UNIVERSITÀ
DI TORINO

Doctoral Dissertation
Doctoral Program in Computer Science (34^{th}cycle)

Multimodal Learning and Feature Fusion Methodologies for Real Case Scenarios

By

Mirko Zaffaroni

Supervisor:
Prof. Marco Grangetto

Prof. Laura Toni, University College of London
Prof. Francesco Verdoja, Aalto University
Prof. Attilio Fiandrotti, Università di Torino

Università degli Studi di Torino
2023

Declaration

I hereby declare that, the contents and organization of this dissertation constitute my own original work and does not compromise in any way the rights of third parties, including those relating to the security of personal data.

Mirko Zaffaroni
2023

I would like to dedicate this thesis to my loving parents

Acknowledgements

I would like to acknowledge Links Foundation for giving me the opportunity to carry out my research activities at their research centre. Additionally, I want to express my gratitude to my supervisor for guiding and supporting me throughout my doctoral years

Abstract

The three key ingredients underlying most Machine Learning algorithms are *features*, *task*, and *model*. The *features* describe the input data, and the *model* generates a correct mapping between the features and the output. What makes this mapping possible is the learning *task*. This thesis will deal extensively with the first of these three ingredients: the features. Namely, we will analyze in depth how features from multiple sources can be merged and used to solve the task with a specific focus on Computer Vision tasks. Thanks to modern architectures based on Convolutional Neural Networks, extracting rich hierarchies of features from images in a data-driven fashion is becoming easier and more accessible. However, what happens when multiple inputs come from different, possibly multimodal, sources? In this thesis, we will try to answer these questions by exploring and designing various methodologies for feature fusion and how these are essential tools for combining multimodal inputs. The methodologies presented are the result of my research works, mainly from the resolution of real problems in the emergency environment, where combining all the information sources in a single predictive architecture is key to supporting the decisions of the first responders. This thesis focuses on designing novel methodologies for multimodal feature fusion through the concatenation of different models (CNN, LSTM, etc ..) for multimodal input. The sources of the inputs are of various origins, such as data extracted from video games, social networks, tweets composed of images and text, and aerial images. Starting from a multi-branch structure, with synthetic inputs extracted from Grand Theft Auto 5, we have devised an architecture for the resolution of a regression task such as estimating the distance and speed of the surrounding vehicles, obtaining good results thanks to the combination of very different features as semantic, movement and spatial location features. Following these results and moving from the automotive context to the emergency one, we have extended the design of the architecture, which has become multi-task, with the addition of more loss functions, one for each output. Thanks to

the features fusion techniques, we have transformed a difficulty into a strength, since thanks to the careful choice and the combination of the loss functions, it was possible to force the features extraction to propose features of higher semantic quality. Finally, as the last follow-up, we have enriched the previous architecture with a dedicated loss function for the features in the merged layer. This function ensures that a more compact representation is obtained within the latent space for very similar features. The results of this last method are two-fold: the design of a multimodal feature fusion methodology that can be applied regardless of the number of inputs or tasks and the definition of a loss function that improves the mapping of the features within the latent space. These represent the most significant contributions of my research work to the multimodal feature fusion task.

Contents

List of Figures **xi**

List of Tables **xiv**

1 Introduction **1**

2 Features Fusion with Multimodal learning **6**

2.1 Hand-crafted vs. Data-driven features 7

2.2 Fusion Strategies 9

2.2.1 Direct Fusion/Early Fusion 9

2.2.2 Flow Fusion 10

2.2.3 Adaptive Fusion 11

2.2.4 Fusion Operations 12

2.2.5 Loss Functions and Feature Fusion Methodologies 13

2.3 The Multimodal tasks 15

2.3.1 Modalities fusion strategies 15

2.3.2 The template of a multi-branch architecture 16

2.4 What is next 17

I Automotive **18**

3 Automotive and synthetic datasets **19**

4 Estimation of speed and distance of surrounding vehicles from a dashboard camera point-of-view **22**

4.1 Introduction . . . 22
4.2 Methodology . . . 23
4.2.1 Data collection . . . 23
4.2.2 Models . . . 25
4.3 Network training . . . 27
4.4 Experimental results . . . 28
4.4.1 Testing on synthetic dataset . . . 28
4.4.2 Testing on real dataset . . . 30
4.4.3 Testing on the road . . . 31
4.4.4 Computational cost . . . 32
4.5 Conclusions . . . 32
4.6 Acknowledgement . . . 33

