as an input and a ranking of possible same identity vehicles are the output.



**Figure 1.6:** *V-reID can be addressed as an image retrieval problem. Given one probe image from an arbitrary camera viewpoint (1), the V-Reid consists of seeking the most relevant (based on a similarity metric) image from the gallery (2). The gallery is generally formed from a multi-camera network.*

The re-identification problem was initially proposed for persons then extended to vehicles. Although they share some challenges, V-Reid poses its own difficulties due to the intra-class variance and inter-class similarity. Common challenges like illumination that can change from one camera to another, different resolutions, and quality between cameras are also factors. Plus, in traffic, vehicles can be occluded. Furthermore, machine learning algorithm requires labeled data to train, so humans need to annotate thousands of hours of camera surveillance. For good generalization capabilities, complex algorithms need to be taught with a large number of labeled data.

## 1.7 Outline

The rest of the thesis is structured as follows. In chapter 2, we present a preliminary review of theories and methods relevant to deep visual learning. We review the basics of convolutional and recurrent neural networks and their applications. Then approaches of learning paradigms applied on deep learning models are presented and improved upon. In chapter 3, we present a literature survey for license plate detection and recognition starting from object detection methods, including traditional detection frameworks and deep learning-based detectors are explored. We, of course, review the evaluation metrics for detection and approaches specialized for vehicle and LP detection. Optical Character Recognition (OCR) methods are reviewed for LP recog-

dition. Then we present datasets and evaluation protocols relevant to our work on LP detection and recognition. Then, we present, first, an ALPR system for multilingual LP detection and recognition in natural scene images. The system architecture uses a pipeline with two deep learning stages. The first network was trained to detect LPs on the whole raw image by using the latest state-of-the-art deep learning-based detector, namely YOLOv2. The second stage is then applied to the cropped image to recognize captured LP photographs. Two recognition engines are compared in this work: a segmentation-free approach based on a convolutional recurrent neural network (CRNN). The recognition is carried out over the entire LP image without any prior segmentation, and a joint detection/recognition approach performs the recognition on the plate component level. We also introduced a new large-scale dataset for automatic LP recognition that includes 9.175 fully annotated images. In order to reduce the time and cost of annotation processing, we propose a new semi-automatic annotation procedure of LP images with labeled components bounding boxes. The produced results are presented, discussed, and compared with the state of the artwork.

Chapter 4 presents our Vehicle Make Model Recognition (VMMR) with a method for more structured feature extraction by leveraging robust multi-stream deep networks architecture. We employ a novel dynamic combination technique to aggregate different vehicle part features with the entire image. This allows combining global representation with local features.

Chapter 5 concerns the V-Reid problem; we start with a review of object re-identification in general and the evaluation metrics used. Then we review works on both person and vehicle re-identification. Finally, we review the public datasets relevant to the tasks studied in this thesis.

For V-Reid, massive data annotation is an expensive task and may be impossible with real-time learning and identification, in which semi or unsupervised learning is needed using learning paradigms. We focus our interest on semi-supervised V-Reid, where each identity has a single labeled and multiple unlabeled samples in training. We propose a framework that gradually labels vehicle images taken from surveillance cameras. Our framework is based on deep CNNs, which are progressively learned using a proposed feature anchoring regularization technique. A second take on the V-Reid problem involves a domain adaptation approach in which we propose a cross camera V-Reid, a system that performs re-identification from different cameras. Usually, vehicle images captured from different cameras exhibit large intra-class variability and do not follow the same distribution. To alleviate this problem, our system models the cross cameras re-identification as a domain adaptation problem. It uses optimal transport to transfer knowledge from different cameras and project vehicle features onto a shared space in which the same vehicle identities are close. The similarities are computed using the Euclidean distance in the common space.

The final Chapter 6 sums up the conclusions.

# Chapter 2

## Research Preliminary

### 2.1 Visual Feature Representation

Earlier work focused on low-level features representation: Scale Invariant Feature Transform (SIFT [84, 82, 83]) for an instant, is used to detect and describe local features in images. Key points of objects are first extracted and stored. Hence, an object is recognized by comparing each new feature with the ones stored. However, this method is computationally expensive. To overcome this issue, Speeded Up Robust Features (SURF [6]) and Histogram Oriented Gradients (HOG) have been proposed for more robustness and speed. Another approach is the SURF method which uses the Hessian matrix approximation to detect key points, which gives more robust results while being faster than the SIFT-based methods. Several variations of the SURF descriptor have also been used. These variations include (1) Features from Accelerated Segment Test (FAST), which is a key-points detection method designed for real-time applications, (2) Binary Robust Independent Elementary Features (BRIEF), and (3) Oriented FAST, which uses FAST detector for key-points detection and BRIEF as a descriptor. In this work, however, we focus on deep visual features extraction (learning). The next sections of this chapter are a preliminary to the theories and tools used in this work.

### 2.2 Deep Visual Feature Learning

#### 2.2.1 Convolutional Neural Network

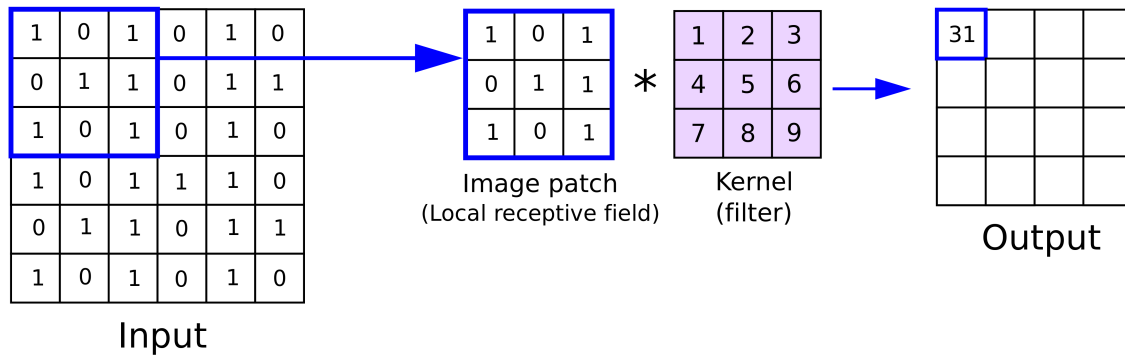
CNNs [65] are a fundamental tool for machine learning researchers, especially for computer vision. A CNN consists of one or more convolutional layers with a sub-sampling step followed by a classifier, for instance, a fully connected layer. Convolutional layers are easier to train and have much fewer parameters than fully connected layers with the same number of hidden units. They are designed to take advantage of the 2D struc-



ture of an image which is achieved with local connections and tied weights followed by pooling resulting in translation-invariant features. In the following sub-sections, we introduce the main operations in a general CNN model.

### Convolution

Convolutional filters receive the input image; they are called convolutional because of the process involving the filters where the present input image is convolved with what it has learned in the past, i.e., the convolved features created by the filters. Figure 2.1 is a schematic diagram of the convolution process.



**Figure 2.1:** Filters are convolved over the image or a feature map in a sliding window fashion.

Each convolutional filter represents a feature; this means that convolution has the property of being translational invariant. The CNN can then comprise the resulting prediction from those extracted features. Practically, the prediction does not depend on where the features are located but only if they are present so an object can change position in an image and still be recognized or detected. Convolution filters are controlled by multiple parameters such as stride, zero-padding, and channel depth. A stride is the number of pixels with which the filter slides. Having a more significant stride produces smaller feature maps and vice versa. Zero paddings allow us to control the size of the feature maps to a certain extent. It pads the input matrix with zeros around the border. The channel depth is the number of filters used in convolution. The more filters, the more image features get extracted.

### Activation Layer

Like neurons are fired in the brain, the activation layer controls how the signal flows from one layer to another. The most common of the activation function is the Rectified Linear Unit (ReLU) given by:

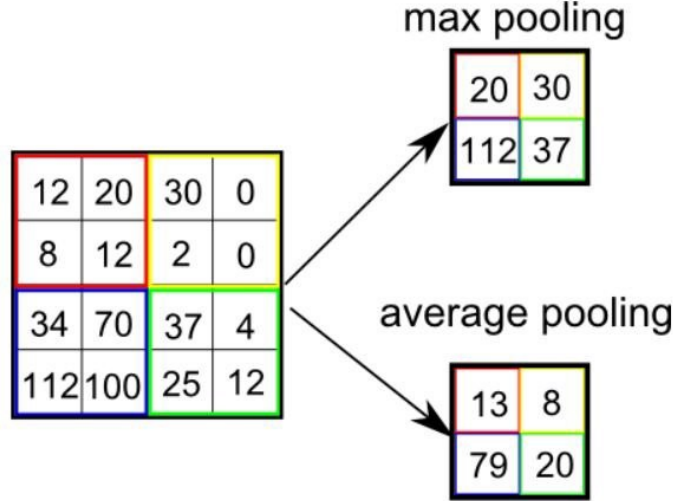
$$f(x) = \max(0, x) \quad (2.1)$$

Other variations of the ReLU activation function were proposed, like the random ReLU

(RReLU), LeakyReLU, and PReLU.

### Pooling

Filters can be sensitive to noise hence, a smoothing technique is employed called pooling or sub-sampling. It can be achieved by taking averages or maximum over a sample of the image or feature map. Figure 2.2 shows an example of the two pooling methods.



**Figure 2.2:** Average pooling and Max pooling methods.

### Fully-Connected Layer

The fully-Connected layer is the final layer in the network. The fully connected term is since the neurons of preceding layers are connected to every neuron in subsequent layers. The fully-Connected layer acts as a classifier; it uses the high-level features from convolutional and pooling layers to classify the input image into classes.

### Back Propagation

Back-Propagation algorithm [107] is used to calculate the gradients of the error with respect to all weights in the network and use gradient descent to update all filter weights and parameter values to minimize the output error. The forward propagation is defined as that given the activated  $i$ -th layer  $a^i$ , the  $(i+1)$ -th layer's activation is computed as:

$$\begin{aligned} z^{i+1} &= W^i a^i + b^i \\ a^{i+1} &= f(z^{i+1}) \end{aligned} \quad (2.2)$$

$W$  and  $b$  and respectively the weights and bias in the network. The error term for the  $(i+1)$ -th layer is defined as  $E^{i+1}$ . The loss function is set as  $L(W, b; x, y)$  where  $(x, y)$  are the training data and label pairs. The error of the  $i$ -th layer is computed as follows:

$$E^i = ((W^i)^T E^{i+1}) \cdot f'(z^i) \quad (2.3)$$

Where ”.” denotes the element-wise product operator. The gradients are:

$$\begin{aligned}\nabla_W^i L(W, b; x, y) &= E^{i+1} (a^i)^T \\ \nabla_{b^i} L(W, b; x, y) &= E^{i+1}\end{aligned}\tag{2.4}$$

In the case the i-th layer is a convolutional and pooling layer then the error is propagated as:

$$E_k^i = ((W_k^i)^T E_k^{i+1}) \cdot f'(z_k^i)\tag{2.5}$$

With k indexes the filter number and  $f'(z_k^i)$  is the derivative of the activation function. The upsample operation has to propagate the error through the pooling layer by calculating the error with respect to each unit incoming to the pooling layer. Finally, the gradient is calculated with respect to the filter maps:

$$\begin{aligned}\nabla_{W_k^i} L(W, b; x, y) &= \sum_{j=1}^m (a_j^i) * \text{rot90}(E_k^{i+1}, 2) \\ \nabla_{b_k^i} L(W, b; x, y) &= \sum_{c,d} (E_k^{i+1})_{c,d}\end{aligned}\tag{2.6}$$

Where  $a^i$  denotes the i-th layer's input.  $(a_j^i) * E_k^{(i+1)}$  is the convolution between the j-th input in the i-th layer and the error with respect to the k-th kernel filter. m is the number of input channels.  $\text{rot90}(A, 2)$  is the function to rotate input A by  $90 \times 2$  degrees.  $c$  and  $d$  are the width and height of the kernel filter, respectively.

### Loss Function

CNN optimization is driven by a loss function, which indicates how close to representation is to the goal using scalar value specifying the ”badness” of the weights. Thus the goal of training is to find the weights that minimize the loss functions. The following are the most common losses.

- Euclidean Loss also known as  $l_2$  loss defines as:

$$E = \frac{1}{2N} \sum_{n=1}^N \|\tilde{y}^n - y^n\|_2^2\tag{2.7}$$

- Softmax Loss computes the multinomial logistic loss for a one if many classification task, passing real-valued predictions through a softmax to get a probability distribution over classes.
- Mean Square Loss used most commonly in regression problems defines as:

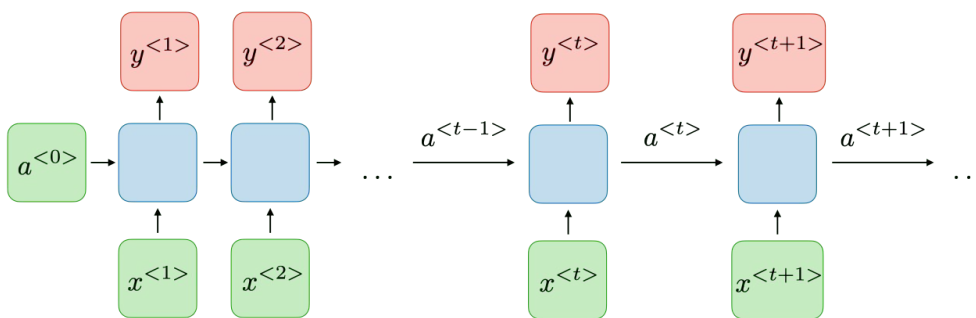
$$E = \frac{1}{N} \sum_{n=1}^N (y^n - \tilde{y}^n)^2\tag{2.8}$$

## Dropout

Dropout is an essential concept in training a deep model. Dropout refers to ignoring neurons during the training phase of some random neurons. Dropout aims to prevent the network from overfitting. When the performance of the model on the training set is very high but drops on the test set, then the model is over-fitting. "Ignoring" means these units are not considered during forwarding or backward pass.

## 2.2.2 Recurrent Neural Network

The rise in popularity of RNNs can be attributed to its ability to model sequential data as shown in Figure 2.3.



**Figure 2.3:** *Recurrent neural network architecture.*

RNNs have been successfully applied to multiple problems: image captioning [128], language modeling [39] and action recognition [75, 76]. In the following, we review RNN's important concepts and methods.

### Vanilla RNN

For sequence type data, both the current input and previous states are necessary to infer a correct prediction. For the current state  $t$  for example, both the input  $X_t$  and the previous state  $h_{t-1}$  are used to compute  $h_t$ :

$$h_t = f(X_t, h_{t-1}) \quad (2.9)$$

However, this model falls short for long-term dependencies.

### Long Short-Term Memory

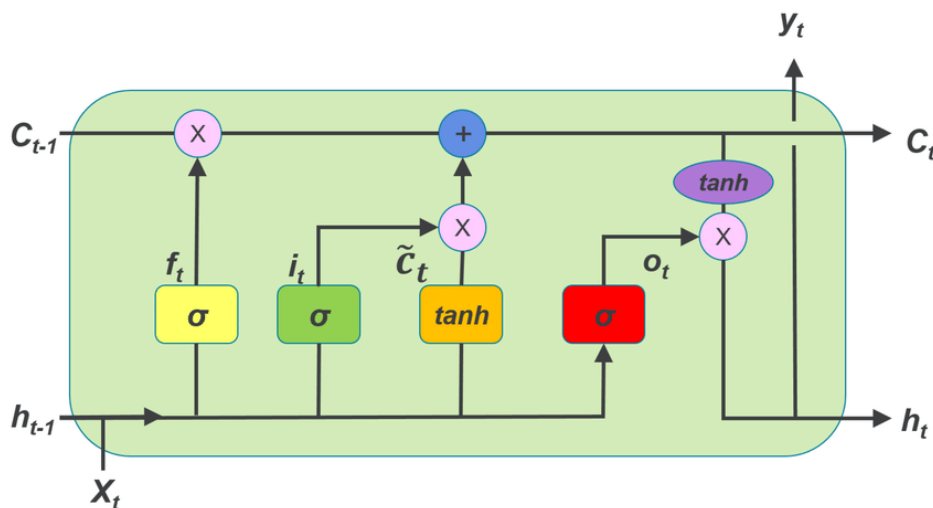
Long Short-Term Memory networks (LSTMs) [49] aim to handle learning for long-term dependencies. LSTM splits the  $h_t$  into 2 variables  $h_t$  and  $C$ . It has three gates to control what information will pass through:

$$\begin{aligned} gate_{forget} &= \delta(W_{fx}X_t + W_{fh}h_{t-1} + b_f) \\ gate_{input} &= \delta(W_{ix}X_t + W_{ih}h_{t-1} + b_i) \\ gate_{output} &= \delta(W_{ox}X_t + W_{oh}h_{t-1} + b_o) \end{aligned} \quad (2.10)$$

$gate_{input}$  controls what part of the new information will be added to state  $C$ .  $gate_{output}$  controls the part of the cell state that will be exposed as the hidden state and  $gate_{forget}$  as the name implies controls what part of the previous cell state will be removed. With  $\delta(x) = (1 + e^{-x})^{-1}$ . Hidden and Cell state will be updated as:

$$\begin{aligned}\tilde{C} &= \tanh(W_{cx}X_t + W_{ch}h_{t-1} + b_c) \\ C_t &= gate_{forget} \cdot C_{t-1} + gate_{input} \cdot \tilde{C} \\ h_t &= gate_{output} \cdot \tanh(C_t)\end{aligned}\tag{2.11}$$

$C_t$  is formed by forgetting part of the previous cell state while adding new proposal  $\tilde{C}$ . Figure 2.4 shows the LSTM architecture.



**Figure 2.4:** Long Short-term Memory architecture.

Previous models examined have augmented the underlying structure of a simple RNN to improve its performance in learning the contextual dependencies of single dimension sequences. However, there exist several problems, which require an understanding of contextual dependencies over multiple dimensions. The most popular network architectures use CNNs to tackle these problems.

The incorporation of recurrent connections into each convolutional layer can shape a recurrent convolutional neural network. (RCNN) [36]. The activation of units in RCNN evolves over time, as they are dependent on the neighboring unit. This approach can integrate the context information, important for object recognition tasks. This approach increases the depth of the model, while the number of parameters is constant by weight sharing between layers. Using recurrent connections from the output into the input of the hidden layer allows the network to model label dependencies and smooth its own outputs based on its previous outputs.

## 2.3 Learning Paradigms

### 2.3.1 Machine Learning Paradigms

Supervised learning is a class of machine learning algorithms that learn from labeled data. However, giving a label to each sample of the training set can be an expensive and time-consuming job because of the labor necessary to annotate the data. Hence, semi-supervised and Unsupervised learning gained interest in recent years. Semi-supervised learning is a class of machine learning algorithms that learn from both labeled and unlabeled data. Unsupervised learning, on the other hand, is when all of the training data is unlabeled. Furthermore, recent extensions of semi-supervised learning were researched: (1) One-shot learning where only a single labeled sample of each class is available. (2) Few-shots learning when few samples are labeled per class.

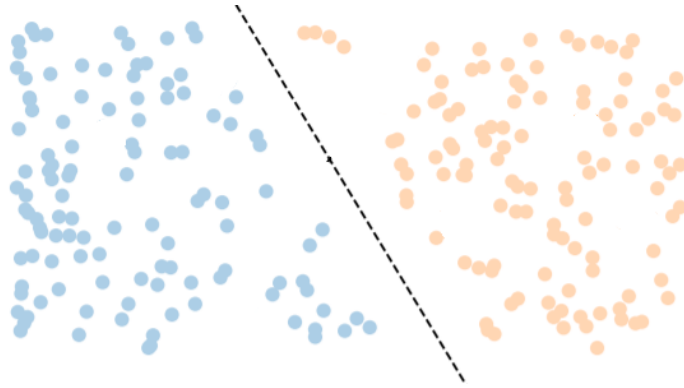
A second take on the labeled versus unlabeled problem is Transfer Learning. The main concept of transfer learning is the transfer of knowledge from a previously trained model to a new model that performs a different task. For example, models that are trained on the ImageNet [65] dataset can be used as the basis the new a new model to classify objects that are not in the ImageNet dataset. The process is called fine-tuning. Domain adaptation is a special case of transfer learning that is specific to transfer knowledge between representations. In other words, domain adaptation is when the labels of the two domains are the same, but the distribution is different.

#### Semi-Supervised Learning

Extracting features from unlabeled data to improve learner performance is not always possible; some assumptions must be established first. For instant, the marginal data distribution  $p(x)$  over the input space contains information about the posterior distribution  $p(y|x)$  where  $x$  is a point in the input space and  $y$  is it label. Otherwise, it is impossible to improve performance using the unlabeled data [147]. Though, in most deep learning problems, this condition is satisfied. The next assumptions describe how  $p(x)$  and  $p(y|x)$  interact and how to decide whether the unlabeled sample can be used to improve performance or not.

#### **The smoothness and low-density assumptions:**

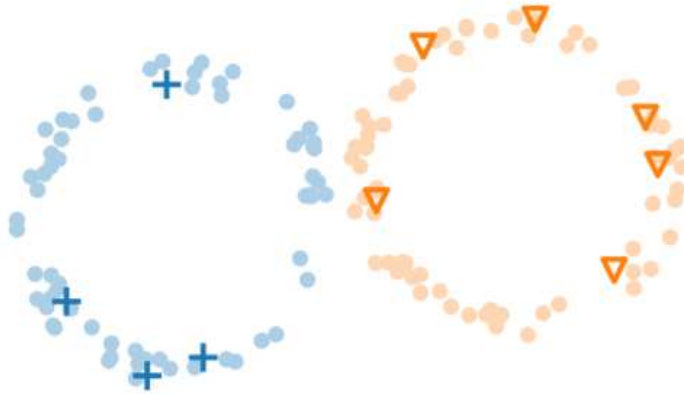
If two samples  $x_1$  and  $x_2$  are close in the input space, their labels  $y_1$  and  $y_2$  are also close. This assumption can be applied transitively to unlabelled data. For if  $x_1$  is close to  $x_2$  and  $x_2$  is close to  $x_3$  then due to the smoothness assumption, we can expect that the label of  $x_1$  is close to the label of  $x_3$ . On the other hand, the low-density assumption dictates that the decision boundary should not pass through high-density areas in the input space. This assumption is defined over the true input data distribution  $p(x)$ , and it means that the boundaries of a classifier should be in an area where few data samples are observed. Figure 2.5 shows an example that respects both assumptions.



**Figure 2.5:** *Distribution obeying both smoothness and low-density assumptions.*

**The manifold assumption:**

In serious learning problems, data points are observed in high-dimensional input space  $R^d$  and concentrated on lower-dimensional substructures, also known as manifolds (topological spaces locally Euclidean). This assumption states that the input space comprises multiple lower-dimensional manifolds on which all data points lie, and points on the same manifolds have the same label.



**Figure 2.6:** *Distribution obeying the manifold assumption.*

High-dimensional data, like images, Euclidean feature distance is rarely a good indicator of the similarity between data points. Hence, most semi-supervised learning approaches for images rely on a weak variant of the smoothness assumption that requires predictions to be invariant to minor perturbations in the input.

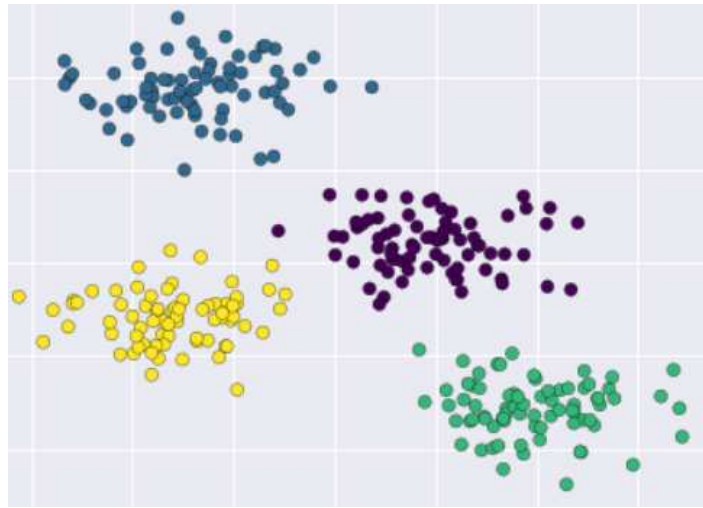
## Unsupervised Learning

Unsupervised learning is the most natural type of learning; grouping or clustering objects according to similarities and same characteristics is a fundamental skill for learning. Unsupervised learning can be classified into two main categories.

(1) Parametric Unsupervised Learning, where it is assumed that sample data comes from a population that follows a probability distribution based on a fixed set of parameters. In such distribution, all members are parameterized by mean and standard deviation. Parametric Unsupervised Learning involves the construction of Gaussian Mixture Models and using Expectation-Maximization algorithm for prediction.