5 AA-SGAN: adversarially augmented Social GAN with synthetic data **34**

5.1 Introduction . . . 35
5.2 Background and Related Works . . . 37
5.2.1 Trajectory Prediction Methods . . . 37
5.2.2 Real-World Datasets . . . 38
5.2.3 Data augmentation and synthetic datasets . . . 39
5.3 Proposed Method . . . 40
5.3.1 Problem definition . . . 40
5.3.2 Architecture . . . 40

5.3.3 Training Procedure 43
5.4 Experimental results 45
5.4.1 Path Prediction Accuracy 47
5.4.2 Ablation Study 48
5.5 Conclusions 51

II Natural disasters management 52

6 The emergency scenarios 53

7 AI-based flood event quantification using online media and satellite data 57
7.1 Introduction 57
7.2 Related Work 58
7.3 Approach 59
7.4 Results 63
7.5 Analysis and conclusions 64
7.6 Acknowledgments 64

8 Road passability detection during flood events using social media data 65
8.1 Introduction 66
8.2 Related Work 67
8.3 Dataset 69
8.3.1 Metadata 71
8.3.2 Images 71
8.4 Proposed Solutions 71
8.4.1 Algorithm Based on Metadata Only 71
8.4.2 Algorithms Based on Image Only 76
8.4.3 Algorithm Based on Metadata and Visual Information . . . 83

8.5 Evaluation and Results 84
8.5.1 Results Using Metadata Only 85
8.5.2 Results Using Images Only 85
8.5.3 Results Using Images and Metadata 89
8.6 Conclusions 90

9 Emergency scene description using deep learning approaches 92
9.1 Introduction 92
9.2 LADI Dataset 93
9.3 Models 94
9.4 Experiments 96
9.5 Conclusion 100
9.6 Acknowledgements 101

10 Water segmentation for flood detection and monitoring 102
10.1 Introduction 102
10.2 Related Works 103
10.3 The Water Segmentation Open Collection Dataset 104
10.4 Methodology 106
10.5 Evaluation metrics and configurations 107
10.6 Results 109
10.7 Conclusion and Future Works 111
10.8 Acknowledgements 112

11 Conclusions 113

References 117

Appendix A Compactness Loss equation derivation 130

List of Figures

2.1 Machine Learning (Hand-crafted) vs. Deep Learning (Data-driven features) . . . 7

2.2 Direct fusion template . . . 10

2.3 Flow fusion template. Each box represents a feature map. . . . 10

2.4 Example of multimodal architecture . . . 16

3.1 Examples of synthetic datasets created with the aim of GTAV . . . 20

4.1 Photorealistic frame extracted from the game environment, featuring an overlay of the prediction . . . 24

4.2 Proposed model for vehicle distance prediction . . . 25

4.3 Proposed model for vehicle speed prediction . . . 26

4.4 RMSE on distance (a) and speed (b) estimate. . . . 29

4.5 RMSE on distance estimate (synthetic and real images). . . . 30

4.6 Example of distance estimate in real environment. . . . 31

5.1 Real, synthetic, and synth-augmented trajectories. . . . 36

5.2 Social GAN for pedestrian trajectories prediction adversarial architecture . . . 38

5.3 AA-SGAN for adversarially augmented pedestrian trajectories prediction architecture . . . 41

5.4 Examples of frames from the JTA dataset and the corresponding trajectories 46
5.5 Visual comparison between the groud-truth and predicted trajectory 49

6.1 The occurrence of natural disasters and flood events since 1950 . . . 54
6.2 Examples of emergency events related to flooding disasters 55

7.1 Sample images extracted from articles from the NITD dataset 59
7.2 Ensemble architecture used for the NITD task 59
7.3 Sample from MFLE 60
7.4 Architecture used for the MFLE task 61
7.5 Sample from CCSS 62
7.6 Model used for the CCSS task 62

8.1 Examples of images from the MediaEval2019 dataset 72
8.2 Correlation matrix of the features 75
8.3 Architecture of the neural network to process metadata 75
8.4 Schematic of the ensemble architecture 77
8.5 Representation of the double-ended architecture. 79
8.6 Examples of images created with Grad-CAM 80
8.7 Double-ended classifier with compactness loss 82
8.8 Combination of the Double-ended classifier with compactness loss and the metadata system. 83
8.9 Evolution of F1-score on the road evidence task as we ensemble more networks 88
8.10 Evolution of F1-score on the road passability task as we ensemble more networks 88
8.11 Examples of tweets within the dataset 90

9.1 Examples of images contained in the LADI dataset 94

9.2 Different class distributions for the LADI Dataset 95
9.3 Single classifier 97
9.4 Five classifiers 97
9.5 Five Networks 98
9.6 Team scores in terms of MAP 100

10.1 WSOC proposed experimental methodology 106
10.2 Examples of the best samples on the validation set 110

List of Tables

2.1 Overview of feature fusion methodologies examined in the following chapter 17

4.1 Execution time of different algorithmic steps. 32

5.1 Trajectory prediction accuracy in terms of ADE and FDE 44

5.2 Trajectory prediction accuracy of AA-SGAN versus independent training of the Augmenter 48

5.3 ADE and FDE values for 1-to-1 and 1-to-10 ratio between real and synthetic 50

7.1 MediaEval2019 results per subtask 63

8.1 MediaEval2018 dataset information 70

8.2 MediaEval2019 metadata description 73

8.3 F1-scores on the challenge test set for both tasks using metadata information only 85

8.4 F1-scores on the challenge test set for both tasks using only the images from the tweet 87

8.5 F1-scores on the challenge test set for both tasks using the metadata and image information 89

9.1 Best scores obtained during training 100

10.1 Water Segmentation Open Collection (WSOC) key metrics. 105

10.2 Model accuracy comparison 108

10.3 Models performance comparison in terms of VRAM, size and prediction speed. 108

Chapter 1

Introduction

In recent years in the field of computer vision, there has been a growing interest in neural networks and Deep Learning (DL). This is after AlexNet's results on the benchmark ImageNet dataset [1, 2], managing to obtain a ten percentage point increase in top-1-accuracy for the image classification task. The different approach to extracting the features was ground-breaking in neural networks after the advent of modern DL. We have gone from features handcrafted wisely by the data scientist to a data-driven approach, where learning is left to the learning algorithm. In the field of Computer Vision, the building block behind most Deep Learning algorithms is the Convolutional Neural Networks (CNN) [3]. This type of network takes its name from the convolutional layers, which take the information within the image and transform it into features maps thanks to convolutional filters. The information is spatially reduced as the layers follow one another, thanks to the pooling layers. The progressive reduction of spatial resolution is called down-sampling. This process makes it possible to obtain a compact representation of the information, the embedding. With this term, we want to indicate a numerical vector representing a point in the space of the features. This space is also called latent space. The feature learning process involves identifying the best representation of the input data within the latent space through the embedding that describes it. In this thesis, we will use the term embedding and features interchangeably to indicate the same concept: a compact numerical vector representation of the image content. Then we can define a CNN as an extractor of features, which can then be used as the input of any classifier (Decision Tree, SVM, etc ..). Another advantage introduced by deep learning models is the ease with which features can be merged, obtaining input with more information

for the next model layers. This can be done on different levels and in different ways. In the following chapters, we will introduce some ideas behind feature fusion with the related methodologies and approaches. Furthermore, we will also analyze how feature fusion is a valuable and indispensable tool for multimodal tasks, i.e., when the inputs we feed to the model have been created using different sensors (e.g., LIDAR, IR, etc ..) or come from different domains (audio, text, images). This thesis will present some application cases in which features fusion techniques have been successfully applied to resolve multi-modal tasks.