(2) Non-parametric models, on the other hand, do not make any assumptions about the distribution; rather, data is grouped into clusters [108, 59] where each cluster represent a category of the data. The goal from having unlabeled samples can be clustering [108, 59], good representation [15, 34] (to determine how the data is distributed in the space) or good representation at first then clustering [31].

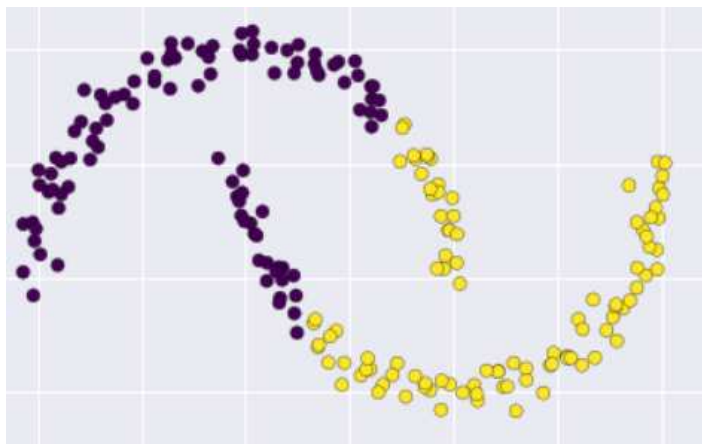
Clustering can be divided into multiple categories: **Exclusive Clustering** is when data grouped elusively, meaning a sample cannot be in two or more different clusters. Figure 2.7 is an example of exclusive clustering.



**Figure 2.7:** *Example of Exclusive Clustering.*

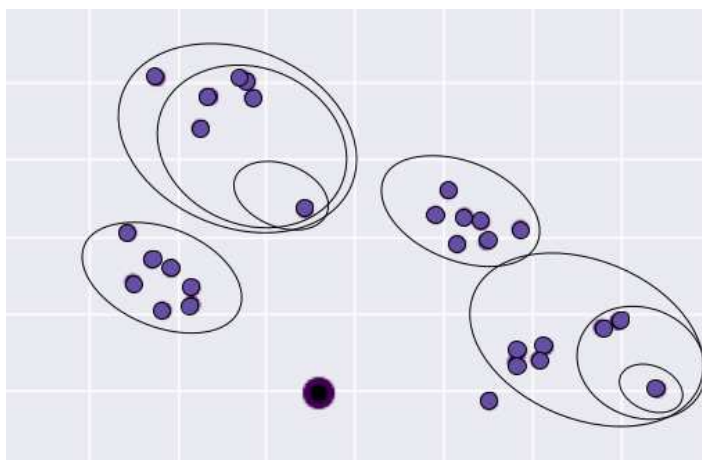
K-means is a standard algorithm for exclusive clustering; given a prior-fixed number of clusters ( $k$ ), the algorithm defines  $k$  centers for each cluster. In contrast, **Overlapping Clustering**, is when a sample belong to two or more clusters with different degrees of membership. Figure 2.8 is an example of overlapping clustering.





**Figure 2.8:** *Example of Overlapping Clustering.*

**Hierarchical Clustering** starts by setting every sample as a cluster then based on the union between the two nearest clusters after few iterations it reaches the final clusters. Figure 2.9 is an example of hierarchical clustering.



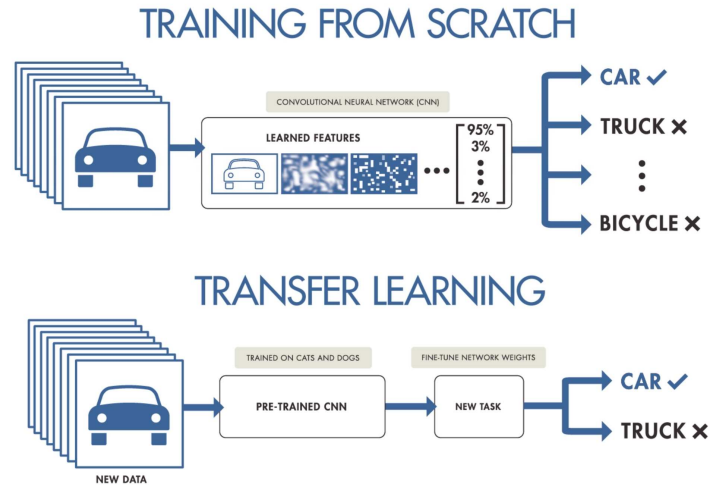
**Figure 2.9:** *Example of Hierarchical Clustering.*

Finally, **Probabilistic Clustering** is where each sample has a probability of belonging to a cluster.

### Transfer Learning

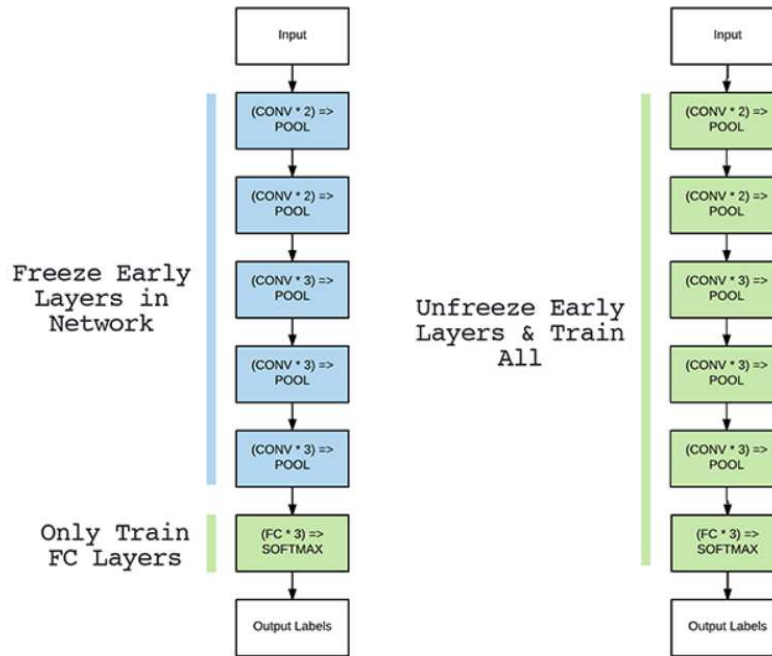
Transfer learning's primary goal is to reduce the amount of data needed to train a model. A very common method for transfer learning is **fine-tuning** show in Figure 2.10. With the huge success of AlexNet on the ImageNet[65] competition, deep

learning acquired much attention from both researchers and the public. However, training a model from scratch requires resources and time. Fine-tuning solved this problem by initializing the model using weights from a source model training on ImageNet; the new model or target model can be trained on a new dataset with lesser time and resources. This process is called Fine-tuning a model.



**Figure 2.10:** *Transfer learning using fine-tuning.*

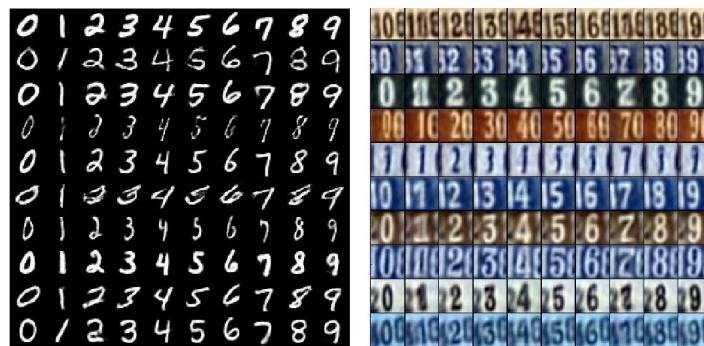
First, the weights from the source model are uploaded to the target model except for the last layer. Then freeze all or some of the convolutional layers (freezing is not updating the weights of those layers). Finally, the target model is trained on the new dataset using an optimizer and a dropout layer. The second way to do fine-tuning is not to freeze any of the layers and train the whole model. Figure 2.11 shows both approaches.



**Figure 2.11:** *The two methods for fine-tuning a deep convolutional network.*

### Domain adaptation

Contrary to fine-tuning, domain adaptation is applied when the labels of the source and target task are the same but their respective data samples are different as shown in Figure 2.12. Hence, domain adaptation is the transfer of knowledge between representations.



**Figure 2.12:** *The MNIST dataset (left) and SVHN dataset (right) share the same labels (0, 1, 2, 3, 4, 5, 6, 7, 8, 9) but the samples are different.*

### 2.3.2 Progressive & Curriculum Learning

Curriculum learning describes a type of learning where training samples are aligned from easy to complex; progressive learning is when the alignment of training sample from easy to hard depends on the "trained on" samples, meaning that the model trains on some of the samples then chooses the easiest of what remains to train on. Progressive & Curriculum learning has both an effect on the speed of convergence of the training process and the model's performance. The basic idea is to start small, learn more accessible aspects of the task, and gradually increase the difficulty level. For machine learning its important to define "start small" and "easier aspects" to have an effective progressive & curriculum Learning.

In deep learning, or more precisely, deep feature extraction, starting small and easier aspects can be defined as the relationship between features and geometrical properties. Distance Metrics is a widespread tool to compare features in a geometrical space:

- Hamming Distance calculates the distance between two binary vectors. The hamming distance between two binary vectors  $V_1$  and  $V_2$  is calculated as follows:

$$HammingDistance = \sum_i^N | V_1[i] - V_2[i] | \quad (2.12)$$

Hamming distance is the number of bit positions in which the two bits are different.

- Euclidean distance represents the shortest distance between two real-valued points in vector space.

$$EuclideanDistance = \sqrt{\sum_i^N (V_1[i] - V_2[i])^2} \quad (2.13)$$

It is common to normalize values in vectors otherwise the distance measure will be dominated by large values.

- Manhattan Distance is calculated as the sum of the absolute differences between the two vectors  $V_1$  and  $V_2$  as shown in the following equation:

$$ManhattanDistance = \sum_i^N | V_1[i] - V_2[i] | \quad (2.14)$$

Manhattan distance is favored over euclidean distance when the values are integers.

- Minkowski Distance is a generalization of the Euclidean and Manhattan distance measures by adding a parameter  $p$ , called *order*, Hence, for two vectors  $V_1$  and

$V_2$  with real values, the distance is calculated as follows:

$$MinkowskiDistance = \sum_i^N (|V_1[i]-V_2[i]|^p)^{1/p} \quad (2.15)$$

for  $p = 1$  the distance is equal to the Manhattan distance and for  $p = 2$  the distance is equal to the Euclidean distance.

# Chapter 3

## LP Detection and Recognition

### 3.1 Literature Survey

#### 3.1.1 Object Detection

Object detection is the identification of an object in the image, along with its localization and classification. Viola-Jones Object Detector is the first object detector; it came out in 2001[122]. It was used as facial detection and was wildly used at the time. However, with the recent success of deep learning approaches, new methods were proposed with far more robustness and accuracy. The first Deep Learning object detector model was called the Over feat Network[111] which used CNN's with a sliding window approach.

Since 2012, object detector models have gone through many changes; the first breakthrough in object detection was the RCNN [36]: it is made of three main parts, the region extractor, the feature extractor, and finally, the classifier. A region proposal algorithm extracts the Region Of Interest (ROI), then each region is fed to a classifier, and finally, the extracted features are classified.

Soon after came the Fast RCNN[35] with substantial improvement. The RCNN Model takes every region proposal and runs them through the convolutional base. This is quite inefficient. The Fast RCNN aims to reduce this overhead by running the convolutional base just once. The Faster RCNN [106] came out after the Fast RCNN paper. It proposed a detector that learns in an end-to-end fashion.

#### Evaluation Metrics

Object detection metrics help assess how the model performs on an object detection task. Competitions such as PASCAL VOC[26] and MSCOCO[73] provide predefined metrics to evaluate object detectors on their datasets. The object detection task localizes the object with a bounding box associated with its confidence score. Therefore to determine how many false positives (FP), false negatives (FN), and True positives (TP) were generated, we use the Intersection over Union (IoU) metric. The IoU score

ranges from 0 to 1.

Intersection over Union, also referred to as the Jaccard Index, quantifies the similarity between the ground truth bounding box and the predicted bounding box:

$$IoU = \frac{AreaofOverlap}{AreaofUnion} \quad (3.1)$$

A second metric is Precision: the probability of the predicted bounding boxes matching actual ground truth boxes.

$$Precision = \frac{TP}{TP + FP} \quad (3.2)$$

Recall is the true positive rate measures the probability of ground truth objects being correctly detected.

$$Recall = \frac{TP}{TP + FN} \quad (3.3)$$

Average precision (AP) is a single number metric that encapsulates both precision and recall and summarizes the Precision-Recall curve by averaging precision across recall values from 0 to 1.

The Mean Average Precision (mAP) is the averages AP over the N classes.

## Vehicle Detection

For detecting vehicles, a variety of appearance features have been used. Features like edge, shadow, and symmetry are used for detecting vehicles. However, in recent years, general and robust feature sets are used for detecting vehicles rather than from simpler image features like edges and symmetry. These feature sets allow for direct classification and detection of objects in images, now common in the computer vision literature. Features extractors like HOG and Haar-like are extremely well represented in the vehicle detection literature, as they are in the object detection literature. To extract the HOG features, edges are evaluated over the image and then discretizing and ditching the orientations of the edge intensities into a histogram[22]. HOG features are expressive image features, showing good detection presentation in a range of computer vision tasks. In [18], for vehicle detection, the symmetry of the HOG features extracted in a given image patch, along with the HOG features themselves. In some cases, vehicle pose is also determined by HOG features. The main drawback of HOG features is that they are pretty slow to compute.

## LP detection

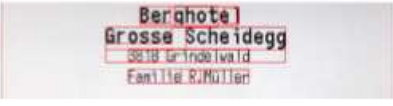

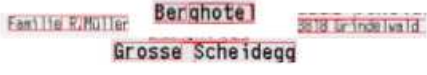

Since localization of LPs can be considered as a detection problem, different regional CNN methods that were recently developed for fast and precise object detection can be applied, such as Faster-RCNN [106], YOLO [103], YOLOv2 [104] and SSD [77]. The detection step consists of localizing the bounding box containing the LP from the

whole raw input image. The output of this step dramatically influences the accuracy of the recognition step. Plenty of LP detection algorithms have been proposed. Most of them used traditional machine learning techniques with hand-crafted features based on specific descriptors such as edge, color, and texture descriptor [24, 141, 3]. The LP detection can be considered as a detection of a homogeneous text zone by detecting the characters directly from the image [93]. Although fast and straightforward, this kind of approach leads to a low detection accuracy because the features learned from the characters may not be enough to confidently find all plate characters in the image. In addition, other characters may exist in the image, which can be confusing for the LP detection. Other approaches consider the LP as a region with rich contrast and high edge density [124], or as a region containing a high density of key points detected with SIFT descriptor [92]. Recently, deep learning-based approaches have been proposed for localizing LPs. In [69], the authors use a 4-layer CNN to detect the presence of text zones in the input image. Then a second 4-layer plate/non-plate CNN classifier is used to distinguish LPs from general text. [118] uses a classifier based on the FAST-YOLO network architecture to detect the frontal view of the car from the input image and then extracts the LP from the detected frontal view image. In [90], the authors propose an LP detection method based on Faster-RCNN network which assures both high accuracy and low time cost. [86] introduces a pipeline architecture based on a sequence of deep CNNs for LP detection under different conditions (variations in pose, lighting, occlusion, etc.) and working across a variety of LP templates (sizes, backgrounds, fonts, etc.)

### 3.1.2 Optical Character Recognition

Text is one of the most important inventions in human history; it is used in almost all aspects of human life. Hence, vision-based text recognition has attracted much interest for many years. [16] is a survey on most important approaches, old and new, for both recognition of text from documents as well as from different scenes. In the following, we discuss the recent developments in both text and scene text recognition accordingly. Although both problems are very different, as shown in Figure 3.1 they are similar from a semantic perspective.



Document Text Recognition	Text Scene Recognition
	
	
<ul style="list-style-type: none"> <li>- Single Color Background</li> <li>- Single color, regular font, consistent size, and uniform arrangement.</li> <li>- Clear and frontal.</li> </ul>	<ul style="list-style-type: none"> <li>- Complex background</li> <li>- Multiple colors, irregular fonts, different sizes, and diverse orientations</li> <li>- Distorted by nonuniform illumination, low resolution, and motion blurring.</li> </ul>

**Figure 3.1:** Differences between text and scene text recognition.

## Text Recognition

Text recognition problems can be viewed as multiple sub-tasks: (1) Text Localization [72] aims to group text components into candidate text regions with as little background as possible. Early text localization methods are based on low-level features, such as color [68, 132], gradient [87, 115], stroke width transform [25, 89], maximally stable extremal regions (MSER) [91, 114] and canny detector [12, 19] to mention a few. However, recent works are based on deep learning methods [46, 137, 133] (2) Text Verification [66] as the name implies verifies the text candidate regions as text or non-text to filter the candidate regions produced by text localization since sometimes it introduces false positives. Methods for text verification used prior knowledge [64, 117, 87], support vector machine (SVM) classifier [131] and conditional random fields (CRFs) [72] (3) Text detection [130, 70] determines whether text is present using localization and verification procedures. Approaches can be divided to regression based methods [80, 135, 138] and instance segmentation-based methods [126, 43] (4) Text segmentation (5) includes text line segmentation [101, 64] and character segmentation [94, 116] Text recognition [62] translates a cropped text into a target string sequence. Most recent studies have used deep learning encoder decoder frameworks [17, 85] (6) End-to-end system [62], Given a scene text image, an end-to-end system can directly convert all text regions into the target string sequences. It includes text detection and recognition as two sub-problems. A second approach is to jointly optimize text detection and text recognition by sharing information [54, 47]. Scene Text Recognition (STR) acquired

interest from both academia and industries alike for both its research and practical impact. The following section discusses an application of scene text recognition.

### LP Recognition

In the literature, we distinguish two kinds of approaches for LP recognition: the segmentation-based and the segmentation-free approaches. [24] presents a complete review of traditional approaches for LP recognition. We focus more in this part on works using deep learning techniques.

Segmentation-based approaches extract each character in the LP firstly. Then an OCR algorithm is performed to recognize each of them. Existing works on LP segmentation can be divided into two main categories: projection-based and connected component-based. Projection-based approaches exploit the fact that characters and background have obviously different colors in an LP, giving opposite values in the binary image. Histograms of vertical and horizontal pixel projections can then be exploited for character segmentation [100, 41]. Such approaches can be easily affected by the rotation of the LP. Connected Component-based approaches [13, 30, 33, 61] perform segmentation by labeling all connected pixels in the binary image into components. This type of method is robust to rotation, but it fails to segment characters correctly when they are joined together or broken. Since the characters are segmented, a recognition step can be performed as a classification task with one class per alphanumeric character. Existing algorithms can be divided into two categories: template matching and learning-based methods.

Template matching-based methods [57, 63] consists of comparing the similarity of a given character and templates. The most similar template is then chosen. Several similarity measures have been proposed, including Mahalanobis distance and Hamming distance [24]. These methods have usually been applied to binary images and are limited since they work for single character size and font, and they do not support rotation or broken characters.

Learning-based methods are more robust and can deal with characters of different font, illumination, or rotation. They use machine learning techniques to discriminate characters using one or multiple features such as edge density, gradient, scale-invariant transform (SIFT), etc. [102] uses a 5-layer CNN to recognize Malaysian LPs where individual characters are manually extracted and segmented. The recognition is then performed as a classification task with 33 classes and achieves 98.79% accuracy on a reduced number of samples.

[98] proposes an approach using a CNN classifier for the recognition of LP characters and firstly uses some pre-processing techniques on input images, such as filtering, thresholding, and then segmentation.

For segmentation-free methods, the recognition is performed on the global LP image without character segmentation. Usually, a sliding window over the input image generates many tentative characters in small steps. Each tentative character is then used by a recognizer. When the sliding window totally swipes the input image, the predicted

outputs are analyzed, and the final sequence is decided. Consecutive same characters are considered a single character, and character space is used to separate others. In the context of LP Recognition, few works have proposed segmentation-free approaches based on DL techniques. In [14], the authors propose a CNN to perform feature extraction on the LP and a recurrent neural network to learn the sequential order of character features. [69] proposes to recognize the license characters as a sequence labeling problem using a recurrent neural network with long short-term memory (LSTM), which is trained to recognize the sequential features extracted from the whole LP via CNNs. [56] uses a deep (16-layers) CNN based on Spatial Transformer Networks [55], to perform a less sensitive character recognition to spatial transformations on whole LP image and avoiding the challenging task of image segmentation into characters. In [118], the authors use a YOLO-based network to detect and recognize LP characters using a joint classification-detection engine. In [10], the authors use a CNN architecture for identifying characters within an image of LP and localizing the character bounding box corners. This step deal with a 33 classes classification task for Italian LPs.

In general, we note that the use of deep learning techniques to LP detection and recognition is still limited and restrictive to specific conditions (specific nations, unilingual plates, special LP formats...). Only a few works perform ALPR in an end-to-end fashion, and most of the proposed methods rely on hand-crafted features.

## 3.2 Datasets and Evaluation Protocols

### 3.2.1 GAP-LP Dataset

The GAP-LP dataset images were acquired with different quality cameras under different resolutions, view angles, and in daylight to better test the robustness of LP detection/recognition algorithms. GAP-LP dataset <sup>1</sup> is freely available to the research community. It is composed of 9175 fully annotated images for both LP detection and recognition. The dataset is split as follows: 7117 images for training, 456 for validation, and 1602 images for the test. Figure 3.2 shows some image samples from the GAP-LP dataset.

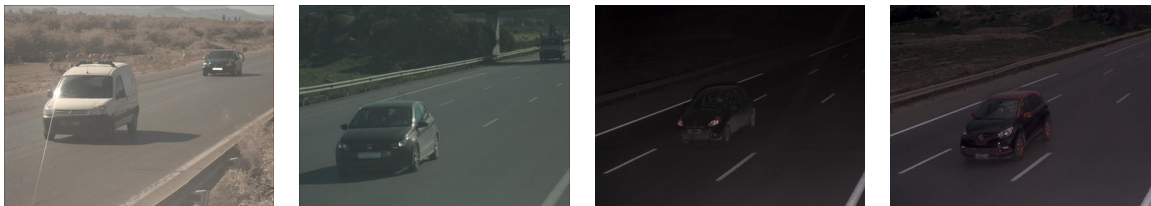


**Figure 3.2:** *GAP-LP sample images and their respective plates*

<sup>1</sup><https://sites.google.com/site/matdbparking>

### 3.2.2 RADAR Dataset

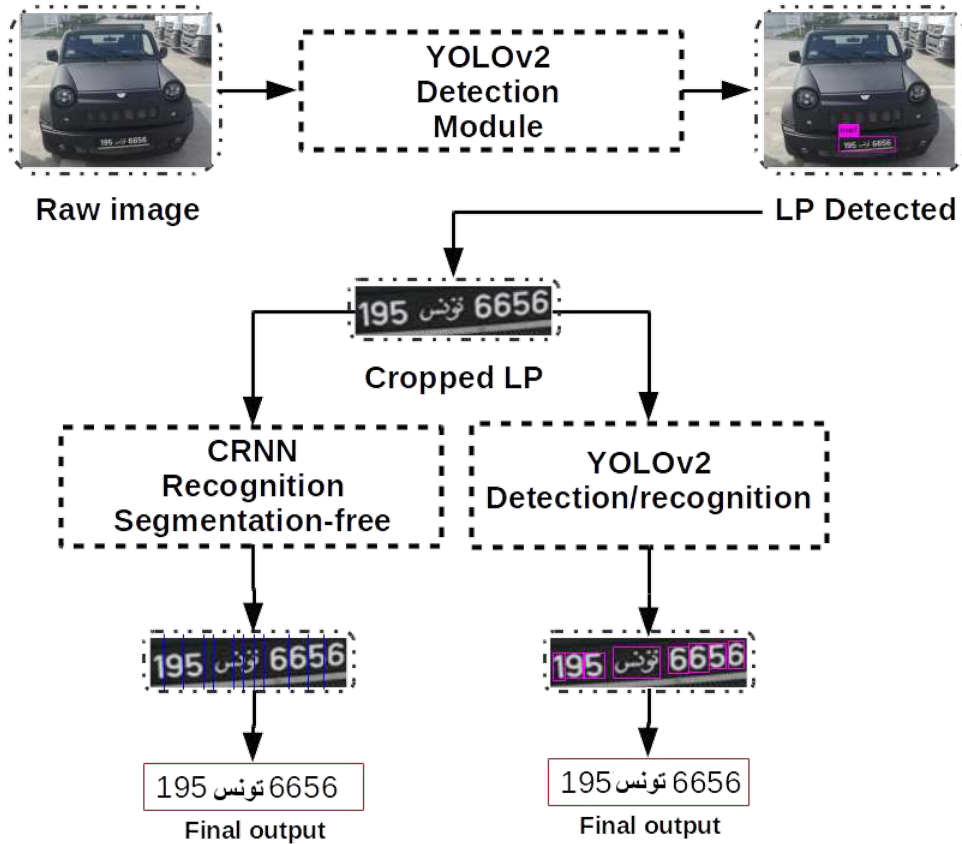
The Radar dataset is composed of 6448 JPEG color images taken from radars on different Tunisian roads. The images were taken from natural scenes with complex backgrounds and under various lighting conditions (day, night, sunny, rainy, ...), different angles and positions, and several LP shapes (square and rectangle) and norms. The dataset is divided into three sets: 3998 images for training, 2000 for the test, and 450 for validation. Figure 3.3 shows some image samples from the Radar dataset.