In this thesis, we will explore my Ph.D. research studies pursued at LINKS Foundation, a research institute located in Turin, in the "Data Science for Societal and Industrial Application" department. The laboratory works mainly with projects of the Societal Challenges of the Horizon 2020 (H2020) funding program of the European Commission. The main scopes of the societal challenges in H2020 are to protect the freedom and security of Europe and its citizens. In this context, I have focused my research on deep learning for multi-modal imaging applied to security and emergency application. In the following chapters, I will present the methods used to develop deep learning systems capable of merging features from different sources to solve real-case scenarios by dividing the presentation into two main topics: 1) the automotive task and the emergency scenario field. The thesis is structured as follows:

- in **Chapter 2:** we will further define the concept of features and analyze what differentiates the hand-crafted features of classic machine learning from the data-driven features of modern deep learning. We will also explore how deep learning uses and merges this information. Finally, we will present the multimodal task and an example of architecture that combines multimodal learning with the methodologies for feature fusion.
- in **Chapter 3:** we present the automotive task and the usefulness of synthetic data in its context. In fact, the rapid transformation of the automotive industry with new technologies such as autonomous vehicles, ADAS and connected cars relies heavily on computer vision. To develop these technologies, large, labelled datasets are needed to train and evaluate the performance of computer vision models. However, collecting and labelling real-world data is costly, time-consuming, and may not always be feasible. Computer graphics simulation

is emerging as a powerful alternative, quickly providing a large dataset and inexpensive resources. It allows for the generation of diverse scenarios with different weather and lighting conditions, making it a valuable tool for the automotive industry to train and evaluate the performance of computer vision models. Computer graphics simulation, such as the use of game engines, can also provide data that is difficult to obtain in real-world scenarios and improve the model's robustness and generalization.

– in **Chapter 4:** we have developed a system capable of querying the game engine of a popular automotive game. Leveraging that system, we created a synthetic dataset containing the distance, position, and speed information of the surrounding vehicles from a dashboard-camera point of view. The dataset was used for training two multi-branch models to calculate the distance and speed of the surrounding vehicles. The first model uses the vehicle crop and the bounding box coordinates as input. The inputs are then transformed into semantic and spatial features of the vehicle. The second model takes as input the optical-flow frame and the spatial coordinates of the vehicle, thus obtaining movement and spatial features. For both approaches, we applied a direct fusion through concatenation. Finally, both models feed the merged features to a regression branch, which predicts the required measure.

– in **Chapter 5:** We propose a method and architecture to augment synthetic trajectories at training time with an adversarial approach, including an "ad hoc" loss function to merge the features of real and synthetic trajectories in a common feature space. This allows the model to learn better from real-world and synthetic data, resulting in improved predictions for pedestrian trajectory prediction in applications such as autonomous driving or service robotics. Our proposed method improves the performance of state-of-the-art generative models when evaluated on real-world trajectories.

– in **Chapter 6:** we present how interest in computer vision techniques for emergency scenarios use cases to detect and predict natural disasters, particularly floods, is growing. Computer vision algorithms can analyze satellite imagery, drone footage, and other visual data to detect changes in land cover, water levels, and other indicators of potential flooding. These algorithms can also analyze social media images and videos to detect and track floods in real time. Additionally, computer vision algorithms can be used in conjunction with other

sensor data, such as weather forecasts and river flow measurements, to improve the accuracy of flood predictions. Overall, computer vision algorithms have the potential to significantly improve our ability to detect and predict floods and thus help mitigate the impacts of these natural disasters.

– in **Chapter 7:** trying to solve a common problem in the emergency field, namely the lack of simplicity of labels, we have extended the dataset provided to us to develop a multi-branch model capable of obtaining better performance for the use case. Starting from a simple binary label assigned to the image to identify if there were people in the scenario with the water above their knees. We annotated every single knee within the images and used the two inputs: knee and full image, to refine a baseline model trained only with scene images. This multi-branch model can learn context (the whole image) and local (area around the knee) features and merge them with direct fusion through concatenation to improve baseline predictions.