**Figure 3.3:** *Radar image samples, LPs blurred for privacy*

## 3.3 Material and Methodology

We propose an end-to-end approach for both detecting and then recognizing the vehicle LP in a cascade fashion (LP detection - LP recognition). The proposed system proceeds on two stages: plate detection stage and plate recognition stage (as shown in Fig. 3.4). The first module is fed the full raw image as input; YOLOv2 detects plates in the image and then outputs the cropped plate image to the second module. YOLOv2 bases its prediction partly on contextual information, i.e., the surrounding of the object. Since the LP is placed in the same position in every vehicle, YOLOv2 is an effective LP detector. The recognition modules take as input the cropped plate image. Two LP recognition engines are compared in this work. The segmentation-free approach integrates the advantages of both CNN for feature extraction and RNN for LP sequence modeling and transcription and does not need any prior segmentation in the training step. Only the text on the LP is provided. The second approach is based on a joint detection/recognition approach using YOLOv2 and requires sophisticated manual labeling of the LP components with bounding boxes. Nevertheless, the robustness of an object detector is very effective in recognizing multi-norm LPs.



**Figure 3.4:** Illustration of our LP detection/recognition system based on YOLOv2 and CNN. The output of the YOLOv2 detection module becomes the input of both the CRNN recognition module and the YOLOv2 detection/recognition module.

LP detection is the first stage of the proposed pipeline. The quality of the cropped plate image can influence the whole system's performance. If the cropped image of the LP doesn't contain the whole text, the recognition cannot predict the correct label. The detection system must be robust to environment variations and to Tunisian LP specifications, where the LP formats vary significantly, as described in section 3.2.1. This requires a combination of a solid visual feature extractor and a mechanism to find the LP boundaries efficiently. To perform LP detection and achieve a good compromise between accuracy rates and running times, we use an overall state-of-the-art real-time object detection system in its second version dubbed YOLOv2 [104]. YOLOv2 considers object detection as a single regression problem, straight from image pixels to bounding box coordinates and class probabilities. It uses a multi-scale training method and offers a trade-off between speed and accuracy. To detect LPs, YOLO uniformly divides the image into cells in a grid. During training, the deep neural network takes

images from the training dataset as input, forms a loss function that incorporates the cross-entropy loss, the  $L2$  regression loss, and the randomness, and back-propagates the gradients to update the parameters. Thus, during testing, the network gives as output the LP bounding box.

We adapt the original YOLO Architecture to perform LP detection. YOLO predicts  $B$  bounding boxes ( $B = 5$ ) each with four coordinates and confidence, so the number of filters is given by:  $N_{filters} = (N_{classes} + B) \times 5$ . We intend to detect only one class corresponding to the LP, so the number of filters is fixed to 30. This also allows YOLO to better detect small objects since LP can occupy a small space of the whole image.

YOLO returns a confidence score for each LP bounding box it predicts. However, the bounding box will show up in the final prediction only if the confidence score is above a certain threshold. By changing this threshold value, we can directly impact the detection precision. A higher threshold means that only highly confident detections are returned, yielding fewer detections and therefore fewer false positives and more false negatives. A lower threshold yields more detections and, therefore, more false positives and fewer false negatives. In our case, the confidence value does not considerably affect the detection performance since each image in our datasets contains only one vehicle, so only one LP. So that, we keep only the detection with the most considerable confidence in cases where more than one LP is detected. In this work, the value of 0.25 is empirically chosen based on recall/precision on the validation set. It guarantees that YOLO does not predict an LP bounding box when the confidence is low. So only a good quality LP is detected, making sure the recognition module predicts a correct value.

Since all parameters are fixed, we train the YOLO network on the fully annotated LP regions. For the detection stage, we run the network over the full raw input images without any pre-processing. Since the LP is detected, it is cropped and sent to the second stage for recognition (see Figure 3.4).

The second stage of the proposed system is LP recognition. In this work, two different recognition engines are compared as described in the following.

The network architecture consists of two components, including the convolutional layers and the recurrent layers. The first approach considers the recognition of the LP characters as a sequence-labeling problem. The deep recurrent neural network with a bidirectional LSTM (BLSTM) is trained to recognize the sequential features extracted from the whole LP image via CNN. The main advantage of this approach is that it is segmentation-free; we need only LP images and their corresponding label sequences, avoiding the labor of labeling positions of individual plate characters. However, it fails when the background is changed or if the plate has characters of different heights, as discussed in a later section.

The CNN model is used to extract a sequential feature representation from the input LP image. The sequence of feature vectors is then used as the input of RNN, which considers the whole sequence history. To avoid gradient vanishing, LSTM is employed instead of the vanilla RNN unit. LSTM is a particular type of RNN unit, which consists

of a memory cell and gates. To integrate both past and future context into the model, we combine two LSTMs, one forward and one backward, into a bidirectional LSTM which is generally beneficial for image text recognition. The network configuration used in our experiments is summarized in Table 3.1. It is based on the architecture proposed in CRNN[113]. The architecture of the convolutional layers is based on the VGG architecture. To make it suitable for recognizing LP, max-pooling layers are modified to  $1 \times 2$  sized rectangular pooling windows instead of the conventional squared ones. The convolutional layers are succeeded by 2-layer BLSTM. The output of BLSTM is connected to Connectionist Temporal Classification (CTC) that extends the use of RNN to non-segmented data. In fact, by adding a new class (Blank), the CTC transforms the output of BLSTM into a conditional probability distribution over the classes (characters and blanks). Finally, the output from the CTC is decoded by removing the blanks into the most probable sequence of characters.

Before being fed into the network, all the images need to be scaled to the same height. The image width was chosen based on the Tunisian LP's ratio ( $4 \times 1$ ) to make it suitable for recognizing the Tunisian LPs. Experiments on the validation set have shown that setting up the input image resolution to  $174 \times 32$  helps to extract practical and valuable features for recognition. To train the network, we only need images and their corresponding label sequences. We define 23 character models corresponding to digits, letters composing the Arabic word "twns", the four Arabic letters "n, t, s, d", the Latin letters "C, D, R, S" and the space character. The complete recognition steps are shown in Figure 3.5.

The second LP recognition strategy proposed in this work is based on the YOLOv2 network, where segmentation and recognition are performed as part of the detection step. Detecting LP components directly from the cropped LP image would perform better than the pipeline detecting LP components on the total raw image since it will only need to search in the regions where LPs are found.

This module detects and recognizes all possible components in Tunisian plate including digits, the Arabic word "twns", the Arabic letters "n, t, s, d" and the Latin letters "C, D, R, S". This second detection/recognition approach presents many advantages compared to the free segmentation method presented previously. It is more suitable to recognize double-lined plates thanks to its versatility and ability to learn general components features independently of their positions in the LP. Moreover, it is more adapted to recognize Arabic words in the LP, which are considered a global text region, contrary to the CRNN, which recognizes a word by recognizing its character sequence. Finally, it can be trained to detect flag images in the LP, which can perturb the recognition process of CRNN, especially when it is located in the middle of the LP characters.

The entire system architecture is shown in Figure 3.6. To adapt the YOLOv2 architecture to detect the LP components, we have changed the number of filters in the last convolutional layer to 125 to match the number of classes (20 LP components).

LP components detection/recognition usually requires large annotated training sets. The creation of such datasets requires expensive manual annotation. In order to reduce



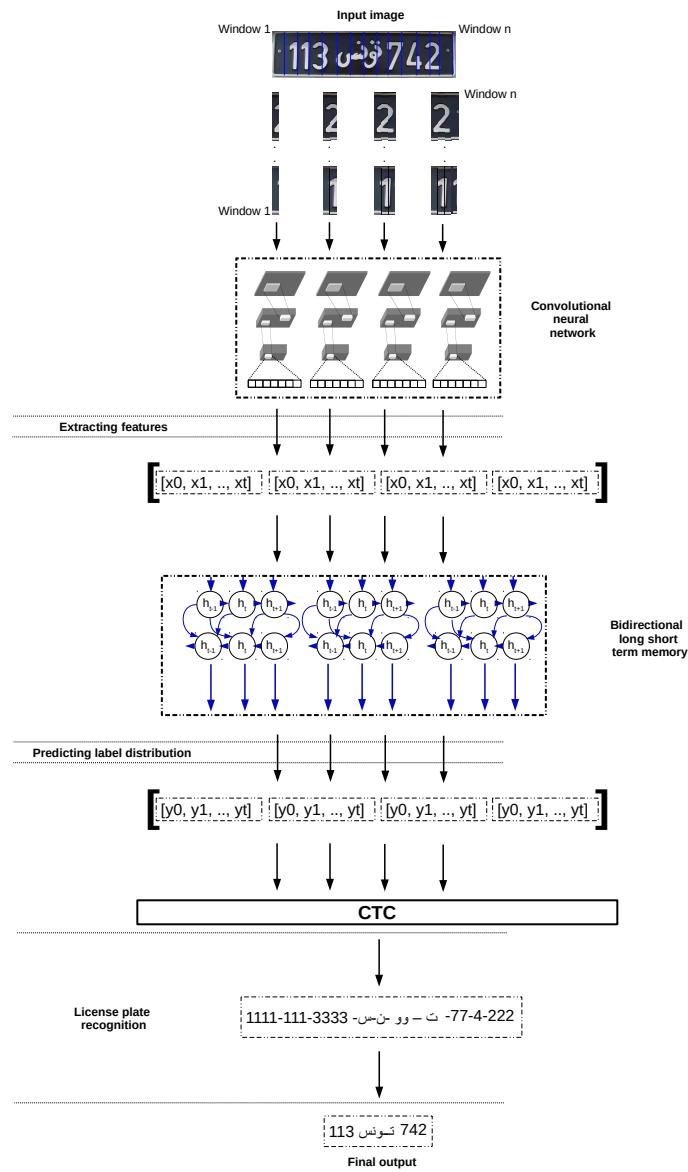


Figure 3.5: Architecture of the segmentation-free LP recognition engine

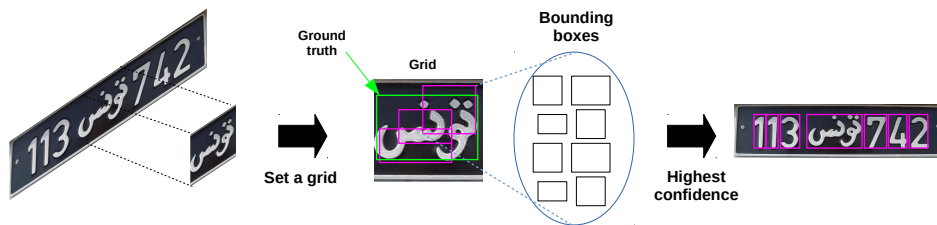


Figure 3.6: LP recognition pipeline using YOLO network



Layer Type	Configurations
Transcription	-
Bidirectional-LSTM	#neurons:256
Bidirectional-LSTM	#neurons:256
Map-to-Sequence	-
Convolution	#filters : 512, k : 2×2, s : 1, p : 0
MaxPooling	kernel : 1×2, s : 2
BatchNormalization	-
Convolution	#filters : 512, k : 3×3, s : 1, p : 1
BatchNormalization	-
Convolution	#filters : 512, k : 3×3, s : 1, p : 1
MaxPooling	kernel : 1×2, s : 2
Convolution	#filters : 256, k : 3×3, s : 1, p : 1
Convolution	#filters : 256, k : 3×3, s : 1, p : 1
MaxPooling	kernel : 2×2, s : 2
Convolution	#filters : 128, k : 3×3, s : 1, p : 1
MaxPooling	kernel : 2×2, s : 2
Convolution	#filters : 64, k : 3×3, s : 1, p : 1
Input	174 × 32 gray-scale image

**Table 3.1:** The network configuration of the convolutional recurrent neural network  $k$  : filter size,  $s$  : stride,  $p$ : padding

the time and cost of annotation processing while ensuring accuracy, we propose a semi-automatic annotation procedure. The core idea is using automatic incremental training to roughly annotate the LP image, just before the human review and revision.

To generate the automatic annotation, we start with a subset  $D$  of LP images. Each image in  $D$  is manually annotated at components level (bounding box + label), which we denote by detailed annotation. The algorithm takes as input a set  $SNL$  of LP images and their corresponding label sequences (without detailed annotations) and returns as output a set  $SL$  that contains all LP images that are automatically annotated at the components level. The proposed semi-automatic annotation algorithm proceeds in several iterations. We start with training YOLO on  $D$  to obtain the network model  $M_0$ . Then, for each iteration  $j$ , we construct a new subset  $S_j$  containing images of  $SNL$  that are correctly recognized by  $M_{j-1}$  and we fine-tune the YOLO network on  $S_j$  to obtain  $M_j$ . We reiterated this process until all the  $SNL$  are labeled at the components level, or no image of  $SNL$  can be correctly recognized. At each iteration,  $S_j$  is added to  $SL$ , and each LP image that is correctly annotated is removed from  $SNL$ . The detailed algorithm steps are given in Algorithm 1.

We present in tables 3.2 and 3.3 the evolution of the semi-automatic annotation on GAP-LP and Radar LP image datasets of varying levels of difficulty. We present the number of LP images that are correctly labeled at the component level for each

---

**Algorithm 1** Semi-automatic annotation of LP components
 

---

**Inputs :**

$D = \{I_1, \dots, I_N\}$  a subset of images  $I_i$  with labeled LP components (detailed annotation)

$SNL = \{I_1, \dots, I_M\}$  a subset of LP images and their corresponding sequence labels

$L = \{L_1, \dots, L_M\}$  (without detailed annotations)

**Output :**

$SL = \{l_0, \dots, l_k\}$  : a set of LP images  $l_i$  and their corresponding components labels

- 1: Train YOLO on  $D$  to obtain a CNN model  $M_0$
  - 2: Initialize  $j$  to 1
  - 3:  $SL = \phi$
  - 4: **repeat**
  - 5:    $S_j = \phi$
  - 6:   **for** each image  $I_i$  in  $SNL$  **do**
  - 7:     Test  $I_i$  with YOLO using  $M_{j-1}$  to detect its components labels
  - 8:     Concat LP components labels to obtain the sequence label  $GT(l_i)$
  - 9:     **if** ( $GT(l_i) == L_i$ ) **then**
  - 10:        $S_j \leftarrow S_j \cup \{l_i\}$
  - 11:        $SNL \leftarrow SNL \setminus \{l_i\}$
  - 12:     **end if**
  - 13:   **end for**
  - 14:    $SL \leftarrow SL \cup S_j$
  - 15:   Fine-tuning of Yolo on  $S_j$  to obtain  $M_j$
  - 16:   Increment  $j$
  - 17: **until**  $SNL == \phi$  or  $S_j == \phi$
-

iteration. We also present the percentage (the total of all iterations) of the correctly labeled LP images. It can be shown that the proposed algorithm annotates 95.11% of the dataset in only 7 iterations when starting with 500 images with detailed annotation. We show that 250 detailed annotated images are enough to annotate 96.10% of the dataset as shown in the second line of table 3.2. It is interesting to note that more than 90% of the LP images are annotated since the first two iterations. The proposed semi-automatic annotation algorithm is also evaluated on the Radar dataset containing LPs images in various conditions such as low resolution, severe weather conditions, and variable lighting conditions. As shown in table 3.3, more than 80% of the images are automatically annotated, starting with only 250 images with detailed annotation.

start_size/ Iteration	1	2	3	4	5	6	7
<b>500</b>	6150 (86.41%)	328 (91.02%)	168 (93.38%)	68 (94.33%)	38 (94.87%)	17 (95.11%)	×
<b>250</b>	5130 (72.08%)	1247 (89.60%)	328 (94.21%)	57 (95.01%)	35 (95.50%)	33 (95.96%)	10 (96.10%)

**Table 3.2:** Number per iteration and percentage (the total of all iterations) of annotated LP images on GAP-LP dataset

start_size/ Iteration	1	2	3	4	5	6	7
<b>500</b>	1263 (63.18%)	124 (69.38%)	141 (76.43%)	55 (79.18%)	38 (81.09%)	23 (82.24%)	13 (82.89%)
<b>250</b>	380 (19%)	934 (65.73%)	197 (75.58%)	67 (78.93%)	24 (80.14%)	16 (80.94%)	×

**Table 3.3:** Number per iteration and percentage (the total of all iterations) of annotated LP images on Radar dataset

The annotation quality of the semi-automatic annotation process is evaluated using 250 LP images with hand annotated component locations. We obtained an average IoU of 88 for the GAP-LP dataset and of 85 for Radar dataset which confirms the robustness of the proposed algorithm.

Our method aims primarily to tackle the problem of large variations between LPs, figure 3.7 show a sample of possible variations in the background.



**Figure 3.7:** *Background changes in LPs.*

The first approach using CRNN for segmentation free recognition is a lot more sensitive to background changes while using the robustness of an object detector i.e. the second approach allowed it to perform well on changing backgrounds.

Furthermore, view angle changes and two-lined LP show in figure 3.8 pose more challenges. In particular, for the segmentation free approach. Our second approach however, is capable of recognizing the LP successfully since a CNN is not affected by the location of the object in an image.



**Figure 3.8:** *Background changes in LPs.*

### 3.4 Results and Discussion

We conduct experiments to verify the effectiveness of the proposed ALPR system. All the experiments were performed on an NVIDIA GTX 1080 Ti GPU (3584 CUDA cores and 11 GB of RAM). Experiments were conducted in two datasets collected from real road surveillance and parking access control environments.

It can be observed from table 3.4 that our method achieved impressive detection rates with a precision of 100% and a recall of 100% on the GAP-LP dataset and 99.09 on the Radar dataset. Here a detection is considered correct if the overlap between the detection and the ground-truth bounding box is greater than 0.5 ( $IoU > 0.5$ ).

Dataset	GAP-LP	Radar
Recall	100%	99.09%
Precision	100%	100%

**Table 3.4:** *LP Detection: precision and recall rates for IoU threshold of 0.5*

In Table 3.5, we show the LP detection recall values considering different IoU acceptance values (from 0.3 to 0.8). As can be observed, our method is not much sensitive for IoU variations, except for very high values.

DatabaseIOU	30	40	50	60	70	80
Radar (Recall %)	100	99.79	99.09	97.93	85.49	65.88
GAP-LP (Recall %)	100	100	100	99.06	93.70	67.83

**Table 3.5:** *LP Detection: recall rates for IoUs ranging from 0.3 to 0.8*

Figure 3.10 shows some qualitative LP detection examples from the Radar dataset in different illumination conditions. The results confirm the robustness of the proposed detection approach.

The LP recognition rate (LP-RR) is defined as the number of correctly recognized LPs divided by the total number of ground-truth samples. A correctly recognized LP means all the characters of the LP are recognized correctly. We also report the character recognition rate (CRR), defined as the number of correctly recognized LP characters divided by the total number of ground-truth character samples.

We present in tables 3.6 and 3.7 the LP recognition accuracy of the segmentation free approach based on the CRNN architecture. The obtained results show that the CRNN successfully recognizes 95.88% of the LPs on the GAP-LP dataset. An improvement of 0.88% is achieved compared to YOLO when using the same training dataset (GAP-LP). Contrarily, the performance on the Radar test set is vastly decreased to 42.88% when the Radar training set is used. This can be explained by the reduced size of the Radar training set. Using a more extensive training set composed of a mix of Radar and GAP-LP datasets, an improvement of 30.15% is obtained. We conclude that CRNN needs a more extensive training dataset than YOLO to correctly learn the parameters of CNN and LSTM. Hence, we have tried to expand the size of the training dataset with synthetic LP images.

As concluded above, achieving a higher performance requires many annotated LP image data covering various situations. To tackle this problem, we develop a script to synthesize images of LPs with different fonts, colors, and component composition rules. Figure 3.9 shows some examples of the synthesized LP images. We have generated 40,000 LP training images.

Training Dataset	GAP-LP	Radar	GAP-LP + Radar
LP-RR	<b>95.88%</b>	92.88%	94.25%
CRR	<b>98.83%</b>	96.33%	98.11%

**Table 3.6:** LP and character recognition rates obtained by CRNN on GAP-LP dataset

Training Dataset	GAP-LP	Radar	GAP-LP + Radar
LP-RR	73.03%	42.88%	<b>78.13%</b>
CRR	91.03%	80.33%	<b>92.31%</b>

**Table 3.7:** LP and character recognition rates obtained by CRNN on Radar dataset

Experiments show that the LP recognition system based on YOLO architecture trained with the additional generated data did not improve the prediction accuracy. Contrary, generated data slightly improves the performance of CRNN as shown in table 3.8. An improvement of 2.73% is achieved on the Radar test set and 0.05% on the GAP-LP test set.



**Figure 3.9:** Synthesized images of LPs

Test Dataset	GAP-LP	Radar
CRNN	95.93%	80.86%

**Table 3.8:** LP recognition rates obtained by CRNN using generated data

Figure 3.10 illustrates some of the recognition results obtained by the proposed system on the GAP-LP and Radar datasets. It is noteworthy that our system can generalize well and correctly recognizes LPs under different lighting conditions, as presented in the first two rows of figure 3.10. The following rows show examples of incorrectly recognized LP by CRNN or by YOLOv2, or by both. It can be shown that CRNN can not recognize double-lined LP, contrarily to YOLO, which is more suitable to recognize them thanks to its versatility and ability to learn general component features independently of their positions. We notice that some LPs written in red on a white background are not correctly recognized by neither YOLO nor CNN. This is not surprising as the training datasets do not include enough LP images with this norm. In some other cases, the LP recognition fails due to the inclination of the LP or to the presence of noise.



**Figure 3.10:** Qualitative results obtained by the proposed ALPR system in the GAP-LP and Radar datasets. Green text refers to correctly recognized LP, while miss-recognized LP are written in red.

### 3.4.1 Comparison with state-of-the-arts ALPR systems

To compare the proposed system with state-of-the-art ALPR systems on GAP-LP and Radar datasets, two commercial systems Sighthound [86] and OpenALPR<sup>2</sup> are evaluated. Unfortunately, the comparison was not possible because the two commercial systems do not support Arabic script and fail to recognize Tunisian LPs. Alternatively, we have conducted additional experiments on the AOLP public dataset [51], which has 2049 images of Taiwan LPs. This database is categorized into three subsets with different levels of difficulty: access control (AC), traffic law enforcement (LE), and road patrol (RP). The detection results of our LP detection system are compared to three state-of-the-art LP detection systems. Note that all ALPR systems are evaluated in the same training and test sets. It can be shown from Table 3.9 that our LP detection approach produces a higher recall and precision for each dataset category. The detection results of experimentation for the AOLP dataset are provided in Table 3.9. It can be shown that our LP detection approach produces a higher recall and precision for each dataset category.

Method	AC		LE		RP	
	Recall	Precision	Recall	Precision	Recall	Precision
<b>Proposed</b>	99.82	100	96.42	100	99.17	100
<b>Hsu, Chen, Chung[51]</b>	96	91	95	91	94	91
<b>Selmi, B H, A[110]</b>	96.8	92.6	93.3	93.5	96.2	92.9
<b>Li, Wang, Y, Shen[69]</b>	98.38	98.53	97.62	97.75	95.58	95.28

**Table 3.9:** Comparison of plate detection rates on three subsets of the AOLP dataset

We present in table 3.10 the character and the plate recognition rates of the proposed CRNN and YOLO recognition approaches, and we compare their performance against five state-of-art results published on the AOLP dataset. The obtained results show that our LP recognition approaches provide the best performance in terms of character and LP accuracies.

### 3.4.2 Runtime Evaluation

Experiments have been done on a computer with 2.6 GHz Xeon, 64 GB RAM, and NVIDIA GTX 1080 Ti GPU. We implemented our system on Linux Ubuntu operating system. For each processing stage, the average runtime has been computed from different runs made in the experiment. The time computation of the LP identification step and the LP recognition step are given in table 3.11. It is worth noting that our system can process (detection + recognition) LP images in 0.0443 s with CRNN and in 0.0487 s with YOLOv2, which is sufficient for real-time usage.