– in **Chapter 8:** we studied the problem of detecting blocked/open roads from photos during floods by applying a two-step approach based on classifiers: does the image have evidence of road? If it does, is the road passable or not? We propose a single double-ended neural network (NN) architecture that addresses both tasks simultaneously. Both problems are treated as a single-class classification problem using a compactness loss. The study was performed on tweets posted during flooding events containing 1) metadata, 2) textual data, and 3) visual information. We studied the usefulness of each data feature and the combination of both. This analysis was carried out using a dedicated loss function on the merging layer. The purpose of the function is to provide a better characterization of the features and a better clustering of similar features in the latent space. The compactness loss, in combination with the merging of the features experimentally, proved to be the best model, as the best scoring model, and won the challenge of MediaEval 2018 for the Flood classification task.

– in **Chapter 9:** we present our proposed model for the Disaster Scene Description and Indexing (DSDI) Challenge of TRECVIDI2020. For the challenge, we used the LADI Dataset, composed of images, each presenting 32 tags at most. Each tag belongs to one of five macro categories: vehicle, infrastructure, environment, water, and damage. We analyzed the best approach to use the

features extracted from the images and the best blending mode. The strategies we have implemented are 1) common backbone and common classifier, 2) common backbone but dedicated classifier for each dataset macro category, and 3) dedicated backbone and classifier. All features have been combined through concatenation and direct fusion. Experimentally, the last approach was the best, dividing the features for each macro category and leaving the multi-tagging to a dedicated model.

– in **Chapter 10:** we present a work for detecting flood events. For this task, it is crucial to have a considerable amount of images containing water pixels. Since there are not many available, we have created a collection of datasets, namely WSOC, containing images with water elements. Subsequently, we made a segmentation model analysis, trying to identify the best from an architectural and feature extraction point of view. Many of these models have a standard feature extractor combined with subsequent segmentation layers. These layers use a flow feature fusion methodology to combine information from different resolution scales. Thanks to this feature fusion approach, it was possible to obtain good results in the segmentation phase, and we also presented the model that best suits our scenario.

– in **Chapter 11:** we lay the conclusions for the thesis.

Chapter 2

Features Fusion with Multimodal learning

In Machine Learning (ML), the term *features* refer to a piece of information obtained from the input, allowing the resolution of a computational task such as prediction, regression, or generation. The features are derived from the raw input and are typically represented in the form of a numerical vector. We can define this transformation as $F(I) = E_n$, where F represents a function or a set of functions, known as *feature extractor*, that transform the input I into n set of embedding E (the features). In the machine learning field, the number of features, or the size of the feature vector n, depends on the number of analyses and processing methods performed on the raw input. For example, for an RGB image, normally the raw input consists of values of the pixels that compose it. While for Deep Learning algorithms, n will be the size of the embedding created by the model and will be a hyper-parameter defined a priori. In this chapter, we will explore the main advantages of using features derived from a Deep Learning algorithm compared to a classic one. Furthermore, we will present the various methodologies with which the information in the embeddings generated by Deep Learning models can be easily merged and combined. We will also analyze how these approaches can be successfully chained within a multimodal training pipeline.

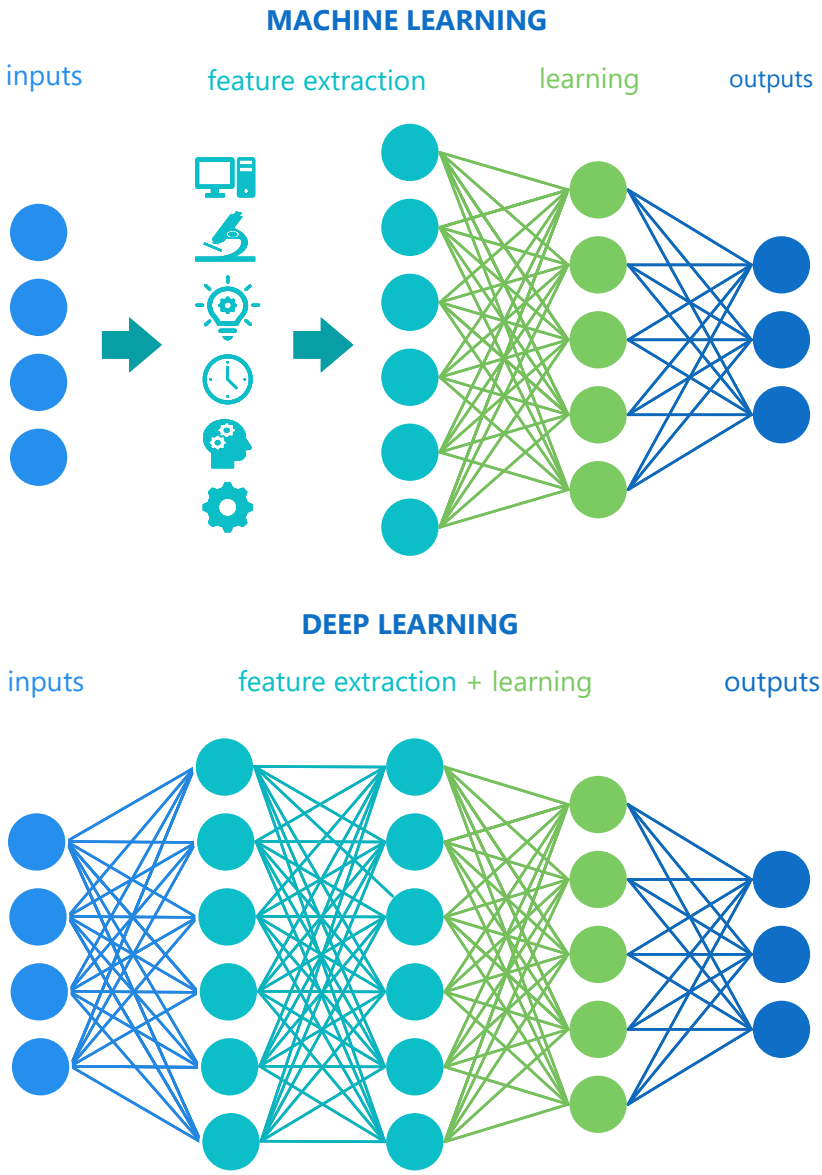


Fig. 2.1 Machine Learning (Hand-crafted) vs. Deep Learning (Data-driven features)

2.1 Hand-crafted vs. Data-driven features

In classic machine learning, the choice and/or extraction of features is an engineering work of very high precision. Carefully engineered features are necessary to solve the task successfully. In Deep Learning, on the other hand, this feature extraction work is left to the model in a data-driven fashion, which learns which features to use

and how to apply them for prediction during the learning process. When working with images, raw pixels are not fed directly to the algorithms, and this is due to the poor correlation between pixels and the high computational cost of this operation. The pixels in the neighbouring regions are highly correlated. Still, they cannot capture correlations globally inside the image, which is why more complex image descriptors are needed (e.g., shape, edges, lines, etc.). For this reason, computer vision algorithms extract semantic information from the image content, and this information is called feature image. Feature extraction is the starting point for many computer vision algorithms. The features are, in fact, the main element for the prediction calculation. Therefore the better the quality of the features, the better the algorithm's performance will be. Furthermore, in classic machine learning, the result is highly dependent on the excellent selection and creation of the features for the algorithm. Although there are techniques such as Principal Component Analysis (PCA) to select quality features, these must still have been intelligently created by the engineer. On the other hand, with a data-driven approach, the selection of features is automated by the learning process. Thanks to a wise choice of the loss function, the model can correctly discriminate between useful features and not as long as the model's capabilities allow them to be identified and the embedding of the features is of a suitable size. Another difference between classic Machine Learning and Deep Learning is feature vector creation. In the first case, the size of the vector depends on the number of algorithms for the extraction of the features that you decide to apply (Canny Edge Detector, SUSAN, SIFT, etc ...). While in the second case, the size depends on the number of nodes present after the convolutional layers. In this case, it is always decided a priori during the construction of the model. Furthermore, in both cases, it is not guaranteed that all the features created will prove helpful in the prediction, but the model selects the most suitable ones if its capacity allows it. Each feature constitutes an element of the feature vector for both approaches. The set of vectors, therefore, comprises what is called Latent Space. This vector space is essential for the artificial intelligence model, and the decision boundaries are defined and applied in this space for classification tasks. Instead, if we are trying to solve an image generation task, we can use this space for sampling to generate new data.
One of the main challenges facing data-driven features today is the lack of explainability of deep learning algorithms. Despite these algorithms' many benefits, it cannot be easy to trace precisely which features are responsible for a given prediction. To address this issue, various techniques have been developed to visualize