<sup>2</sup>OpenALPR Cloud API, <http://www.openalpr.com/cloud-api.html>



Method	AC		LE		RP	
	CRR	LP-RR	CRR	LP-RR	CRR	LP-RR
<b>Proposed (CRNN)</b>	99.49	97.97	98.97	95.95	95.62	81.81
<b>Proposed (YOLO)</b>	95.31	93.75	97.62	98.91	93.43	90.42
<b>Hsu, Chen, Chung[51]</b>	95	88.5	93	86.6	94	85.7
<b>Selmi, B H, A[110]</b>	96.2	-	95.4	-	95.1	-
<b>Li, Wang, Y, Shen[69]</b>	-	94.85	-	94:19	-	88:38
<b>Jiao, Ye, Huang[61]</b>	90	-	86	-	90	-
<b>N.E.An, E.An, L, K[3]</b>	92	-	86	-	91	-

**Table 3.10:** Comparison of plate Recognition rates on three subsets of the AOLP dataset

Stage	Time (s)
<b>Detection</b>	0.0367
<b>Recognition (CRNN)</b>	0.0076
<b>Recognition (YOLO)</b>	0.012

**Table 3.11:** The computational time required in each ALPR stage

## 3.5 Conclusion

We have presented a robust real-time ALPR system using a pipeline with two deep learning stages. The LP detection stage is based on the state-of-the-art YOLOv2 object detection CNNs. For the second stage, we compare two recognition engines: a sequence labeling method that recognizes the whole LP without character-level segmentation and a joint detection/recognition approach that performs the recognition on the plate component level. The proposed system is robust to illumination and weather conditions and can achieve a full LP recognition rate of 97.67% in the GAP-LP dataset and 91.46% in the Radar dataset, a reasonable computational time. We also introduced a new public dataset for multi-norm and multilingual ALPR that includes 9,175 fully annotated images. Compared to the existing datasets for this task, GAP-LP is the largest ALPR dataset making it suitable for trying and evaluating deep learning techniques. In order to reduce the time and cost of annotation processing, we have proposed a new semi-automatic annotation procedure of LP images with labeled component bounding boxes. Future work consists of integrating vehicle make and model recognition to improve the vehicle identity recognition process and help check correlation with data stored on police and homeland security databases.

# Chapter 4

## Vehicle Make and Model Classification

### 4.1 Literature Survey

Early works on vehicle model recognition focused on low-level features representation: [99] uses SIFT ([84, 82, 83]) to describe make-model instances, but it is computationally expensive. Hence, SURF ([6]) and HOG were adopted by [50] for more robustness and speed. Other variations of SURF that have also been used include (1) FAST, a key-point detection method designed for real-time application, (2) BRIEF, a short descriptor, and Oriented FAST, which uses FAST for key-points detection and BRIEF as a descriptor.

Unlike the standard feature extraction algorithms (e.g., SIFT, HOG), CNN uses several hidden layers to learn a high-level representation of the image hierarchically. Convolution filters (or kernels) on the image allow the network to extract more relevant features. Activation functions and pooling layers allowed the network to be more robust to scaling, translation, and rotation variations. Moreover, high-level features representation creates resilience to noise. CNN became a prevalent research subject in the computer vision community. In particular, in VMMR, part-based approaches are yielding impressive results.

Works done by He, Shao, and Tan[44] propose to detect make and model from surveillance camera by first, detecting front-view components such as the grilles, plate, and lights. Then, specialized CNN's classify each part of the vehicle, yielding a global car classification. While this method achieves good performance, it is limited to front viewpoints. However, showing that not aggregating local features to a global representation enhance the CNN performance.

A different approach to part-based recognition employed by Biglari, Soleimani, and

Hassanpour[9] is to find the most relevant parts for each class; one model is learned per class, so a cascade of classifiers is applied to the input. Hsieh, L. Chen and D. Chen[50] uses SIFT-like local descriptors to train weak classifiers over a grid of vehicle parts.

SWP-CNN[52] propose Spacial Weighted Pooling (SWP) instead of the standard pooling in a CNN. SWP layer provides the fully connected layers with robust feature representation by magnifying features corresponding to discriminative parts of the image. But the SWP layer's performance drops with significant variation in scale and position.

WindowResnet[32] propose a deep convolutional architecture built upon multi-scale attention windows. Through those windows, the most discriminative parts of the vehicle are aggregated over different scales. The model uses Residual Networks RESNET[45] with Spatial transformer networks (STN) [55] to improve resilience to affine transformations. However, in an STN with multiple feed-forward alignment modules, the output image of the previous alignment module is directly fed into the next. This is problematic as it can create unwanted boundary effects as the number of geometric prediction layers increase.

A different approach to the latter was proposed by BoxCars[120], by extracting additional data from the surveillance video stream, besides the vehicle image itself, and feeding it into the CNN boosts the recognition performance. This additional information includes a 3D vehicle bounding box used for "unpacking" the vehicle image.

Researchers in vehicle model recognition have either worked on global representation or discriminative local representation. However, VMMR is both a fine-grained and a coarse-grained classification problem making a global representation necessary as part-based representation.

Many datasets have been proposed in this research field [27, 7, 1, 88]. In this work, we test our method on a publicly available dataset, namely CompCars [1] dataset. Other sets, also presented, are less varied than CompCars. BVMMR[88] for an instant, is exclusive to Iranian vehicles.

## 4.2 Datasets and Evaluation Protocols

### 4.2.1 The Comprehensive Cars (CompCars) Dataset

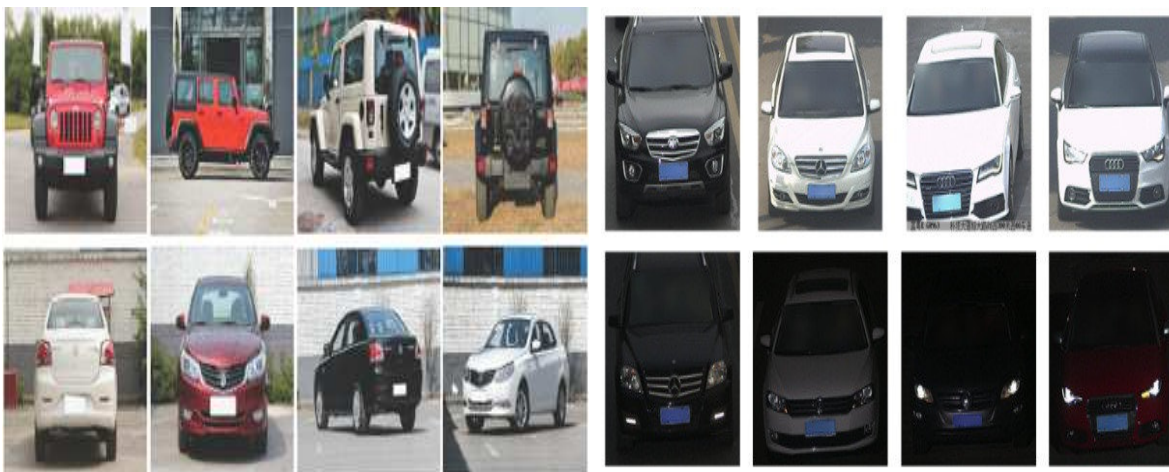
The Comprehensive Cars (CompCars) [1] is a publicly accessible dataset containing data from two scenarios: (1) Web-data a total of 136, 727 images of the entire car and 27, 618 capturing the car parts. The database respects the standard hierarchy with

163 car Marks and 1, 716 car models covering most commercial car models from 2005 to 2015. In terms of viewports there are five: 'Front-view', 'Rear-view', 'Side-view', 'Front-Side' and 'Rear-Side'. Table 4.1 shows the total number of images per viewport and the average number of view-port images per model. (2) The surveillance data contains 50, 000 car images captured in front view.

View-port	No. in total	No.per model
Front	18431	10.9
Rear	13513	8.0
Side	23551	14.0
Front Side	49301	29.2
Rear Side	31150	18.5

**Table 4.1:** *Quantity distribution in different view ports.*

The web dataset contains most of the 1716 car models however the CompCars article [1] proposes a train/test split on 431 models. We adopt this split to compare our work to state-of-the-arts results. Figure 4.1 shows different examples of the web and surveillance dataset.



**Figure 4.1:** *Web (left) and Surveillance (right) datasets.*

## 4.2.2 Other datasets

In this work, we focus our analysis and results solely on CompCars. Compared to other make and model classification datasets, CompCars has more diverse vehicle models with images from different view-ports. The following sections present an overview of some the most known publicly available datasets.

### The Cars dataset

The Cars dataset contains in total 16,185 images. The training data has 8,144 images and 8,041 images in the test set, The dataset has 196 classes where each class has been split roughly in a 50-50 split. The quality of images is similar to the CompCars Web dataset. Figure 4.3 shows the quality of images in the Cars dataset.



Figure 4.2: *The Cars dataset.*

### The VMRRdb dataset

The Vehicle Make and Model Recognition dataset (VMRRdb) is contains 9,170 classes consisting of 291,752 images. Models in the VMRRdb are manufactured between 1950 and 2016.



Figure 4.3: *The VMRRdb dataset.*

### The BVMMR dataset

BVMMR dataset is relatively small. It includes 676 models of 19 different make. For every image, there are a frontal and rear view taken with an 8 megapixel camera. The main draw backs of this dataset that it contains only Iranian vehicles with images taken from front view.



Figure 4.4: *The BVMMR Dataset.*

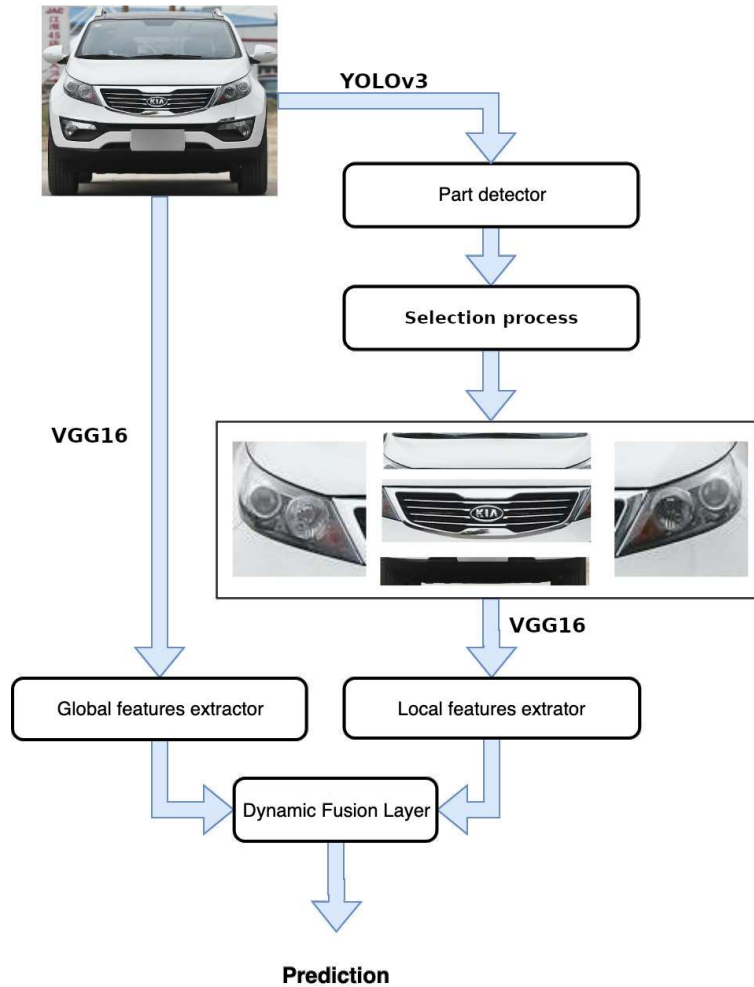
## 4.3 Material and Methodology

We propose a multi-stream robust architecture to extract and combine both local and global features representations for VMMR. Our system comprises separate modules that extract features from cropped images of vehicle parts (one module per part) and the main module that extracts features from the whole image. A novel technique for aggregating the different outputs into a final prediction is proposed.

First, a part detector is employed to detect available parts in an image; then, a multi-stream architecture is used to feed parts into their respective modules and the whole image to the main module. Extracted features are then concatenated and, finally, aggregated to a final prediction using the novel fusing method. Multiple challenges are implicated in such architecture. The number of detected parts varies from one image to another, making the feature vector of different sizes depending on the output of the part detector. The size of the entire system can cause performance problems as the number of parts increases. Furthermore, our experiment shows that a good choice of parts can be critical to better performance. However, combining a part-based representation with a global perspective allowed the system to capture both the coarse and fine-grained nature of vehicle make and model classification. In the following sections, we demonstrate the unique problems the VMMR poses and our system’s resilience to them.



An overview of the main steps of our system are depicted in Figure 4.5.

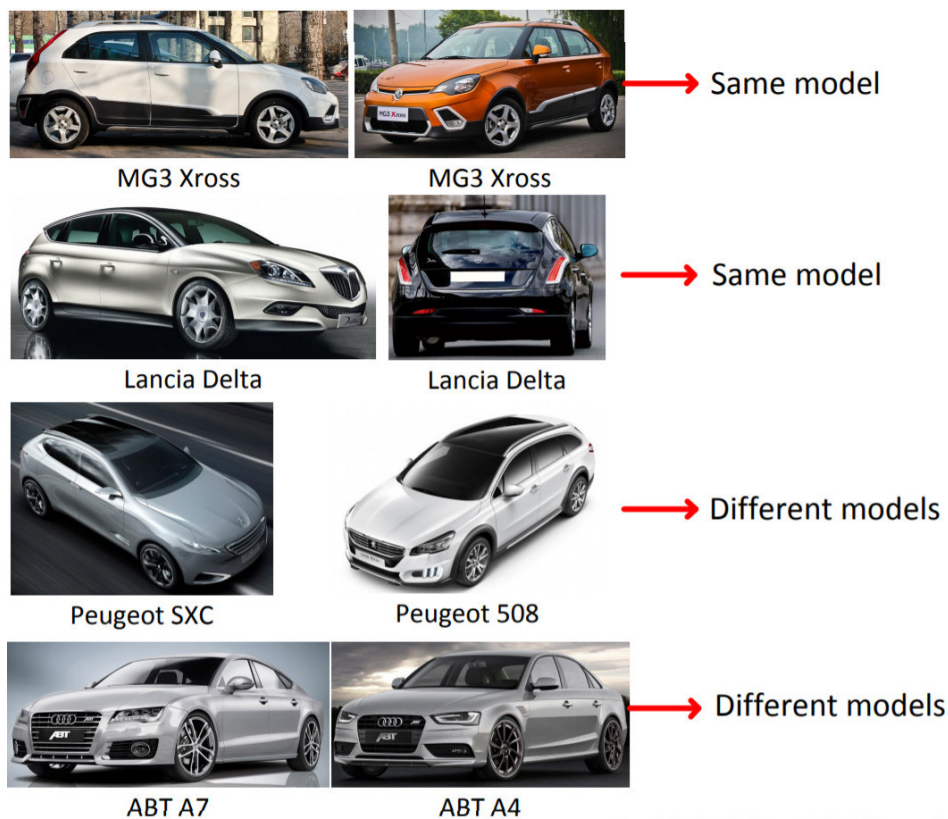


**Figure 4.5:** *system's main steps. YOLO first processes the image to detect the vehicle parts. Global and local features are then extracted from the full vehicle image and the selected parts using VGG. The global and the local representations are then fed to the dynamic fusion layer to perform the final classification.*

Our approach starts by extracting a set of parts  $\{P_1, P_2, \dots, P_n\}$  from a given image  $I$ . Since vehicle pose may vary across images, we do not assume that all parts appear in  $I$ . Conventional detection methods learn classifiers to perform detection. Classifier rules are generally evaluated on a sliding window, and binary output is computed in the corresponding location. More recent deep learning-based architectures get around the sliding window techniques and produced higher performance while being faster. Following this advance, we choose to use YOLO [105] to detect each vehicle part.

Thanks to the encoded context information, the YOLO detector provides robust results in all our experiments.

Vehicle manufacturers copy traits from previous models to produce new ones. This increases the complexity of the VMMR task since the model recognition relies on the subtle variations between vehicle parts. Figure 4.6 shows the subtle differences between models. To detect these subtle variations, we employ a multi-stream architecture to apply specialized features extractors for every part and every combination of parts. Figure 4.6 also shows same models with different variations. In this case, a global representation may benefit the recognition task rather than the part information. Our multi-stream architecture successfully combines both global and local representations. It also provides a flexible system that can use any available stream that feeds the input.



**Figure 4.6:** *Examples of subtle differences between vehicle models. From a global perspective the vehicles seem similar, yet they belong to different vehicle models. This is due to very subtle variations.*

The multi-stream architecture provides important advantages such as robustness and specialized feature extractors. However, the resulted features will have different shapes depending on the used streams. Moreover, a single static classification layer may not be sufficient for representing all of the variations of the multi-stream architecture.

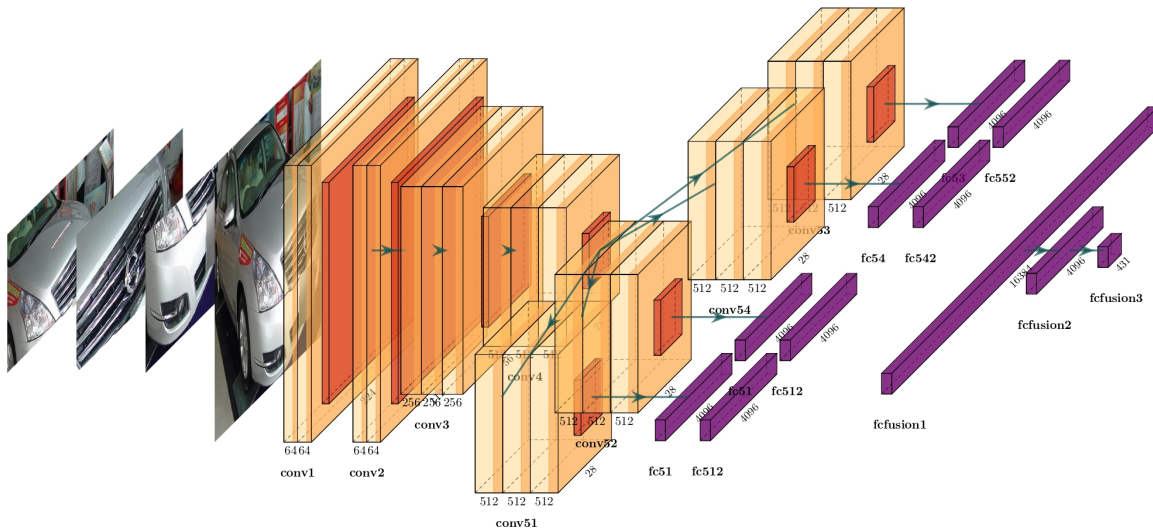


As a solution, we introduce a dynamic fusion layer which only considers relevant weights that fit the input and swap the others at the run-time.

Each part has a weight matrix  $W$  of shape  $4096 \times 431$ , 431 is the number of classes, and a bias vector  $b$  of shape 431. The system stores in memory all of the weights and biases. Then, at run-time, depending on available parts, the system dumps unrelated weights and biases to the memory and load only the weights and biases of present parts.

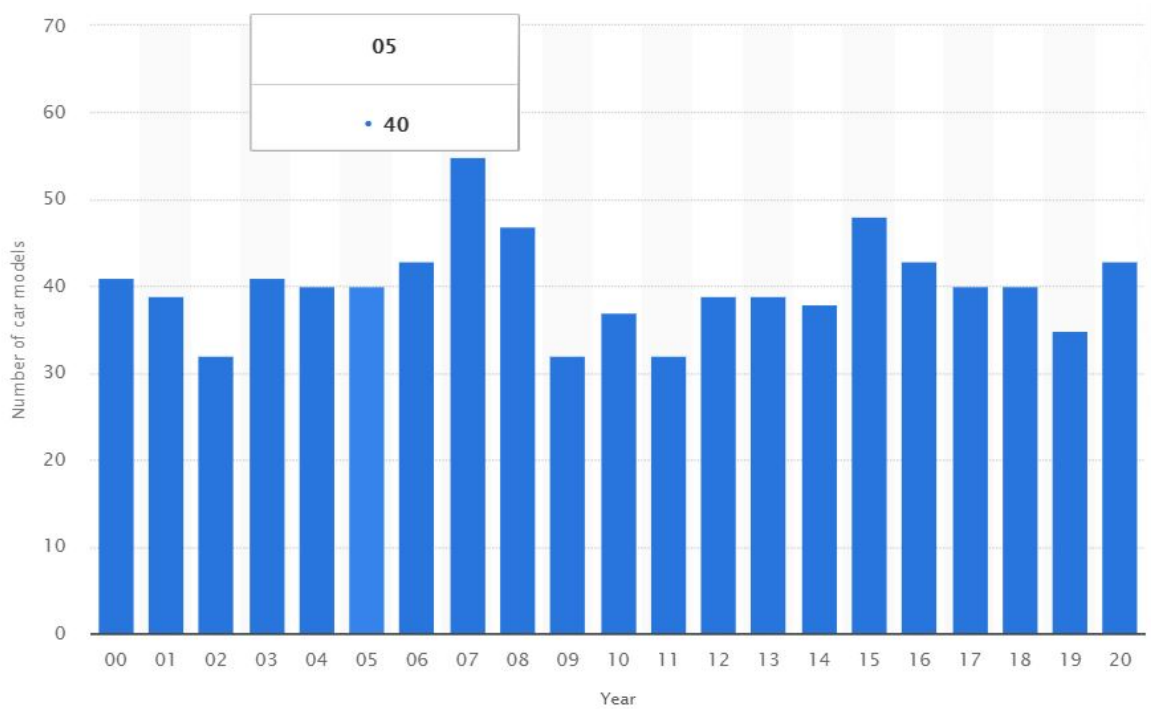
This technique allows the fully connected layer to have variable input shapes and to store part-specific features.

The model architecture for each stream is composed of three main sections. (1) The shared section: convolutional blocks common to all parts and the entire image. (2) specialized feature extractors: convolutional blocks and fully connected layers specialized for each vehicle part and the entire image. (3) The dynamic fusing layer for features aggregation and classification. Figure 4.7 shows the model architecture of a single stream. The Cross-entropy loss, or log loss  $\sum_{c=1}^M y_{oc} \times \log(p_o, c)$  where  $M$  is the number of classes,  $y$  is a binary indicator (0 or 1) if class label  $c$  is the correct classification for observation  $o$  and  $p$  is the predicted probability observation  $o$  is of class  $c$ . Log loss is used for all of the training sessions. The loss is back-propagated until the fifth convolutional layer leaving the first four layers unchanged.



**Figure 4.7:** Model architecture. Initially, parts are introduced to the model sequentially from a single input. The batch  $B$  is composed of  $n$  mini-batches  $\{b_1, b_2, \dots, b_n\}$ ,  $n$  being the number of parts, each mini-batch contains similar parts  $\{p_1, p_2, \dots, p_k\}$  for  $k \leq n$  so the batch size is  $k \times n$ . The images are passed through the first four convolutional block then the batch  $B$  is split into  $n$  groups  $\{\{p_{1_1}, p_{1_2}, \dots, p_{1_n}\}, \{p_{2_1}, p_{2_2}, \dots, p_{2_n}\}, \dots, \{p_{k_1}, p_{k_2}, \dots, p_{k_n}\}\}$  each image in the group is passed to a part-specific fifth convolutional block. Finally, the image and its parts features are aggregated with the dynamic fully-connected layer.

The multi-stream architecture offers an essential advantage over single-stream systems. Although it has been overlooked by most research, this advantage can have great importance in the practical usage of VMMR. As new models are produced with an average of 38 models per year in the US (Figure 4.8), VMMR systems are required to incorporate new vehicle models without disturbing the system.



**Figure 4.8:** *Models yearly production in the US in 2000-2020*

The multi-stream nature of our systems enables new models to be incorporated into separate streams. Hence, older models' classification will not be affected.

### 4.3.1 Parts qualitative study

The difference between parts is considerable, and there is an upper limit on the number of parts to use. Hence choosing the right parts can have a significant impact on the model performance. Thus we did a study on the factors that can determine the best parts to use. This variation in the discriminative nature of parts can be traced back to two main categories of factors: (1) The intrinsic nature of the parts and environment. The grills, for instance, serves a particular purpose and must have a specific geometry. (2) Design principles, proportions between parts, and the ecstastic appeal of the design impose specific criteria that all models must adhere to.

#### **Intrinsic nature of the parts and environment**

Parts that contains additional information yields better results. For instant, the mark logo or model name on the Trunk and the Grilles shown in figure 4.10.

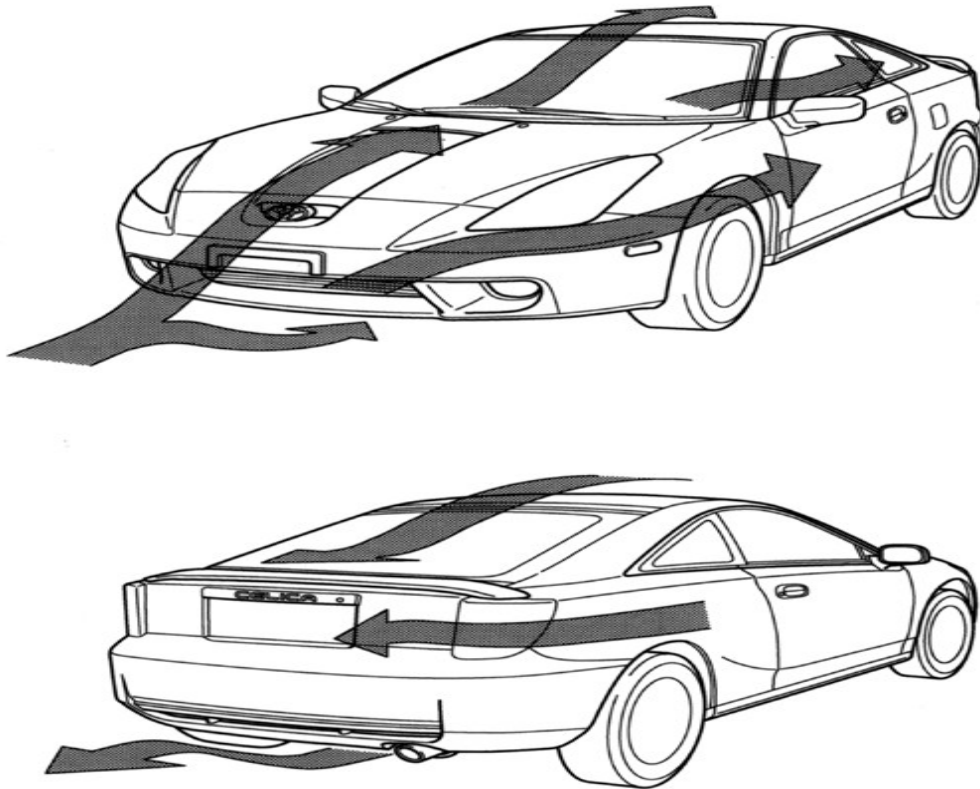


**Figure 4.9:** Logo of the vehicle's mark on the grilles.

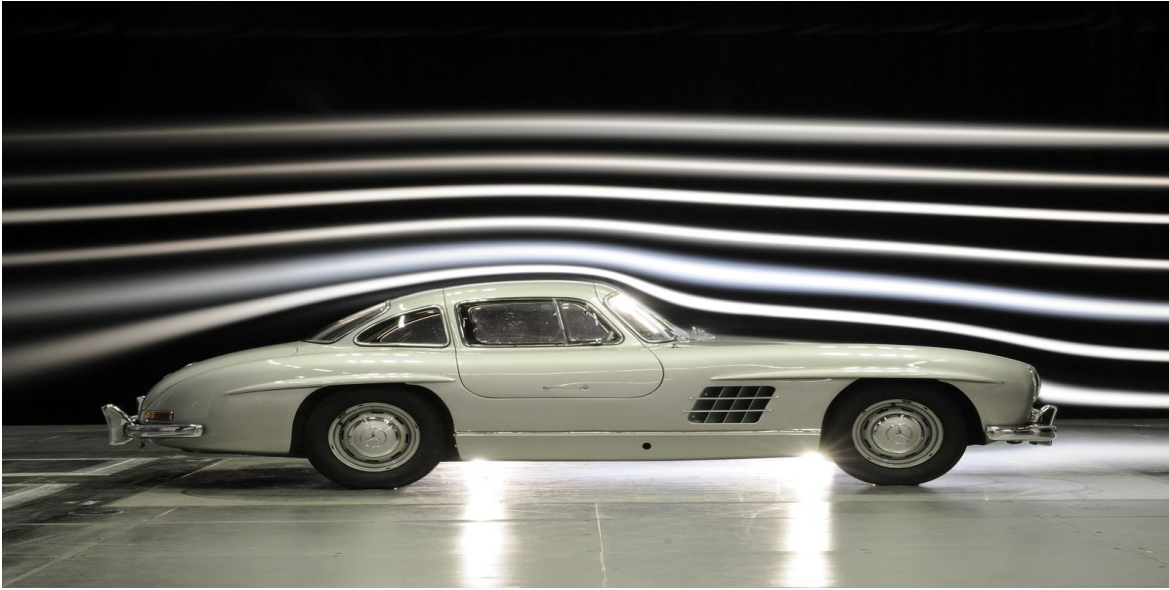


**Figure 4.10:** Logo of the vehicle's mark and the model the bumper.

For a vehicle to move at a certain speed, it needs to adhere to the physical laws of vehicular aerodynamics. The air coming from the front of the vehicle is split into six dominant air streams. These air streams cover multiple parts of the vehicle, as shown in figure 4.11. Hence, every part should have a round shape not to block the air and create resistance that will slow or damage the vehicle. Figure 4.12 is an example of how the round shape allows the air to travel smoothly across the upper parts.



**Figure 4.11:** Air flux smoothly going over a vehicle.



**Figure 4.12:** *Logo of the vehicle's mark and the model the bumper.*

Both figures 4.11 and 4.12 show that the rear bumper and the trunk do not impact the air stream. Hence, the design of those parts can have great variation. As is supported by table 4.3.

We also see a drop in performance both in front and rear view in the left and right light. This is due to a lack of variation in design, i.e., there is not much visual change in vehicle lights across different models. Norms and the technology itself mainly impose this: Most of the part is composed of glass, leaving variations only on the edges as it shows in figure 4.13.



**Figure 4.13:** *Front and Rear lights with little room for design variation.*

### Design principles

As with any other artificial object, vehicles tend to follow specific proportions that do not necessarily have functional purposes but rather an ecstatic one. Figure 4.14 show different makes and models following the same proportions. The distance between the wheels is equal to three times the diameter of a wheel; the distance between the hood and the wheel is also equal to one wheel. Moreover, the height of a window is one-third



the vehicle height. All these constraints impose more minor variations between models on vehicle sides.

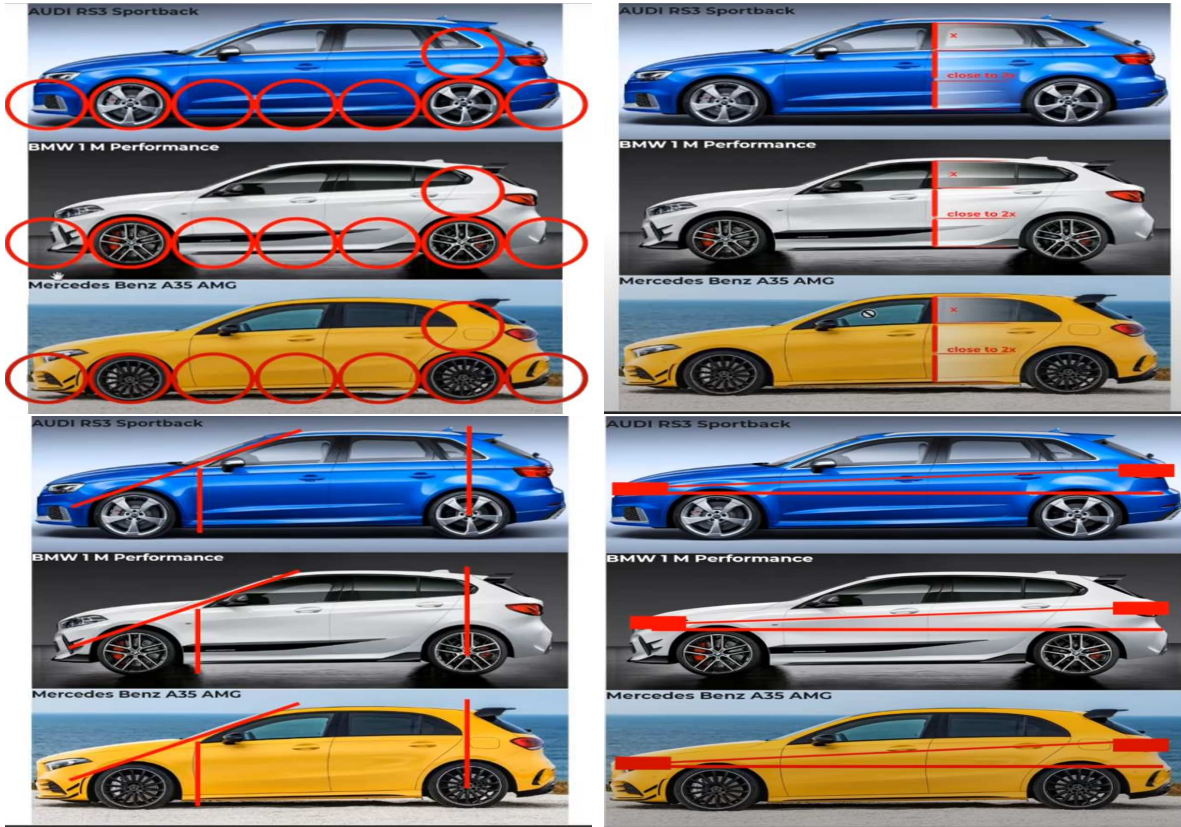


Figure 4.14: Same design proportions across different makes and models.

## 4.4 Results and Discussion

### 4.4.1 Results from Individual Parts

Table 4.2 and 4.3 shows the results of the VGG16 with Batch normalization on their respective cropped parts. Every set of cropped parts is a dataset with a train-test split. The cropped parts train set is a sub-set of the training set; the same goes for the sub-set of the test set.

Some parts are more descriptive than others. For example, in the front-view-bumper, the baseline VGG16 achieves 92.60% while the front-right-light baseline VGG16 achieves 61.76%. The grilles and the trunk, the only two parts that have the make logo and model name, has the best results while left and right lights on both front and rear have the least results. As mentioned in the previous section, the lights are mainly composed of glass so most variation is on the surroundings which does not allow for

much variation between models.

Part	VGG16	Train	Test
Bumper	92.60%	7022	6740
Hood	92.26%	6988	6757
Grilles	93.83%	6622	6385
Left Light	64.72%	5875	5683
Right Light	61.76%	5942	5790

**Table 4.2:** *Individual front parts of CompCars’s web data.*

Part	VGG16	Train	Test
Bumper	89.56%	5603	5312
Trunk	93.90%	5225	5221
Left Light	72.89%	5124	4859
Right Light	82.10%	5587	5300

**Table 4.3:** *Individual rear parts of CompCars’s web data.*

#### 4.4.2 Multi-Stream Dynamic Fusion

In this section, we compare different combinations of parts, table 4.4 shows the recognition rate of best performing combination per view-port, 19 combinations in total. On average, the front is more discriminative than the rearview: combination 9, where all are front parts with no full image is immensely better than its counterpart from the rear (84.12% - 65.45%). Adding more parts can lower the performance; for example, combination 1 is better than combination 7, although all parts in 1 are also in 7. Some parts can dramatically reduce the performance, for instance, combination 19 versus combination four or combination 5, from 91.23% to 45.36% and 46.85%, respectively.

Index	Combination	VGG16
1	full image + FB	96.12%
2	full image + FH	94.33%
3	full image + FG	92.74%
4	full image + FLL	45.36%
5	full image + FRL	46.85%
6	All Front parts	91.36%
7	full image + FB + FH + FG	92.65%
8	full image + FLL + FRL	90.12%
9	FB + FH + FG + FLL + FRL	84.12%
10	full image + RB	91.11%
11	full image + RT	94.14%
12	full image + RLL	68.12%
13	full image + RRL	65.45%
14	full image + RB + RT + RLL + RRL	88.24%
15	RB + RT + RLL + RRL	65.26%
16	full image + RB + RT + RLL	87.26%
17	full image + RB + RT	89.22%
18	full image + RLL + RRL	62.25%
19	full image	91.23%

**Table 4.4:** Fusion using different combinations on the CompCars’s web data. FB: Front Bumper, FH: Front Hood, FG: Front Grilles, FLL: Front Left Light, FRL: Front Right Light, RB: Rear Bumper, RT: Rear Trunk, RLL: Rear Left Light, RRL: Rear Right Light

Table 4.5 shows recent results on the CompCars dataset where approaches with no deep convolutional networks achieve the worst result, such as Yang [1]. BoxCars [120] and Baseline VGG16 [119] rely on deep networks for global features representation. However, best results such as SWP-CNN[52] and WindowResnet[32] used part-based approaches. Our approach combined both global and local representation allowing the system to be robust.

Approach	CompCars (web)
Yang[1]	76.7%
BoxCars[120]	84.8%
Baseline VGG16	92.66%
Ours (VGG16)	95.07%
SWP-CNN[52]	97.6%
WindowResnet	97.8 %

**Table 4.5:** Comparison with our approach.



# Chapter 5

## Vehicle Re-Identification

### 5.1 Literature Survey

#### 5.1.1 Object Re-identification

##### Evaluation Metrics

In order to evaluate the performance of V-reID methods, Three different criteria are commonly used:

- The Cumulative Matching Characteristic (CMC). The CMC curve shows the probability that a query appears in the candidate list. Each element in the list is calculated as  $CMC_k = \frac{\sum_{q=1}^Q gt(q,k)}{Q}$ , where  $Q$  is the number of total query instances.  $gt(q, k)$  is 1 when the ground-truth of  $q$  appears before  $k$  and 0 otherwise.
- The Mean Average Precision (mAP). It is used to evaluate the overall performance. It is defined as:  $mAP = \frac{\sum_{q=1}^Q AP(q)}{Q}$ , where the average precision  $AP$  is defined by:  $AP = \frac{\sum_{k=1}^n P(k) \times gt(k)}{Q}$  and  $P(k)$  corresponds to the precision at rank  $k$ .
- The rank-1, rank-5, .., rank- $k$  scores... measures the precision at  $k$ , i.e., the sum of test samples where the ground truth is in the top  $k$  predictions over the total number of samples in the test set (query set).

##### Person Re-identification

Although V-reID and Person Re-identification (P-reID) are two very different problems, vehicles with the same model issued in the same year are identical. However, research on vehicle re-identification was heavily influenced by person re-identification research

and vice-versa.

P-reID has been widely studied as a specific person retrieval problem across non-overlapping cameras Gong et al.[37], Zheng et al.[140]. Given a query person-of-interest, the goal of P-reID is to determine whether this person has appeared in another place at a specific time captured by a different camera.

### Vehicle Re-identification

Earlier V-reID methods [136, 20, 78] heavily rely on the accordance of handcrafted feature extraction, using for instance SIFT [82] or HOG [23], and a supervised feature classification step using conventional classification algorithms [21, 11]. In the era of deep learning ImageNet [65], the reliance on manually designed features has been reduced, and single-step (end-to-end) models have been introduced, outperforming earlier methods [139, 143, 145]. Zang et al. [139] have studied the application of triplet-wise training of a CNN. The CNN has been adopted to automate the feature extraction from images, and the training adopts triplets of (query, positive example, negative example) to capture the relative similarity between them to learn representative features. The authors proposed improving the triplet-wise training by adding a classification-oriented loss and a different triplet sampling method based on pairwise images. In order to boost the feature learning process, Zhu et al. [145] proposed a shortly and densely connected CNN (SDC-CNN). The proposed SDC-CNN mainly consists of short and dense units (SDUs), pooling, and normalization layers. Each SDU contains a shortlist of densely connected convolutional layers in the proposed short and dense connection mechanism, and each convolutional layer is of the same appropriate channels. By doing so, the number of connections and the input channel of each convolutional layer is limited in each SDU.

The advantage of Deep learning approaches relies on their ability to learn features from the data automatically. This makes them faithful to the ground truth classification. However, one of the most significant disadvantages of earlier deep learning-based methods is their strong dependence on a large amount of annotated data ImageNet [65]. Since the availability of large data is not always ensured, particularly in real-time training scenarios, heavily supervised deep learning-based methods would not be applicable for the first and second time seen vehicle V-reID.

## 5.2 Datasets and Evaluation Protocols

For the re-identification problem, we use three publicly available datasets: VehicleID, VeRi-776, and VeRi-Wild. The choice to use these datasets is based on diversity. The scope of V-reID is immense, meaning their many variations, and a system can perform well in one aspect but not in another. VeRi-776 has few identities (776) but a high number of images per identity, plus all images are taken in the same hour of the day with a clear view. VehicleID, on the other hand, has a large number of identities

(26267), images taken at different times with challenging angles, and the number of images per identity is low. Finally, the VeRi-Wild is similar to VehicleID but with a more significant number of images, roughly double that of VehicleID.

### 5.2.1 PKU VehicleID Dataset

VehicleID[74] contains 221,763 images of 26,267 identities from 12 cameras. Images are taken in the morning and afternoon. No further annotations are provided. Figure 5.1 presents a few samples taken from the VehicleID.



Figure 5.1: Samples from the VehicleID dataset.

### 5.2.2 VeRi-776 Dataset

VeRi-776 [79] contains 49,360 images of 776 identities in total from 18 cameras. All images are taken in the afternoon with spatio-temporal relation annotation. Figure 5.2 shows some samples from the VeRi-776 dataset.



Figure 5.2: Samples from the VeRi-776 dataset.

### 5.2.3 VeRi-Wild Dataset

VERI-WILD[81] is the largest public dataset for V-reID with 413,314 images of 0,671 identities across 174 cameras. Figure 5.3 presents some vehicle examples taken from the VeRi-Wild.



**Figure 5.3:** *Samples from the VeRi-Wild dataset.*

Table 5.1 presents a comparison between the VehicleID [74], the VeRi-776 [79] and the VERI-WILD [81] based on different criteria. In this table, one can notice that the VERI-WILD dataset is far more challenging than Veri-776 and VehicleID datasets. VeRI-WILD contains almost ten times the size of Veri-776 images, but also the vehicles are also taken under different acquisition conditions (e.g., occlusions, night view, and weather variations).

Dataset	VehicleID	VeRi-776	VERI-Wild
Images	221,763	49,360	413,314
Identities	26,267	776	40,671
Cameras	12	18	174
Capture time	N/A	18h	125,20h
View	2	6	Unconstrained
Spatio-temporal Relation Annotation	no	yes	yes
Tracks Across cameras	no	no	yes
Camera ID	no	no	yes
Timestamp	no	no	yes
Occlusion	no	no	yes
Complex Background	no	no	yes
Morning	yes	no	yes
Afternoon	yes	yes	yes
Night	no	no	yes
Rainy Weather	no	no	yes
Foggy Weather	no	no	yes

**Table 5.1:** *Comparisons between the VehicleID [74], the VeRi-776[79] and the VERI-WILD[81]*

## 5.3 Material and Methodology

### 5.3.1 Progressive One-Shot/Few-Shots Vehicle Re-Identification

Following the progressive learning paradigm, in this work, we propose a method for gradually labeling vehicle samples starting from one single annotated example per identity. Our method can be partitioned into two modules. First, we use a CNN to learn the classification of one example per identity (the training set corresponds to  $\mathcal{L}$  at the first iteration), and second, the CNN parameters are gradually updated to finally succeed to classify all samples in the unlabeled set  $\mathcal{U}$ .

The first step is performed by back propagating the cross entropy loss function:

$$\min_{\theta, \omega} \sum_{i=1}^{n_l} l_{CE}(f(\omega, \phi(\theta, x_i)), \hat{y}) \quad (5.1)$$

Indeed, the classification would not be efficient due to the reduced size of the training set. To alleviate this issue, we initialize our network using weights from a model pre-trained on the ImageNet [65] classification problem. This initialization may help to have an efficient feature extractor but not an optimized classifier. For this reason, we perform a forward pass to all available data ( $\mathcal{L} \cup \mathcal{U}$ ) using the CNN. For each sample in the data, we extract a feature vector given from the penultimate layer of the model. The CNN acts as a projector that maps sample images to a euclidean feature space. The features in the produced space can be compared in turn using the  $L_2$  distance.

Once the projection has been performed for all data, we calculate the euclidean distance to the initially labeled sample belonging to  $\mathcal{O}$ . We then attribute to each unlabeled sample the label of its closest labeled example. The new label of the unlabeled sample is called pseudo-label. Equation 5.10 is used to assign pseudo-labels.

$$\arg \min_{(x_l, y_l) \in \mathcal{L}} \| \phi(\theta, x_i) - \phi(\theta, x_l) \|, x_i \in \mathcal{U} \quad (5.2)$$

Only a few samples are selected from all pseudo-labeled data to have a confirmed label at each iteration. The subset of size  $m$ , which contains selected pseudo labeled samples  $\mathcal{S}$  is added to the labeled set  $\mathcal{L}$ .  $m$  varies with each iteration and is calculated with respect to a curriculum learning paradigm in the following way:

$$m = (step + 1) * p/100, \quad (5.3)$$

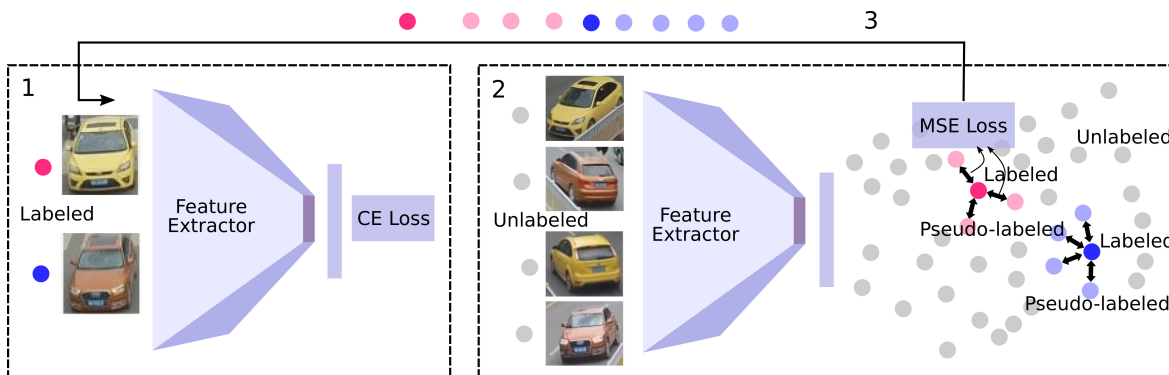
$p$  is the initial enlarging factor empirically initialized to 0.1, and  $step$  is the number of steps. The model at this point is prone to error since false pseudo-labeling can easily occur, which will lead the model to an incorrect optimal solution. To address this problem, we further add to the proposed model an additional regularising aspect named anchoring. The anchoring consists of two folds: (1) computing the MSE loss between pseudo-labeled samples and their closest originally label ones at each selection

iteration. Then (2) backpropagate the MSE loss through the network. The anchoring regularisation, in this way, pulls the features of the pseudo-labeled samples to their corresponding features of labeled samples. This allows the model learning to project the same class samples (the pseudo and the labeled examples) to nearby coordinates in the feature space. In turn, the model succeeds in classifying unlabeled data in future iterations.

Formally, the model is updated over  $n$  iterations to anchor the pseudo-labeled features to the labeled samples by minimizing the mean square error loss given by:

$$\min_{\theta, \omega} l_{MSE}(\phi(\theta, x_s), \phi(\theta, x_l)), x_s \in \mathcal{S} \quad (5.4)$$

Our CNN model starts learning only from one labeled sample per identity. Unlabeled samples are then gradually added to the training data following a curriculum learning paradigm [125]. A mean squared loss is further added to optimize the network and reduce the failure selection of pseudo-labeled samples. Figure 5.4 gives an overview of the proposed method.



**Figure 5.4:** *Method Overview: First, the training stage starts by learning the vehicle identities from labeled data and a cross-entropy loss function (1). Second, unlabeled data are projected to the feature space induced by the CNN penultimate layer. A selected number of unlabeled data close to the originally labeled samples is given pseudo-labels (2). A mean squared loss is further computed between pseudo labeled and the originally labeled data. This loss is backpropagated until narrowing the gap between labeled and pseudo-labeled data (3).*

The anchoring method is effective specifically in the re-identification problem because it regularizes the CNN based on the relation between samples. In standard image classification, for instant, the CNN does not require information other than that of the image itself. However, re-identification is about retrieval and image comparison. Hence, a regulator factor that constantly compares samples can be influential in the



re-identification problem. Furthermore, as opposed to natural objects like humans, vehicles are highly similar if they have the same model, so comparing samples can help the CNN detect very subtle variations between vehicles. Figure 5.5 is an example of different identity vehicles with the same model.



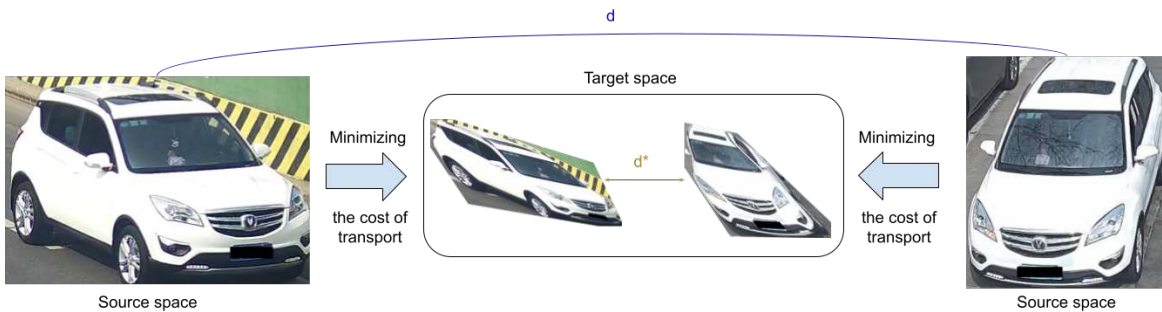
**Figure 5.5:** *Different vehicles with same model.*

### 5.3.2 Cross Camera Vehicle Re-Identification

This section considers the set of vehicle images captured from each camera as a separate dataset. We extract vehicle features from each image using a CNNs [45] model. In order to narrow the gap between feature distributions across different cameras, we use optimal transport as a CNN regulator. In our formulation, we minimize the transfer cost between two feature vectors of the same vehicle from different sources. Figure 5.6 overviews the proposed system. At the training stage, the system takes a couple of images of the same vehicle from different cameras. Features are separately extracted using ResNet50 [45]. The features are then narrowed using optimal transport, then back-propagated using stochastic gradient descent to end fashion. In the V-reID phase, predictions are calculated using the Euclidean distance between features extracted using the training model.