the regions within an image most important to the decision made by the model. An example of this is Grad-CAM [4]. However, whether these methods provide a sufficient level of explainability is still debated. As a result, researchers continue to explore new ways to improve the interpretability of deep learning models, making it an ongoing open issue in the field.

In conclusion, not only have the features extracted from Deep Learning models shown in recent years to obtain results far superior to those obtained from classic machine learning methods, but they have also proved to be much easier to create and use from an applicational point of view. These are the reasons that build research community interest in data-driven features that push the bar of artificial intelligence further and further every day.

2.2 Fusion Strategies

Previously, we highlighted the capability of deep learning models to generate features efficiently. In this section, we will explore various techniques that can be employed to combine these features with improving the performance of a model. These techniques range from simple methods, such as concatenation or averaging, to more complex methods, such as feature fusion and selection with a dedicated loss function. These techniques can be applied to different types of data and tailored to the specific needs of a given task.

2.2.1 Direct Fusion/Early Fusion

Direct fusion [5] is the simplest way to merge features from different modes/branches. This is applied simply by addition/subtraction or merging operations directly between the embeddings, as shown in Fig 2.2. It remains one of the simplest methods but also one of the most effective. The model is left responsible for processing the merged features and deciding how to process the information to implement the correct prediction in the layers following the extraction and merging layers. Typical of this approach are multi-branch models, where each branch deals with a different input, which can be of the same or of another modality, from which it derives a semantic description represented by the feature embeddings. The representation will be subsequently merged and processed by the final layers of the model.

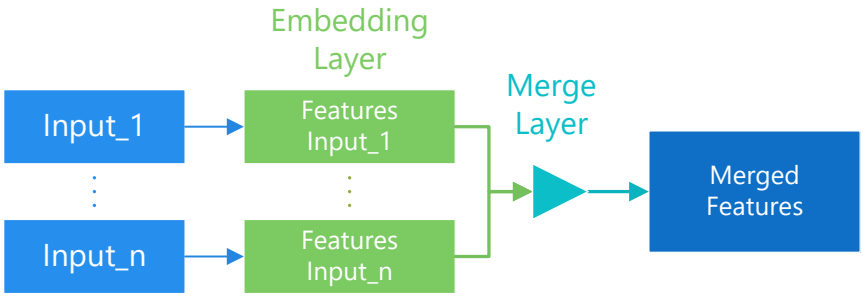


Fig. 2.2 Direct fusion template

2.2.2 Flow Fusion

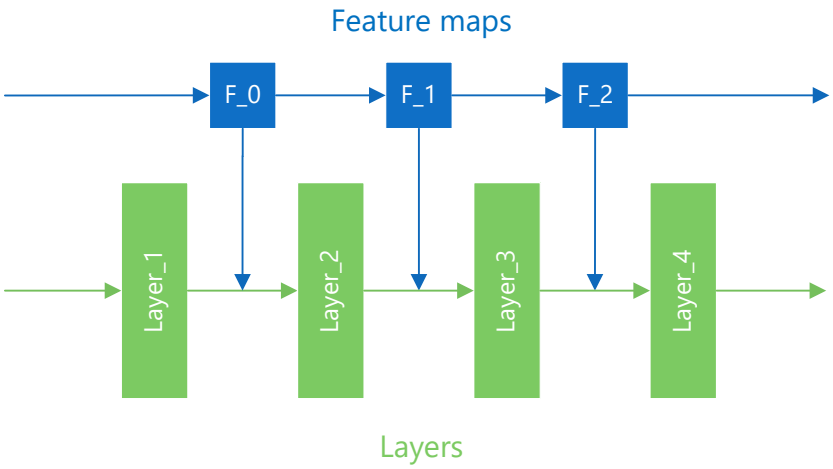


Fig. 2.3 Flow fusion template. Each box represents a feature map.