**Figure 5.7:** Transporting source images to a representation that reduces the euclidean distance between their respective features.

images. To establish this representation, we use three components (1) Cross-entropy loss, (2) Euclidean difference between feature vectors (3) Cost of adaptation from the source domain to target domain.

Optimal transport is a theory that allows a comparison between probability distribution in a geometrical sound manner.

The cost  $\gamma$  of optimal transport is defined as follows:

$$\gamma = \arg \min_{\gamma} \langle \gamma, C \rangle_F \quad (5.6)$$

where  $\langle \cdot, \cdot \rangle_F$  is the Frobenius dot product,  $C \geq 0$  is a cost matrix representing the pairwise costs defined as:

$$C = 0.01 \times \|\phi(\theta, x_1^i) - \phi(\theta, x_2^i)\| + (l_{CE}(f(\omega, \phi(\theta, X_i)), y)) \quad (5.7)$$

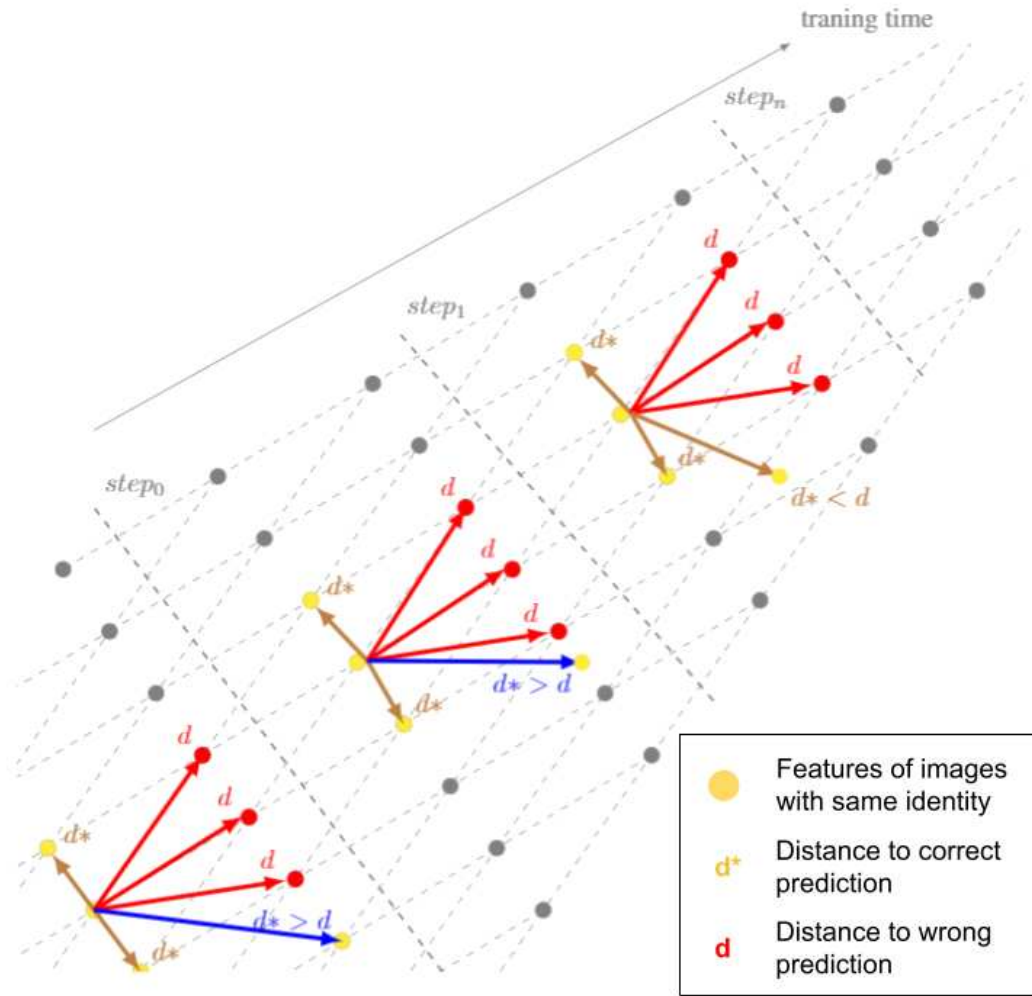
Finally,  $\gamma$  is multiplied by the cross-entropy loss squared to normalize it with the first part and then summed with Euclidean distance between features as shown in equation 5.8. Figure 5.8 shows the contribution of both parts of the loss function.

$$\min_{\theta, \omega, \gamma} \sum_{i=1}^n (\gamma \times (l_{CE}(f(\omega, \phi(\theta, X_i)), y))^2 + \|\phi(\theta, x_1^i) - \phi(\theta, x_2^i)\|) \quad (5.8)$$

The entire loss function proposed is shown in equation 5.9.

$$\min_{\theta, \omega, \gamma} \sum_{i=1}^n (l_{CE}(f(\omega, \phi(\theta, X_i)), y) \times (1/(step + 1) + \gamma \times (l_{CE}(f(\omega, \phi(\theta, X_i)), y) + \|\phi(\theta, x_1^i) - \phi(\theta, x_2^i)\|)) \quad (5.9)$$

The loss is back-propagated using stochastic gradient decent, batch size is 16, learning rate initialized to 0.1. Finally, prediction is calculated as euclidean distance between



**Figure 5.8:** Latent space distribution: each training step is composed of two planes (points alignment) each representing features of images from same camera.  $d^*$  are distances to a correct V-reID while  $d$  are distances to incorrect V-reID. Naturally, V-reID from the same camera (same plane) has  $d^* < d$  however, this is not the case from one plane to another where  $d^* > d$ . The optimal transport aims to project the two planes on a latent space where  $d^* < d$ .

query  $x_q$  and gallery  $x_g$  as shown in the following equation:

$$\arg \min_{x_q \in \mathcal{Q}} || \phi(\theta, x_q) - \phi(\theta, x_g) ||, x_i \in \mathcal{G}. \quad (5.10)$$

Where  $\mathcal{G}$  is gallery set and  $\mathcal{Q}$  the query set.

The need for domain adaptation arises from a drop in performance in the target domain, whether it is due to a lack (or non-existence) of labeled samples. Domain adaptation aims to transfer knowledge from other tasks to build a representation of the target domain. This approach is most called for when the samples to re-identify are radically different due to the difference between position and time of day. Thus, we argue that the re-identification problem can be viewed as a domain adaptation problem because re-identifying a sample represents the similarities between it and other samples; to do so, you need to transfer knowledge from individual representations.

## 5.4 Results and Discussion

### 5.4.1 Progressive One-Shot/Few-Shots Vehicle Re-Identification

#### Evaluation on VehicleID dataset

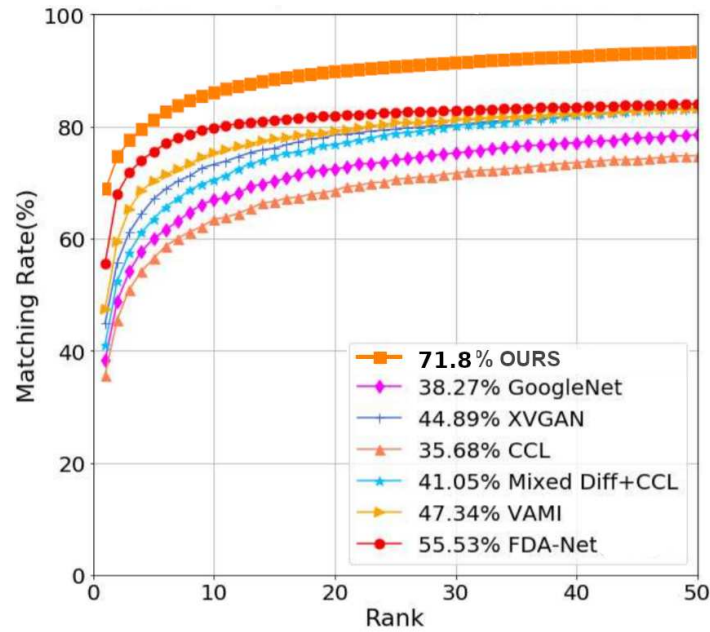
We report in this section the performance of our method using the one-shot and the few-shots learning-based settings. The evaluation has been made on the VehicleID dataset. The training set contains 201,790 images with 23,928 vehicle identities. The query set contains 17378 images, and the proposed approach is evaluated using two gallery sizes composed of 1600 and 2400 images. Table 5.2 shows the obtained results on VehicleID dataset using a gallery size of 1600 samples. Table 5.3 and Figure 5.9 presents the results obtained with a gallery set containing 2400 samples. From these results, one can notice that the proposed method outperforms other state-of-the-art methods by a large margin in both experiment settings and using only 20% of the training set. It is also interesting to notice that using only one label example when training our model, the performance of the proposed method is higher than FDA-net, VAMI, Mixed Diff + CCL, CCL, XVGAN, and GoogleNet. The experiments show that the proposed method is indeed better than other supervised based state-of-the-art method, since using only few training samples, it is able to achieve higher performance.

Gallery size=1600			
Methods	mAP	R=1	R=5
LOMO[71]	-	18.85	29.18
DGD[127]	-	40.25	65.31
GoogleLeNet[129]	42.85	43.0	63.86
FACT[79]	-	44.0	64.75
XVGAN[142]	-	49.55	71.0
CCL[74]	44.8	39.94	62.98
Mixed Diff [74]	48.1	45.05	68.85
HDC [134]	63.1	-	-
VAMI [144]	-	52.87	75.12
FDA [81]	65.33	59.84	77.09
GSTE [4]	74.30	<b>74.8</b>	83.60
Baseline (ResNet-50)[45]	68.48	65.79	76.64
Ours (one example)	33.6	30.0	44.9
Ours (with 20% annotated)	<b>76.4</b>	74.2	<b>85.8</b>

In the next section, we report the performance of the proposed method on the most challenging V-reID dataset (VERI-WILD).

**Table 5.2:** *Performance on VehicleID dataset gallery size = 1600.*

In Table 5.2, it is interesting to note the gap in performance between one-shot, the 20% annotated, and the fully-supervised setup. Suggesting two conclusions (1) a minimum number of samples is required to achieve a good representation. (2) At a certain point, increasing the number of samples does not improve performance as in the case of fully supervised, but the learning paradigm employed, namely progressive learning, boosts the performance considerably, as shown in the 20% annotated setup.



**Figure 5.9:** The CMC comparisons on VehicleID-2400 test set.

Figure 5.9 highlights an important aspect of V-reID: the matching rate of a vehicle in an interval of ranking, i.e., a vehicle is considered correctly identified if its ranking falls in a predefined interval. This curve describes the system's response to complex samples, meaning the correct prediction might not be in the top 5 rankings, but it's not entirely wrong. The curve shows the impact of the anchoring technique as the rank increases, the matching rate increases quite fast compared to other approaches.

Gallery Size=2400			
Methods	mAP	R=1	R=5
LOMO[71]	-	15.32	25.29
DGD[127]	-	37.33	57.82
GoogleLeNet[129]	40.39	38.27	59.0
FACT[79]	-	39.92	60.32
XVGAN[142]	-	44.89	66.65
CCL[74]	38.6	35.68	56.24
Mixed Diff [74]	45.5	41.05	63.38
HDC [134]	57.5	-	-
VAMI [144]	-	47.34	70.29
FDA [81]	61.84	55.53	74.65
GSTE [4]	72.40	<b>74.00</b>	82.70
baseline (ResNet-50)[45]	66.19	63.45	74.70
Ours (one example)	35.7	51.5	51.5
Ours (with 20% annotated)	<b>74.5</b>	71.8	<b>83.8</b>

**Table 5.3:** Performance on VehicleID gallery size = 2400 dataset.

### Evaluation on VERI-Wild dataset

The VeRi-Wild [81] dataset is composed of three test protocols with increasing difficulty depending on the size of the gallery. Tables 5.4, 5.5 and 5.6 shows the results of our approach on the small, medium and large galleries respectively. VeRi-Wild [81] is by far the most challenging V-reID dataset, for instance, the FDA [81] approach achieved mAP of 61.84 on VehicleID dataset however drops to 35.11 on VeRi-Wild. On the other hand, our approach achieves 74.5 on VehicleID and only drops to 54.5 on VeRi-Wild, which confirms the robustness of our approach in difficult re-identification scenarios. For the three test protocols, the proposed approach outperforms the state-of-the-art supervised methods by a large margin.

Small gallery			
Methods	mAP	R=1	R=5
GoogleLeNet[129]	24.27	57.16	75.13
Triplet[109]	15.69	44.67	63.33
FACT[79]	26.41	53.4	75.03
CCL[74]	22.50	56.96	75.0
HDC [134]	29.14	57.1	78.93
GSTE [4]	31.42	60.46	80.13
Unlabeled GAN [146]	29.86	58.06	79.6
FDA [81]	35.11	64.03	82.8
Ours (with 20% annotated)	<b>54.5</b>	<b>77.2</b>	<b>89.1</b>

**Table 5.4:** Performance on VeRi-Wild gallery size = 3000 (small gallery)

Medium gallery			
Methods	mAP	R=1	R=5
GoogleLeNet[129]	24.15	53.16	71.1
Triplet[109]	13.34	40.34	58.98
FACT[79]	22.66	46.16	69.88
CCL[74]	19.28	51.92	70.98
HDC [134]	24.76	49.64	72.28
GSTE [4]	26.18	52.12	74.92
Unlabeled GAN [146]	24.71	51.58	74.42
FDA [81]	29.0	57.82	78.34
Ours (with 20% annotated)	<b>48.1</b>	<b>71.7</b>	<b>86.0</b>

**Table 5.5:** Performance on VeRi-Wild gallery size = 5000 (medium gallery)

Large gallery			
Methods	mAP	R=1	R=5
GoogleLeNet[129]	21.53	44.61	43.55
Triplet[109]	9.93	33.46	51.36
FACT[79]	17.62	37.94	59.89
CCL[74]	14.81	44.6	60.0
HDC [134]	18.30	43.97	64.89
GSTE [4]	19.50	45.36	66.5
Unlabeled GAN [146]	18.23	43.63	65.52
FDA [81]	22.78	49.43	70.48
Ours (with 20% annotated)	<b>39.5</b>	<b>64.2</b>	<b>79.7</b>

**Table 5.6:** Performance on VeRi-Wild gallery size = 10000 (large gallery).

### Evaluation on VeRi776 dataset

Table 5.7 presents the mAP, the rank-1 and the rank-5 of the proposed method with comparison to the state-of-the-art methods. The cumulative matching characteristic curves are plotted in Figure 5.10. According to Table 5.7, one can notice that using only 20% of training data; our progressive V-reID approach ranked second after the FDA-Net [81] approach with slightly lower performance, less than 0.2%. Meanwhile, the proposed approach outperforms all other state-of-the-art approaches in terms of rank-1 score. Using only one example per identity in the training set, our method performs better than several state-of-the-art approaches, including LOMO, DGD, and GoogleLeNet methods. Figure 5.10 confirms the superiority of the proposed method compared to the other state-of-the-art methods. As one can notice, the CMC curve of our method is above the other curve. A slight intersection with the FAD-Net CMC curve can be observed on the rank-5 (x-axis), but the overall superiority of the proposed method can be easily distinguished.



Methods	mAP	rank-1	rank-5
LOMO[71]	9.64	25.33	46.48
DGD[127]	17.92	50.0	67.52
GoogleLeNet[129]	17.81	52.12	66.79
FACT[79]	18.73	51.85	67.16
XVGAN[142]	24.65	60.20	77.03
OIFE[123]	48.00	65.92	87.66
Siamese Visual[112]	29.48	41.12	60.31
FACT+Plate+STR[79]	27.77	61.44	78.78
VAMI[144]	50.13	77.03	90.82
VAMI + ST[144]	61.32	85.92	91.84
Path-LSTM[144]	58.27	83.49	90.04
FDA-Net[81]	55.49	84.27	92.43
baseline (ResNet-50)[45]	51.58	86.71	92.43
Ours (one example)	18.5	57.2	67.4
Ours (with 20% annotated)	55.3	86.2	92.0

**Table 5.7:** Performance on VeRi-776 dataset

Compared to VehicleID and VeRiWild datasets, VeRi-776 contains less challenging samples; all instances share the same lighting conditions. Since our approach starts from easy samples and progressively annotates the more challenging samples, it is ineffective in this particular dataset. Figure 5.10 shows a CMC comparison between our approach and state-of-the-art approaches. The FDA-Net approach for example, almost overlaps with ours. However, on the VehicleID and VeRiWild datasets, the difference is quite significant. This confirms the effectiveness of our approach on complex samples thanks to its progressive learning.

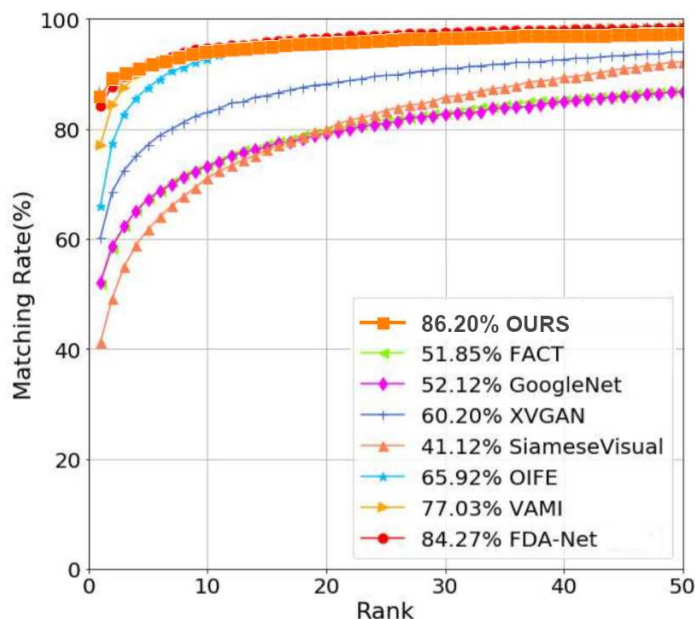


Figure 5.10: The CMC comparisons on VeRi-776 test set.

## 5.4.2 Cross Camera Vehicle Re-Identification

### Ablation study

We start by an ablation study to demonstrate the effectiveness of the proposed loss function. Tables 5.8, 5.9 and 5.10 show the results of the model using the cross-entropy loss alone and the proposed loss with optimal transport on VeRi-776 and VehicelID respectively as well both losses with a regulation factor. We notice that when only cross-entropy is considered, the performance drops in both datasets. This behavior confirms that the integration of the optimal transport loss perfectly aligns the source domain samples and target domain samples, narrowing the gap between vehicle representation computed from different cameras. We also notice that reducing the effect of the cross-entropy loss when the training progresses slightly improves the obtained performance. Figure 5.8 demonstrates the relation between both losses and the effect of reducing the cross-entropy loss. Points in figure 5.8 represent features in the latent space; the diagonally aligned points are features of images from the same camera. Yellow points have the same identity, so the cross-entropy loss reduces the gap of yellow points along the diagonal line while optimal transport reduces the distance across diagonal lines. At a certain point, the same camera features become too close where optimization is no longer required; hence, we utilize a regulating factor.

Table 5.8 show a noticeable increase in performance when both losses are used, but this increase is larger for the second set in Table 5.9 and 5.10. This is mostly due to same identity images being very different from each other allowing the second loss

Methods	mAP	rank-1	rank-5
Cross Entropy	51.58	86.71	92.43
Cross Entropy + OT	65.6	91.5	96.3
Cross Entropy + OT + regulation	<b>66.2</b>	<b>92.8</b>	<b>97.0</b>

**Table 5.8:** Ablation Study on VeRi-776 dataset

(OT) to be more effective. The gallery size and difficulty in the test set of Table 5.10

Methods	mAP	R=1	R=5
Cross Entropy	68.48	65.79	76.64
Cross Entropy + OT	81.4	78.2	92.6
Cross Entropy + OT + regulation	<b>81.7</b>	<b>78.6</b>	<b>93.1</b>

**Table 5.9:** Ablation Study on VehicleID dataset gallery size = 1600.

is higher than that of Table 5.9. We see that OT part of the loss function is more effective since the set has more difficult samples.

Methods	mAP	R=1	R=5
Cross Entropy	66.19	63.45	74.70
Cross Entropy + OT	80.2	77.7	91.3
Cross Entropy + OT + regulation	<b>80.8</b>	<b>78.2</b>	<b>92.4</b>

**Table 5.10:** Ablation Study on VehicleID dataset gallery size = 2400.

**Evaluation on VeRi-776**

Methods	mAP	rank-1	rank-5
LOMO[71]	9.64	25.33	46.48
DGD[127]	17.92	50.0	67.52
GoogleLeNet[129]	17.81	52.12	66.79
FACT[79]	18.73	51.85	67.16
XVGAN[142]	24.65	60.20	77.03
OIFE[123]	48.00	65.92	87.66
Siamese Visual[112]	29.48	41.12	60.31
FACT+Plate+STR[79]	27.77	61.44	78.78
VAMI[144]	50.13	77.03	90.82
VAMI + ST[144]	61.32	85.92	91.84
Path-LSTM[144]	58.27	83.49	90.04
FDA-Net[81]	55.49	84.27	92.43
Ours	<b>66.2</b>	<b>92.8</b>	<b>97.0</b>

**Table 5.11:** *Performance on VeRi-776 dataset*

Table 5.11 shows the performance of our method on VERI-766 dataset. The proposed method surpass state-of-the-art by 4.88% on mAP, 6.88% on rank-1 and 4.57% on rank-5. We see higher improvement on mAP then the rank-1 and rank-5, this is explainable by the fact that the higher a true prediction is ranked and so the value of mAP is higher and since our system is more efficient on difficult samples so they rank higher.

**Evaluation on VehicleID**

The VehicleID dataset is larger and more challenging than the VeRi-776 dataset where we show the system’s ability to detect subtle similarities. What is particular in this dataset is that, in the hard samples, images of the same identity looks very different, so representing each of them separately, i.e, the first part of the loss, will produce features that are distant from each other. However, the second part of the loss function will only represent the subtle indicators of the similarities between the two images. The second gallery is larger (2400 samples) and composed of harder samples, we see a very small drop in performance, yet it remains higher than other approaches.

Gallery Size=1600			
Methods	mAP	R=1	R=5
LOMO[71]	-	18.85	29.18
DGD[127]	-	40.25	65.31
GoogleLeNet[129]	42.85	43.0	63.86
FACT[79]	-	44.0	64.75
XVGAN[142]	-	49.55	71.0
CCL[74]	44.8	39.94	62.98
Mixed Diff [74]	48.1	45.05	68.85
HDC [134]	63.1	-	-
VAMI [144]	-	52.87	75.12
FDA [81]	65.33	59.84	77.09
GSTE [4]	74.30	74.8	83.60
Ours	<b>81.7</b>	<b>78.6</b>	<b>93.1</b>

**Table 5.12:** Performance on VehicleID dataset gallery size = 1600.

Gallery Size=2400			
Methods	mAP	R=1	R=5
LOMO[71]	-	15.32	25.29
DGD[127]	-	37.33	57.82
GoogleLeNet[129]	40.39	38.27	59.0
FACT[79]	-	39.92	60.32
XVGAN[142]	-	44.89	66.65
CCL[74]	38.6	35.68	56.24
Mixed Diff [74]	45.5	41.05	63.38
HDC [134]	57.5	-	-
VAMI [144]	-	47.34	70.29
FDA [81]	61.84	55.53	74.65
GSTE [4]	72.40	74.00	82.70
Ours	<b>80.8</b>	<b>78.2</b>	<b>92.4</b>

**Table 5.13:** Performance on VehicleID gallery size = 2400

# Chapter 6

## Conclusions and Future Work

### 6.1 Conclusions

We first presented a robust real-time ALPR system using a pipeline with two deep learning stages. The LP detection stage is based on the state-of-the-art YOLOv2 object detection CNNs. For the second stage, we compare two recognition engines: a sequence labeling method that recognizes the whole license plate without character-level segmentation and a joint detection/recognition approach that performs the recognition on the plate component level. The proposed system is robust to illumination and weather conditions and can achieve a total LP recognition rate of 97.67% in the GAP-LP dataset and 91.46% in the Radar dataset with a reasonable computational time. We also introduced a new public dataset for multi-norm and multilingual ALPR that includes 9,175 fully annotated images. Compared to the existing datasets for this task, GAP-LP is the largest ALPR dataset making it suitable for trying and evaluating deep learning techniques. In order to reduce the time and cost of annotation processing, we have proposed a new semi-automatic annotation procedure of LP images with labeled component bounding boxes. Future work consists of integrating vehicle make and model recognition to improve the vehicle identity recognition process and help check correlation with data stored on police and homeland security databases.

Secondly, we have proposed a multi-Stream deep network for Vehicle Make and Model Recognition. The proposed approach combines global representation with local representations using a dynamic fully-connected layer; the multi-stream architecture allows the system to use specialized feature extractors to detect subtle inter-class variations. Our approach achieves a state of the art results while being robust. Through the multi-stream architecture, our experiments show that finding the best combination of local features and global representation for input can significantly improve performance. Third, we propose a semi-supervised re-identification system that starts from a small set of annotated samples and progressively label and learn more challenging examples. We proposed a novel anchoring method to ensure our system's capability to learn new features without losing previous knowledge. We show the impact of the

progressive nature of our system by comparing it with the same feature extractor on a fully annotated training set. Our system achieves state-of-the-art results on the most recent databases, especially for the most challenging ones. Finally, considering the V-reID task as a domain adaptation problem allowed our system to find similar vehicles showed across different cameras. This performance is interesting and can be extended to many applications such as V-reID from video captured using drones and street cameras.

## 6.2 Future Work

For years, aspects of deep learning been a complete mystery, and the research community focus more on what deep learning can do rather than on understanding why it behaves in a certain way, for instance, why do extensive neural networks work better than smaller ones and How do they generalize with so many parameters. However, very recently, some works provide exciting insight that will enable future research to discover the true potential of deep neural networks.

### 6.2.1 The Lottery Ticket Hypothesis

Although The Lottery Ticket Hypothesis proposed by Frankle and Carbin [28] is only tested on small datasets like MNIST [67], it delivers very impressive optimization, up to 90% reduction of parameter count. Which indicates a keen insight into the inner workings of neural networks. The hypothesis considers neural networks as giant lotteries; through random initialization, particular sub-networks are mathematically lucky and are recognized for their potential by the optimizer while the rest of the network does not contribute much. I want to work on similar optimization techniques that leverage an excellent understanding of neural networks for more advanced tasks.

### 6.2.2 Zero-Shot Learning

Zero-Shot Learning is a fascinating concept for me. Zero-Shot is when the model can classify unseen classes without any training examples. This might seem far-fetched, but the human brain is capable of doing just that. I presented few-shot learning in this work where every identity in the re-identification problem has one labeled image. Though deep learning techniques were able in the same cases to surpass even humans on some visual tasks, they always relied on a massive amount of data. I want to extend that to zero-shot learning.

### 6.2.3 Meta-Learning

In this work, I visited some learning paradigms. Although I did not work with Meta-Learning (learning to learn), it is fascinating. Meta-learning is the science of observing

how different learners' approaches perform on a range of learning tasks and then learning from this experience to learn new tasks much faster than otherwise possible. It allows us to reflect and inspire from our own experiences as learners.

#### **6.2.4 Self-Driving cars**

Self-Driving cars is an application of deep learning that I am particularly interested in pursuing. It requires both speeds in reacting to change in visual data as well as precision. The environment can change radically and very fast. Furthermore, Once a vehicle is on the road, it will detect objects it has not come across in its training to which it needs to react. This calls for very effective use of machine learning paradigms. This application can push deep-learning methods to their limits. Hence, it allows researchers to mine valuable data and gain insights into the inner workings of deep learning. More particularly, the zero-shot setting discussed above.



# Bibliography

- [1] A large-scale car dataset for fine-grained categorization and verification. 2015. arXiv:1506.08959.
- [2] M. A. Abdelwahab. Accurate vehicle counting approach based on deep neural networks. In *2019 International Conference on Innovative Trends in Computer Engineering (ITCE)*, pages 1–5, 2019. doi: 10.1109/ITCE.2019.8646549.
- [3] C. N. E. Anagnostopoulos, I. E. Anagnostopoulos, V. Loumos, and E. Kayafas. A license plate-recognition algorithm for intelligent transportation system applications. *IEEE Transactions on Intelligent Transportation Systems*, 7(3):377–392, September 2006.
- [4] Y. Bai, Y. Lou, F. Gao, S. Wang, Y. Wu, and L. Duan. Group-sensitive triplet embedding for vehicle reidentification. *IEEE Transactions on Multimedia*, 20(9): 2385–2399, 2018.
- [5] Erhan Bas, A. Tekalp, and F. Salman. Automatic vehicle counting from video for traffic flow analysis. pages 392–397, 07 2007. ISBN 1-4244-1068-1. doi: 10.1109/IVS.2007.4290146.
- [6] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Comput. Vis. Image Underst.*, 110(3):346–359, June 2008. ISSN 1077-3142. doi: 10.1016/j.cviu.2007.09.014. URL <http://dx.doi.org/10.1016/j.cviu.2007.09.014>.
- [7] Amine ben khalifa and Hichem Frigui. A dataset for vehicle make and model recognition. 06 2015.
- [8] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning.
- [9] M. Biglari, A. Soleimani, and H. Hassanpour. A cascaded part-based system for fine-grained vehicle classification. *IEEE Transactions on Intelligent Transportation Systems*, 19(1):273–283, January 2018. ISSN 1524-9050. doi: 10.1109/TITS.2017.2749961.

- [10] Tomas Björklund, Attilio Fiandrotti, Mauro Annarumma, Gianluca Francini, and Enrico Magli. Robust license plate recognition using neural networks trained on synthetic images. *Pattern Recognition*, 93:134–146, 2019. ISSN 0031-3203.
- [11] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.
- [12] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698, 1986. doi: 10.1109/TPAMI.1986.4767851.
- [13] Shyang-Lih Chang, Li-Shien Chen, Yun-Chung Chung, and Sei-Wan Chen. Automatic license plate recognition. *IEEE Transactions on Intelligent Transportation Systems*, 5(1):42–53, March 2004.
- [14] Teik Koon Cheang, Yong Shean Chong, and Yong Haur Tay. Segmentation-free vehicle license plate recognition using convnet-rnn. *CoRR*, abs/1701.06439, 2017.
- [15] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations, 2020.
- [16] Xiaoxue Chen, Lianwen Jin, Yuanzhi Zhu, Canjie Luo, and Tianwei Wang. Text recognition in the wild: A survey, 2020.
- [17] Zhanzhan Cheng, Fan Bai, Yunlu Xu, Gang Zheng, Shiliang Pu, and Shuigeng Zhou. Focusing attention: Towards accurate text recognition in natural images. pages 5086–5094, 10 2017. doi: 10.1109/ICCV.2017.543.
- [18] M. Cheon, W. Lee, C. Yoon, and M. Park. Vision-based vehicle detection system with consideration of the detecting location. *IEEE Transactions on Intelligent Transportation Systems*, 13(3):1243–1252, 2012.
- [19] H. Cho, M. Sung, and B. Jun. Canny text detector: Fast and robust scene text localization algorithm. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3566–3573, 2016. doi: 10.1109/CVPR.2016.388.
- [20] M Cormier, LW Sommer, and M Teutsch. Low resolution vehicle re-identification based on appearance features for wide area motion imagery. in 2016 ieee winter applications of computer vision workshops (wacvw). *IEEE*, 1:3, 2016.
- [21] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.
- [22] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893 vol. 1, 2005.

- [23] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1, pages 886–893. IEEE, 2005.
- [24] S. Du, M. Ibrahim, M. Shehata, and W. Badawy. Automatic license plate recognition (alpr): A state-of-the-art review. *IEEE Transactions on Circuits and Systems for Video Technology*, 23(2):311–325, February 2013. ISSN 1051-8215. doi: 10.1109/TCSVT.2012.2203741.
- [25] B. Epshtein, E. Ofek, and Y. Wexler. Detecting text in natural scenes with stroke width transform. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2963–2970, 2010. doi: 10.1109/CVPR.2010.5540041.
- [26] Mark Everingham, Luc Van Gool, Christopher Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88:303–338, 06 2010. doi: 10.1007/s11263-009-0275-4.
- [27] K. Nishiyama F. Tafazzoli and H. Frigui. A large and diverse dataset for improved vehicle make and model recognition. 2017. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops 2017*.
- [28] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks, 2019.
- [29] William J. Frawley, Gregory Piatetsky-Shapiro, and Christopher J. Matheus. *AI Magazine*, 13(3):57, September 1992. doi: 10.1609/aimag.v13i3.1011. URL <https://ojs.aaai.org/index.php/aimagazine/article/view/1011>.
- [30] Richard G. Casey and Eric Lecolinet. A survey of methods and strategies in character segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 18:690–706, 08 1996.
- [31] Wouter Van Gansbeke, Simon Vandenhende, Stamatios Georgoulis, Marc Proesmans, and Luc Van Gool. Scan: Learning to classify images without labels, 2020.
- [32] Sina Ghassemi, A Fiandrotti, Emanuele Caimotti, Gianluca Francini, and Enrico Magli. Vehicle joint make and model recognition with multiscale attention windows. *Signal Processing: Image Communication*, 72, 12 2018. doi: 10.1016/j.image.2018.12.009.
- [33] Ioannis Giannoukos, Christos-Nikolaos Anagnostopoulos, Vassili Loumos, and Eleftherios Kayafas. Operator context scanning to support high segmentation rates for real time license plate recognition. *Pattern Recognition*, 43(11):3866–3878, 2010.

- [34] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations, 2018.
- [35] R. Girshick. Fast r-cnn. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1440–1448, 2015.
- [36] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 580–587, 2014.
- [37] Shaogang Gong, Marco Cristani, Shuicheng Yan, and Chen Change Loy. *Person Re-Identification*. 10 2013.
- [38] Ian J. Goodfellow, Yaroslav Bulatov, Julian Ibarz, Sacha Arnoud, and Vinay Shet. Multi-digit number recognition from street view imagery using deep convolutional neural networks, 2014.
- [39] Alex Graves, Abdel rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks, 2013.
- [40] Derek Greene, Pádraig Cunningham, and Rudolf Mayer. *Unsupervised Learning and Clustering*, pages 51–90. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.
- [41] J. M. Guo and Y. F. Liu. License plate localization and character segmentation with feedback self-learning and hybrid binarization techniques. *IEEE Transactions on Vehicular Technology*, 57(3):1417–1424, May 2008.
- [42] Guy Hacohen and Daphna Weinshall. On the power of curriculum learning in training deep networks, 2019.
- [43] D. He, X. Yang, C. Liang, Z. Zhou, A. G. Ororbia, D. Kifer, and C. L. Giles. Multi-scale fcn with cascaded instance aware segmentation for arbitrary oriented word spotting in the wild. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 474–483, 2017. doi: 10.1109/CVPR.2017.58.
- [44] H. He, Z. Shao, and J. Tan. Recognition of car makes and models from a single traffic-camera image. *IEEE Transactions on Intelligent Transportation Systems*, 16(6):3182–3192, December 2015. ISSN 1524-9050. doi: 10.1109/TITS.2015.2437998.
- [45] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. URL <http://arxiv.org/abs/1512.03385>.
- [46] Pan He, Weilin Huang, Tong He, Qile Zhu, Yu Qiao, and Xiaolin Li. Single shot text detector with regional attention, 2017.

- [47] Tong He, Zhi Tian, Weilin Huang, Chunhua Shen, Yu Qiao, and Changming Sun. An end-to-end textspotter with explicit alignment and attention, 2018.
- [48] G. E. Hinton, S. Osindero, and Y. W. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18:1527–1554, 2006.
- [49] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997. doi: 10.1162/neco.1997.9.8.1735.
- [50] J. Hsieh, L. Chen, and D. Chen. Symmetrical surf and its applications to vehicle detection and vehicle make and model recognition. *IEEE Transactions on Intelligent Transportation Systems*, 15(1):6–20, February 2014. ISSN 1524-9050. doi: 10.1109/TITS.2013.2294646.
- [51] G. S. Hsu, J. C. Chen, and Y. Z. Chung. Application-oriented license plate recognition. *IEEE Transactions on Vehicular Technology*, 62(2):552–561, February 2013.
- [52] Q. Hu, H. Wang, T. Li, and C. Shen. Deep cnns with spatially weighted pooling for fine-grained car recognition. *IEEE Transactions on Intelligent Transportation Systems*, 18(11):3147–3156, November 2017. ISSN 1524-9050. doi: 10.1109/TITS.2017.2679114.
- [53] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015. URL <http://arxiv.org/abs/1502.03167>.
- [54] Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman. Deep features for text spotting. 09 2014. ISBN 978-3-319-10592-5. doi: 10.1007/978-3-319-10593-2-34.
- [55] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks. *CoRR*, abs/1506.02025, 2015. URL <http://arxiv.org/abs/1506.02025>.
- [56] Vishal Jain, Zitha Sasindran, Anoop Rajagopal, Soma Biswas, Harish S Bharadwaj, and K R Ramakrishnan. Deep automatic license plate recognition system. In *Proceedings of the Tenth Indian Conference on Computer Vision, Graphics and Image Processing, ICVGIP '16*, pages 6:1–6:8, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4753-2.
- [57] Norazira A. Jalil, A. S. H. Basari, Sazilah Salam, Nuzulha Khilwani Ibrahim, and Mohd Adili Norasikin. The utilization of template matching method for license plate recognition: A case study in malaysia. In Hamzah Asyrani Sulaiman, Mohd Azlishah Othman, Mohd Fairuz Iskandar Othman, Yahaya Abd Rahim, and Naim Che Pee, editors, *Advanced Computer and Communication Engineering Technology*, pages 1081–1090, Cham, 2015. Springer International Publishing.

- [58] Rodolphe Jenatton, Julien Mairal, Guillaume Obozinski, and Francis Bach. Proximal methods for hierarchical sparse coding, 2011.
- [59] Xu Ji, João F. Henriques, and Andrea Vedaldi. Invariant information clustering for unsupervised image classification and segmentation, 2019.
- [60] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding, 2014.
- [61] Jianbin Jiao, Qixiang Ye, and Qingming Huang. A configurable method for multi-style license plate recognition. *Pattern Recognition*, 42(3):358–369, 2009.
- [62] Kai Wang, B. Babenko, and S. Belongie. End-to-end scene text recognition. In *2011 International Conference on Computer Vision*, pages 1457–1464, 2011. doi: 10.1109/ICCV.2011.6126402.
- [63] MI Khalil. Car plate recognition using the template matching method. *International Journal of Computer Theory and Engineering*, 2(5):683, 2010.
- [64] W. Kim and C. Kim. A new approach for overlay text detection and extraction from complex video scene. *IEEE Transactions on Image Processing*, 18(2):401–411, 2009. doi: 10.1109/TIP.2008.2008225.
- [65] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.
- [66] John Lafferty, Andrew Mccallum, and Fernando Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. pages 282–289, 01 2001.
- [67] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010. URL <http://yann.lecun.com/exdb/mnist/>.
- [68] S. Lee, M. S. Cho, K. Jung, and J. H. Kim. Scene text extraction with edge constraint and text collinearity. In *2010 20th International Conference on Pattern Recognition*, pages 3983–3986, 2010. doi: 10.1109/ICPR.2010.969.
- [69] Hui Li, Peng Wang, Mingyu You, and Chunhua Shen. Reading car license plates using deep neural networks. *Image and Vision Computing*, 72:14–23, 2018.
- [70] Minghui Liao, Jian Zhang, Zhaoyi Wan, Fengming Xie, Jiajun Liang, Pengyuan Lyu, Cong Yao, and Xiang Bai. Scene text recognition from two-dimensional perspective, 2018.

- [71] S. Liao, Y. Hu, Xiangyu Zhu, and S. Z. Li. Person re-identification by local maximal occurrence representation and metric learning. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2197–2206, June 2015. doi: 10.1109/CVPR.2015.7298832.
- [72] R. Lienhart and A. Wernicke. Localizing and segmenting text in images and videos. *IEEE Transactions on Circuits and Systems for Video Technology*, 12(4): 256–268, 2002. doi: 10.1109/76.999203.
- [73] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2015.
- [74] H. Liu, Y. Tian, Y. Wang, L. Pang, and T. Huang. Deep relative distance learning: Tell the difference between similar vehicles. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2167–2175, June 2016. doi: 10.1109/CVPR.2016.238.
- [75] Li Liu, Yi Zhou, and Ling Shao. Dap3d-net: Where, what and how actions occur in videos?, 2016.
- [76] Li Liu, Yi Zhou, and Ling Shao. Deep action parsing in videos with large-scale synthesized data. *IEEE Transactions on Image Processing*, PP:1–1, 03 2018. doi: 10.1109/TIP.2018.2813530.
- [77] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott E. Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: single shot multibox detector. *CoRR*, abs/1512.02325, 2015.
- [78] Xinchun Liu, Wu Liu, Huadong Ma, and Huiyuan Fu. Large-scale vehicle re-identification in urban surveillance videos. In *2016 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6. IEEE, 2016.
- [79] Xinchun Liu, Wu Liu, Tao Mei, and Huadong Ma. A deep learning-based approach to progressive vehicle re-identification for urban surveillance. volume 9906, pages 869–884, 10 2016. ISBN 978-3-319-46474-9.
- [80] Yuliang Liu and Lianwen Jin. Deep matching prior network: Toward tighter multi-oriented text detection, 2017.
- [81] Yihang Lou, Yan Bai, Jun Liu, Shiqi Wang, and Lingyu Duan. Veri-wild: A large dataset and a new method for vehicle re-identification in the wild. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

- [82] D. G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, pages 1150–1157 vol.2, September 1999. doi: 10.1109/ICCV.1999.790410.
- [83] D. G. Lowe. Local feature view clustering for 3d object recognition. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, volume 1, pages I–I, December 2001. doi: 10.1109/CVPR.2001.990541.
- [84] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, November 2004. ISSN 0920-5691. doi: 10.1023/B:VISI.0000029664.99615.94. URL <https://doi.org/10.1023/B:VISI.0000029664.99615.94>.
- [85] Canjie Luo, Lianwen Jin, and Zenghui Sun. A multi-object rectified attention network for scene text recognition, 2019.
- [86] Syed Zain Masood, Guang Shu, Afshin Dehghan, and Enrique G. Ortiz. License plate detection and recognition using deeply learned convolutional neural networks. *CoRR*, abs/1703.07330, 2017.
- [87] Minhua Li and Chunheng Wang. An adaptive text detection approach in images and video frames. In *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, pages 72–77, 2008. doi: 10.1109/IJCNN.2008.4633769.
- [88] Hamid Hassanpour Mohsen Biglari, Ali Soleimani. Fined-grained vehicle classification using similar auto extracted parts. 4(1):29–39, 2017. ISSN 2383-1197. URL <http://jmvip.sinaweb.net/article-39605.html>. Journal of Machine Vision and Image Processing.
- [89] Ali Mosleh, Nizar Bouguila, and A. Hamza. Image text detection using a bandlet-based edge detector and stroke width transform. 01 2012. doi: 10.5244/C.26.63.
- [90] Amira Naimi, Yousri Kessentini, and Mohamed Hammami. Multi-nation and multi-norm license plates detection in real traffic surveillance environment using deep learning. In *Proceedings of the 23rd International Conference on Neural Information Processing - Volume 9948*, pages 462–469, New York, NY, USA, 2016. Springer-Verlag New York, Inc. ISBN 978-3-319-46671-2.
- [91] Lukas Neumann and Jiri Matas. A method for text localization and recognition in real-world images. volume 6494, pages 770–783, 11 2010. ISBN 978-3-642-19317-0. doi: 10.1007/978-3-642-19318-7-60.
- [92] Hooi Sin Ng, Yong Haur Tay, Kim Meng Liang, Hamam Mokayed, and Hock Woon Hon. Detection and recognition of malaysian special license plate based on SIFT features. *CoRR*, abs/1504.06921, 2015.



- [93] C. D. Nguyen, M. Ardabilian, and L. Chen. Robust car license plate localization using a novel texture descriptor. In *2009 Sixth IEEE International Conference on Advanced Video and Signal Based Surveillance*, pages 523–528, September 2009.
- [94] Shiguelo Nomura, Keiji Yamanaka, Osamu Katai, Hiroshi Kawakami, and Takayuki Shiose. A novel adaptive morphological approach for degraded character image segmentation. *Pattern Recognition*, 38:1961–1975, 11 2005. doi: 10.1016/j.patcog.2005.01.026.
- [95] Jacek Oskarbski, Marcin Zawisza, and Karol Żarski. Automatic incident detection at intersections with use of telematics. *Transportation Research Procedia*, 14:3466–3475, 12 2016. doi: 10.1016/j.trpro.2016.05.309.
- [96] M. Pancharatnam and Upul Sonnadara. Vehicle counting and classification from a traffic scene. 09 2008.
- [97] V. M. Patel, R. Gopalan, R. Li, and R. Chellappa. Visual domain adaptation: A survey of recent advances. *IEEE Signal Processing Magazine*, 32(3):53–69, 2015. doi: 10.1109/MSP.2014.2347059.
- [98] Van Huy Pham, Phong Quang Dinh, and Van Huan Nguyen. Cnn-based character recognition for license plate recognition system. In Ngoc Thanh Nguyen, Duong Hung Hoang, Tzung-Pei Hong, Hoang Pham, and Bogdan Trawiński, editors, *Intelligent Information and Database Systems*, pages 594–603, Cham, 2018. Springer International Publishing.
- [99] Apostolos Psyllos, Christos-Nikolaos Anagnostopoulos, and E Kayafas. Sift-based measurements for vehicle model recognition. 1, 01 2009.
- [100] S. Qiao, Y. Zhu, X. Li, T. Liu, and B. Zhang. Research of improving the accuracy of license plate character segmentation. In *2010 Fifth International Conference on Frontier of Computer Science and Technology*, pages 489–493, August 2010.
- [101] Qixiang Ye, Wen Gao, Weiqiang Wang, and Wei Zeng. A robust text detection algorithm in images and video frames. In *Fourth International Conference on Information, Communications and Signal Processing, 2003 and the Fourth Pacific Rim Conference on Multimedia. Proceedings of the 2003 Joint*, volume 2, pages 802–806 vol.2, 2003. doi: 10.1109/ICICS.2003.1292567.
- [102] Syafeeza Ahmad Radzi and Mohamed Khalil-Hani. Character recognition of license plate number using convolutional neural network. In Halimah Badioze Zaman, Peter Robinson, Maria Petrou, Patrick Olivier, Timothy K. Shih, Sergio Velastin, and Ingela Nyström, editors, *Visual Informatics: Sustaining Research and Innovations*, pages 45–55, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

- [103] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, June 2016.
- [104] Joseph Redmon and Ali Farhadi. YOLO9000: better, faster, stronger. *CoRR*, abs/1612.08242, 2016.
- [105] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv*, 2018.
- [106] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149, 2017.
- [107] D. Rumelhart, Geoffrey E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.
- [108] Lars Schmarje, Johannes Brünger, Monty Santarossa, Simon-Martin Schröder, Rainer Kiko, and Reinhard Koch. Beyond cats and dogs: Semi-supervised classification of fuzzy labels with overclustering, 2020.
- [109] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. pages 815–823, 06 2015. doi: 10.1109/CVPR.2015.7298682.
- [110] Z. Selmi, M. Ben Halima, and A. M. Alimi. Deep learning system for automatic license plate detection and recognition. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 01, pages 1132–1138, November 2017.
- [111] Pierre Sermanet, David Eigen, Xiang Zhang, Michael Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. 2nd international conference on learning representations, iclr 2014. January 2014. 2nd International Conference on Learning Representations, ICLR 2014 ; Conference date: 14-04-2014 Through 16-04-2014.
- [112] Yantao Shen, Tong Xiao, Hongsheng Li, Shuai Yi, and Xiaogang Wang. Learning deep neural networks for vehicle re-id with visual-spatio-temporal path proposals. 08 2017.
- [113] Baoguang Shi, Xiang Bai, and Cong Yao. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *CoRR*, abs/1507.05717, 2015.

- [114] C. Shi, C. Wang, B. Xiao, Y. Zhang, S. Gao, and Z. Zhang. Scene text recognition using part-based tree-structured character detection. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2961–2968, 2013. doi: 10.1109/CVPR.2013.381.
- [115] P. Shivakumara, T. Q. Phan, and C. L. Tan. A gradient difference based technique for video text detection. In *2009 10th International Conference on Document Analysis and Recognition*, pages 156–160, 2009. doi: 10.1109/ICDAR.2009.85.
- [116] P. Shivakumara, S. Bhowmick, B. Su, C. L. Tan, and U. Pal. A new gradient based character segmentation method for video text recognition. In *2011 International Conference on Document Analysis and Recognition*, pages 126–130, 2011. doi: 10.1109/ICDAR.2011.34.
- [117] Palaiahnakote Shivakumara, Weihua Huang, Trung Phan, and Chew Lim Tan. Accurate video text detection through classification of low and high contrast images. *Pattern Recognition*, 43:2165–2185, 06 2010. doi: 10.1016/j.patcog.2010.01.009.
- [118] S. M. Silva and C. R. Jung. Real-time brazilian license plate detection and recognition using deep convolutional neural networks. In *2017 30th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, pages 55–62, October 2017.
- [119] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. URL <http://arxiv.org/abs/1409.1556>.
- [120] J. Sochor, A. Herout, and J. Havel. Boxcars: 3d boxes as cnn input for improved fine-grained vehicle recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3006–3015, June 2016. doi: 10.1109/CVPR.2016.328.
- [121] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11(110):3371–3408, 2010. URL <http://jmlr.org/papers/v11/vincent10a.html>.
- [122] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. volume 1, pages I–511, 02 2001. ISBN 0-7695-1272-0. doi: 10.1109/CVPR.2001.990517.

- [123] Z. Wang, L. Tang, X. Liu, Z. Yao, S. Yi, J. Shao, J. Yan, S. Wang, H. Li, and X. Wang. Orientation invariant feature embedding and spatial temporal regularization for vehicle re-identification. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 379–387, October 2017.
- [124] Hsien-Huang P. Wu, Hung-Hsiang Chen, Ruei-Jan Wu, and Day-Fann Shen. License plate extraction in low resolution video. In *18th International Conference on Pattern Recognition (ICPR'06)*, volume 1, pages 824–827, 2006.
- [125] Yu Wu, Yutian Lin, Xuanyi Dong, Yan Yan, Wei Bian, and Yi Yang. Progressive learning for person re-identification with one example. *IEEE Transactions on Image Processing*, 28(6):2872–2881, 2019.
- [126] Yue Wu and Prem Natarajan. Self-organized text detection with minimal post-processing via border learning. pages 5010–5019, 10 2017. doi: 10.1109/ICCV.2017.535.
- [127] T. Xiao, H. Li, W. Ouyang, and X. Wang. Learning deep feature representations with domain guided dropout for person re-identification. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1249–1258, 2016.
- [128] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention, 2016.
- [129] L. Yang, P. Luo, C. C. Loy, and X. Tang. Dgd. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3973–3981, June 2015. doi: 10.1109/CVPR.2015.7299023.
- [130] C. Yao, X. Bai, W. Liu, Y. Ma, and Z. Tu. Detecting texts of arbitrary orientations in natural images. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1083–1090, 2012. doi: 10.1109/CVPR.2012.6247787.
- [131] Qixiang Ye, Wen Gao, and Debin Zhao. Fast and robust text detection in images and video frames. *Image and Vision Computing*, 23:565–576, 06 2005. doi: 10.1016/j.imavis.2005.01.004.
- [132] C. Yi and Y. Tian. Text string detection from natural scenes by structure-based partition and grouping. *IEEE Transactions on Image Processing*, 20(9):2594–2605, 2011. doi: 10.1109/TIP.2011.2126586.
- [133] Fang Yin, Rui Wu, Xiaoyang Yu, and Guanglu Sun. Video text localization based on adaboost. *Multimedia Tools and Applications*, 78, 03 2019. doi: 10.1007/s11042-018-6064-8.

- [134] Yuhui Yuan, Kuiyuan Yang, and Chao Zhang. Hard-aware deeply cascaded embedding. pages 814–823, 10 2017. doi: 10.1109/ICCV.2017.94.
- [135] Liu Yuliang, Jin Lianwen, Zhang Shuaitao, and Zhang Sheng. Detecting curve text in the wild: New dataset and new solution, 2017.
- [136] Dominik Zapletal and Adam Herout. Vehicle re-identification for automatic video traffic surveillance. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 25–31, 2016.
- [137] Fangneng Zhan and Shijian Lu. Esir: End-to-end scene text recognition via iterative image rectification, 2019.
- [138] Sheng Zhang, Yuliang Liu, Lianwen Jin, and Canjie Luo. Feature enhancement network: A refined scene text detector, 2017.
- [139] Yiheng Zhang, Dong Liu, and Zheng-Jun Zha. Improving triplet-wise training of convolutional neural network for vehicle re-identification. In *2017 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1386–1391. IEEE, 2017.
- [140] Liang Zheng, Yi Yang, and Alexander G. Hauptmann. Person re-identification: Past, present and future. *ArXiv*, abs/1610.02984, 2016.
- [141] W. Zhou, H. Li, Y. Lu, and Q. Tian. Principal visual word discovery for automatic license plate detection. *IEEE Transactions on Image Processing*, 21(9):4269–4279, September 2012.
- [142] Yi Zhou and Ling Shao. Cross-view gan based vehicle generation for re-identification. 01 2017. doi: 10.5244/C.31.186.
- [143] Yi Zhou, Li Liu, and Ling Shao. Vehicle re-identification by deep hidden multi-view inference. *IEEE Transactions on Image Processing*, 27(7):3275–3287, 2018.
- [144] Y. Zhou and L. Shao. Viewpoint-aware attentive multi-view inference for vehicle re-identification. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6489–6498, June 2018. doi: 10.1109/CVPR.2018.00679.
- [145] Jianqing Zhu, Huanqiang Zeng, Zhen Lei, Shengcai Liao, Lixin Zheng, and Canhui Cai. A shortly and densely connected convolutional neural network for vehicle re-identification. In *2018 24th International Conference on Pattern Recognition (ICPR)*, pages 3285–3290. IEEE, 2018.
- [146] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. pages 2242–2251, 10 2017. doi: 10.1109/ICCV.2017.244.

- [147] Xiaojin Zhu. Semi-supervised learning literature survey. *Comput Sci, University of Wisconsin-Madison*, 2, 07 2008.

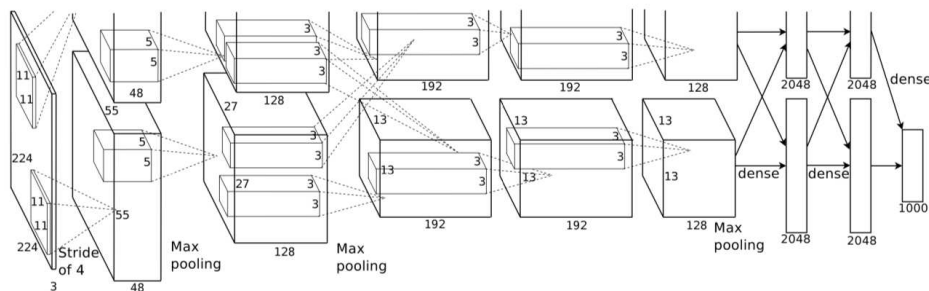
# Appendices

# Appendix A

## Deep Learning Models

### A.1 AlexNet

AlexNet participated in the ImageNet Large Scale Visual Recognition Challenge in September 2012. The network achieved a top-5 error of 15.3%, 10.8% points lower than that of the runner up. The architecture shown in Figure A.1 consists of eight layers: five convolutional layers and three fully-connected layers.



**Figure A.1:** *AlexNet Architecture.*

AlexNet uses Rectified Linear Units (ReLU) instead of the tanh function. Using ReLU, AlexNet reached a 25% error on the CIFAR-10 dataset six times faster than a CNN using tanh. Furthermore, AlexNet allows for multi-GPU training by dividing the model's neurons on multiple GPUs. AlexNet improvements were in terms of speed and in solving the overfitting problem since AlexNet has 60 million parameters. Hence, a technique called DropOut is used that consists of turning of neurons with predetermined probability 40% example, this forces each neuron to have more robust features that can be used with other random neurons.



## A.2 VGG-16

VGG-16, proposed by K. Simonyan and A. Zisserman from the University of Oxford in the paper “Very Deep Convolutional Networks for Large-Scale Image Recognition”, was submitted to ILSVRC-2014. It improves AlexNet by replacing large kernel-sized filters (11 and 5 in the first and second convolutional layer, respectively) with multiple 3x3 kernel-sized filters one after another. The model achieves 92.7% top-5 test accuracy in ImageNet. The architecture of VGG-16 (shown in Figure A.2) is composed of 16 layers in total. The input to the first convolutional layer is of size 224 x 224 RGB image. Followed by a stack of convolutional (conv.) layers in which filters with a small receptive field were used with a stride value of 1. Spatial pooling is achieved using max-pooling with stride 2. Finally, three fully connected layers (FCs) follow. The first two have 4096 channels each and the third, having several channels depending on the number of classes, acts as a classifier.

	Layer	Feature Map	Size	Kernel Size	Stride	Activation
Input	Image	1	224 x 224 x 3	-	-	-
1	2 X Convolution	64	224 x 224 x 64	3x3	1	relu
	Max Pooling	64	112 x 112 x 64	3x3	2	relu
3	2 X Convolution	128	112 x 112 x 128	3x3	1	relu
	Max Pooling	128	56 x 56 x 128	3x3	2	relu
5	2 X Convolution	256	56 x 56 x 256	3x3	1	relu
	Max Pooling	256	28 x 28 x 256	3x3	2	relu
7	3 X Convolution	512	28 x 28 x 512	3x3	1	relu
	Max Pooling	512	14 x 14 x 512	3x3	2	relu
10	3 X Convolution	512	14 x 14 x 512	3x3	1	relu
	Max Pooling	512	7 x 7 x 512	3x3	2	relu
13	FC	-	25088	-	-	relu
14	FC	-	4096	-	-	relu
15	FC	-	4096	-	-	relu
Output	FC	-	1000	-	-	Softmax

Figure A.2: VGG16 Architecture.

However, the VGG-16 network is slow to train, and The network architecture weights themselves are pretty large. Compared with other networks, VGG-16 has more parameters with lower accuracy.

## A.3 GoogLeNet (Inceptionv1)

An inception network is a deep neural network with an architectural design that consists of repeating components referred to as Inception modules, where the network learns

which filter size to use or type of pooling. This increased performance and adds the ability to extract features from input data at varying scales.

GoogLeNet is a variant of the inception network. It has nine inception modules stacked linearly. It is 22 layers deep (27, including the pooling layers) as shown in Figure A.3. The architecture contains 1x1 Convolution at the middle of the network. Moreover, global average pooling is used at the network’s end instead of using fully connected layers.

type	patch size/ stride	output size	depth	# 1×1	# 3×3 reduce	# 3×3	# 5×5 reduce	# 5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

Figure A.3: GoogLeNet Architecture.

Inception Modules are used in Convolutional Neural Networks to allow for more efficient computation and deeper Networks through a dimensionality reduction. Figure A.4 shows the inception module used in GoogLeNet.

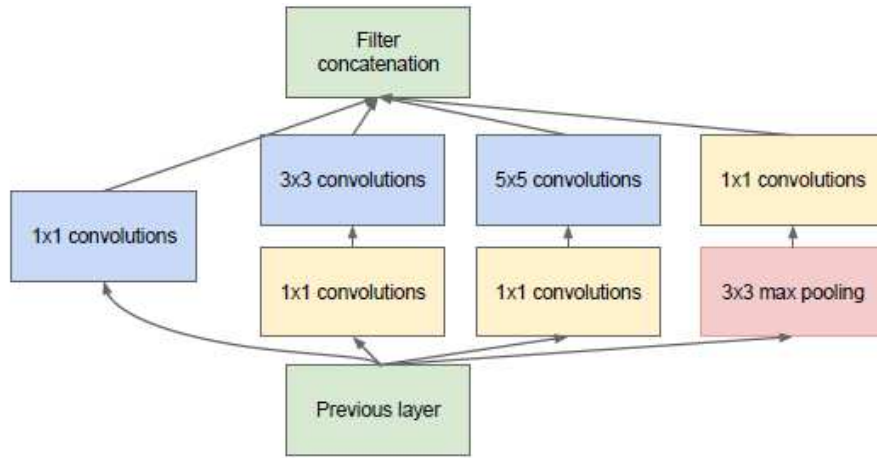


Figure A.4: GoogLeNet Inception Module.

## A.4 ResNet

ResNet, a deep Residual Network, is considered the most groundbreaking computer vision community in the last few years.

Since AlexNet, the state-of-the-art CNN architecture is going deeper and deeper. While AlexNet had only five convolutional layers, the VGG network and GoogleNet had 19 and 22 layers, respectively. However, training very deep neural networks is hard as they suffer from the vanishing gradient problem where the back-propagated gradient becomes very small as it approaches the first layers. To address this problem ResNet proposed "an identity short-cut connection" shown in Figure A.5.

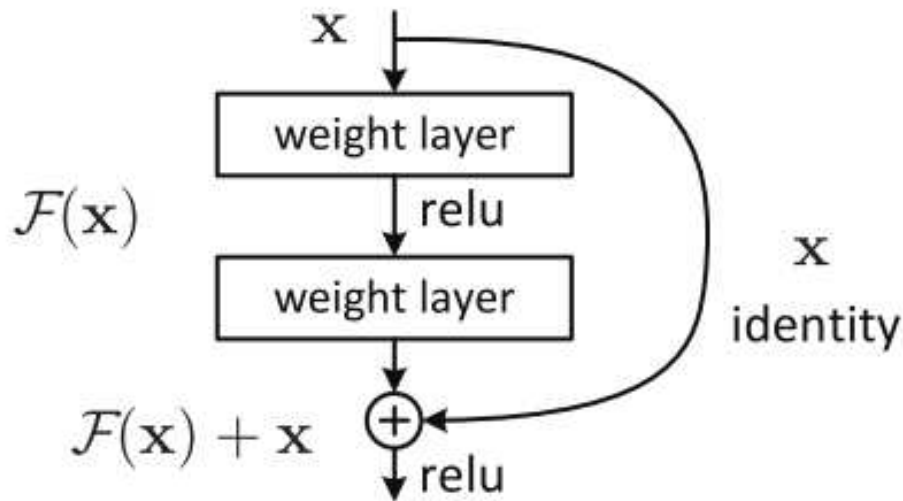


Figure A.5: *Residual Block.*

These identity short-cuts allow the network to skip layers which in turn make it possible to train very deep networks. ResNet uses batch normalization[53] after each convolution network and does not use dropout. The full ResNet50 architecture is shown in Figure A.6.

34-layer residual

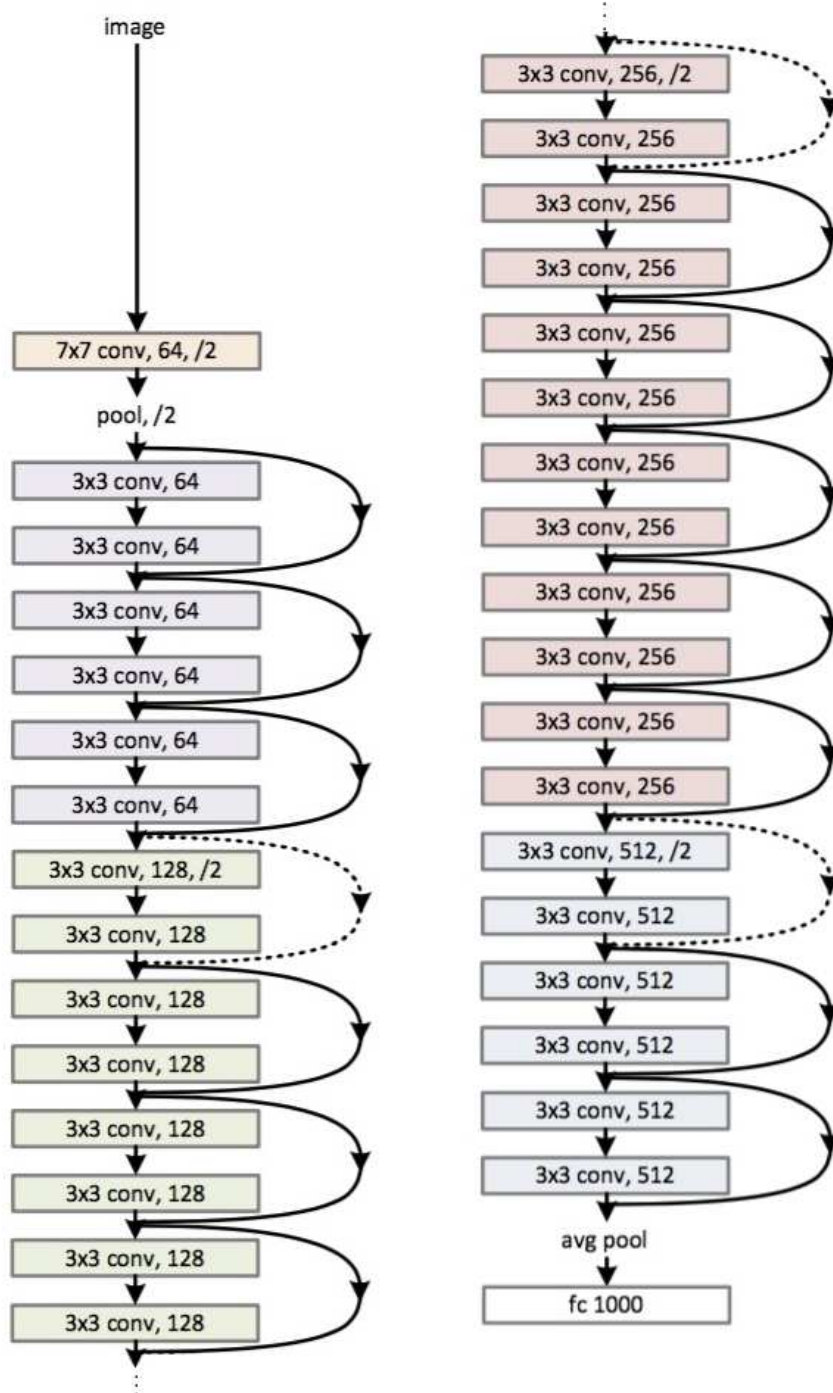


Figure A.6: ResNet50 Architecture.

# Appendix B

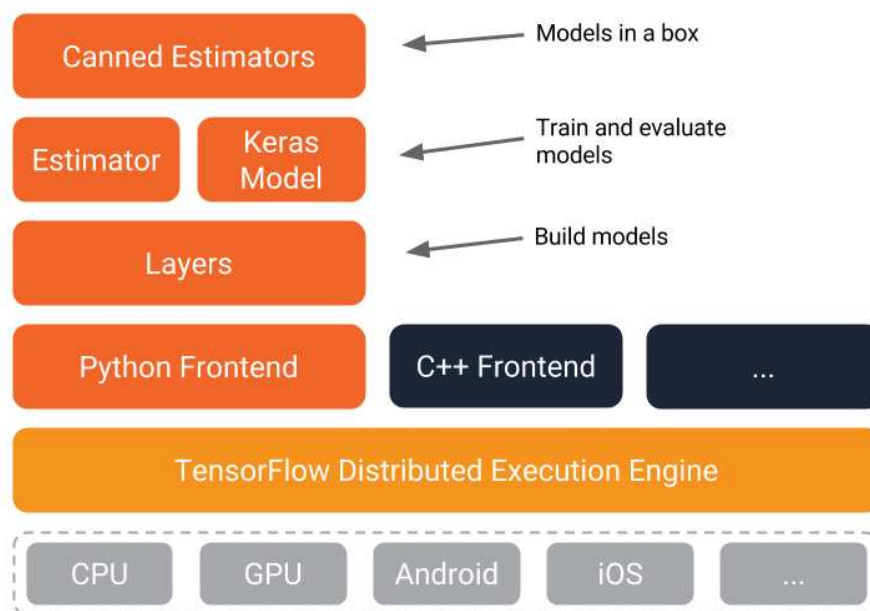
## Deep Learning ToolBox

### B.1 Caffe

Caffe [60] provides researchers with a framework for state-of-the-art deep learning algorithms and a collection of reference models. Caffe started in late 2013 is the first mainstream industry-grade deep learning toolkit. Due to its excellent convolutional model. Caffe is released under the BSD 2-Clause license. Speed makes Caffe perfect for research experiments and commercial deployment. Caffe can process over 60M images per day with a single Nvidia K40 GPU. That's 1 ms/image for inference and 4 ms/image for learning, and more recent library versions are still faster. Caffe is C++-based, allowing to work on multiple devices. Caffe supports C++, Matlab, and Python programming interfaces. Caffe has a large user community that contributes to its deep net repository known as the "Model Zoo." AlexNet and GoogleNet are two popular user-made nets available to the community. More details can be found on <http://caffe.berkeleyvision.org/>.

### B.2 TensorFlow

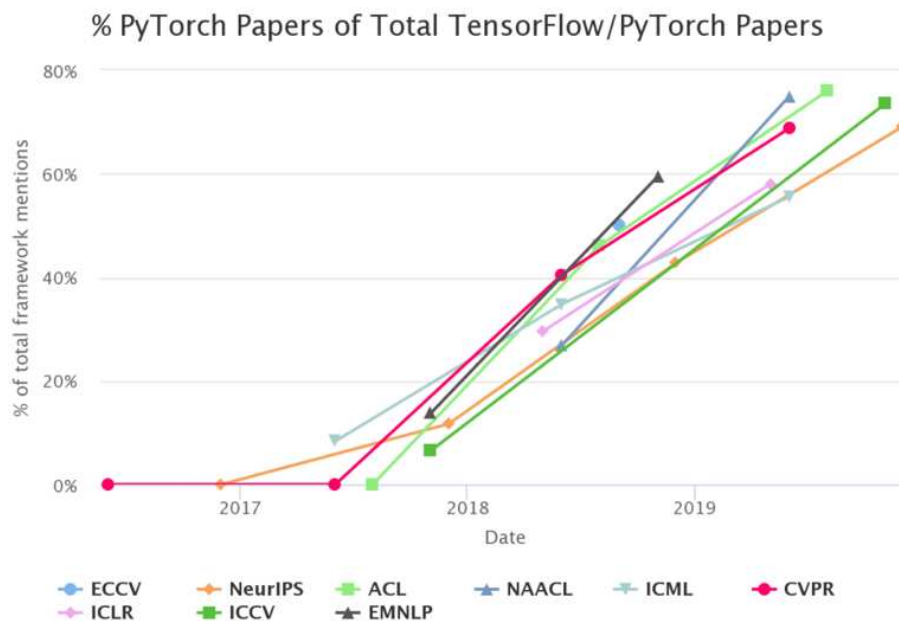
TensorFlow, sourced by Google, is one the most used deep learning toolkit in the world. It is released under the Apache 2.0 open source license in late 2015. It now supports Python, JavaScript, C++, JAVA, Go, and Swift. Few other languages have been added recently, such as C#, Haskell, Julia, MATLAB, R, Ruby, Rust, and Scala. It supports a broad set of capabilities such as image classification and recognition, handwriting and speech recognition, and natural language processing (NLP). Additionally, TensorFlow is supported in almost all Cloud Environments, such as Google and Amazon. TensorFlow supports fine-grain network layers that allow users to build new complex layer types without implementing them in a low-level language. More details can be found on <https://www.tensorflow.org/>. Furthermore, TensorFlow is more accessible to the wider community thanks to Keras, a high-level language that sits on top of TensorFlow (or Theano CNTK), as shown in Figure B.1.



**Figure B.1:** *Tensorflow as an engine to Keras.*

### B.3 Keras

Keras is a model-level library that provides high-level blocks for the development of deep learning models. Keras developers have focused their efforts on creating high-level models by neglecting low-level operations such as tensor products, convolutions, etc. These operations have been entrusted to specialize and well-optimized tensor manipulation libraries that already exist, thus acting as a back-end engine for Keras. Recently, it was made possible to using Keras and its back-end engines interchangeably. So developers can use high-level operations and low-level functions such as tensor operations or convolutions.. when needed; despite the popularity of Keras and its back-end engines, especially TensorFlow, deep learning models developed with this toolkit were challenging to debug. Once a model graph is deployed, not much can be done to see what's happening in run-time. Hence, researchers were more inclined to use PyTorch as a deep learning toolkit. This can be seen in the number of publications that use PyTorch versus TensorFlow, as shown in Figure B.2. Though recently, TensorFlow made some tools to ease debugging. Researchers still favor PyTorch.



**Figure B.2:** *PyTorch Vs Tensorflow in terms of paper publications.*

## B.4 PyTorch

PyTorch is an open-source machine learning library based on the Torch library, used for applications such as computer vision and natural language processing, primarily developed by Facebook’s AI Research lab (FAIR). It is free and open-source software released under the Modified BSD license. Although the Python interface is more polished and the primary focus of development, PyTorch also has a C++ interface. PyTorch is still less popular than TensorFlow; in fact, 26% of python developers use TensorFlow while only 15% uses PyTorch. However, the dynamic models building capability offered by PyTorch, i.e., the model can change at run-time, made PyTorch very popular in the scientific community. Furthermore, contrary to Tensflow, which imposes a specific debugger (TensorFlow Debugger), Pytorch can be easily integrated with other debugging frameworks like PyCharm or ipdb.