Another method is called Flow Fusion [5]. As can be guessed from the name, the work on the features is done during the flow of information through the layers. In this case, the features are continuously merged with the information of the previous layers. That technique allows obtaining a multi-scale representation of the images (from low to high-level features). Thanks to this approach, the model can have a general view of the content of an image. In Fig 2.3, we present a simple template of the concept of flow fusion. Furthermore, the continuous propagation of information helps the model avoid problems such as vanishing gradients. The vanishing gradient

problem is a common issue in deep neural networks, mainly when training recurrent neural networks (RNNs) or deep feedforward networks. In these networks, the gradients of the parameters with respect to the error function can become extremely small, making it difficult for the network to learn. This happens because the gradients are calculated using the chain rule of calculus, which involves multiplying many small numbers. As the number of layers in the network increases, the gradients become increasingly small, making it difficult for the network to learn. This is particularly problematic for RNNs, which have many layers of recurrent connections and, thus, many opportunities for the gradients to become small. One way to mitigate the vanishing gradient problem is to address the vanishing gradient problem is to use methods that combine information from multiple layers, such as Flow Fusion, such as feature fusion methods. These methods allow the network to take advantage of information from different levels of abstraction, making it less likely that the gradients will become small. The most striking example of this type of application is the Features Pyramid Network (FPN) [6]. FPN is a deep learning architecture for object detection. It combines features from multiple layers of a pre-trained backbone network to create a multi-scale representation of the input image, allowing the network to detect objects at different scales. The FPN includes a pyramid network consisting of several layers, each corresponding to a different level of abstraction, with down-sampling and up-sampling layers, lateral connections to fuse information from different levels, and a top-down pathway to propagate information from high-level features to low-level features, to improve the accuracy of object detection.

2.2.3 Adaptive Fusion

So far, we have feature fusion techniques that apply homogeneous combinations between the inputs. However, there is also the possibility of combining features in a weighted manner. In Adaptive Fusion [5], features coming from different layers or branches will contribute differently to the merging layer. This application can be helpful in multimodal contexts where, for example, it is necessary to combine various modes prioritising a particular type of content or sensor. More formally, we have:

$$UnweightedFusion \quad F_{UW}(I_1, I_2) = E_1 + E_2$$

$$AdaptiveFusion \quad F_A(I_1, I_2) = \alpha E_1 + \beta E_2$$

Where I is the input, F is the features fusion function and α, β represents the weights learned from the feature maps to be fused.

2.2.4 Fusion Operations

Several operations can be performed on features embedding from different branches of a model, each of them also has a different reason for use, the most common are presented below.

Concatenation When you want to combine features from two different modalities, the most spontaneous operation to perform is concatenation. Thanks to this operation, it is possible to keep, completely intact, the information of the different modalities. Through the learning process, it will then be the purpose of the subsequent layers to select the discriminating features for the chosen task.

Subtraction/addition Another widely used methodology is based on the combination of embedding through subtraction and addition operations. Obviously, for this to be possible, the size of the two embeddings must coincide. For this reason, addition and subtraction should be used with inputs from the same domain. An example could be the case of facial recognition, where two images of the same person's face are used as input. Once the features have been extracted, an addition/subtraction operation is then applied with the aim of accentuating/inhibiting certain features.

Element wise multiplication Another possible merger methodology is element-wise multiplication. This does not differ much for addition and subtraction purposes and requires the same premises, namely the size of the embedding and the domain of the features. The multiplication operation can be seen as a combined addition/subtraction. In fact, features with very weak activation will be automatically inhibited, and features with very high activations will be accentuated. The possible applications remain the same for addition and subtraction (Face Matching, Signature Verification, Anomaly Detection, etc ..)

2.2.5 Loss Functions and Feature Fusion Methodologies

One of the key challenges in feature fusion methodologies is to effectively combine information from multiple sources to improve the performance of a given task. Loss functions provide a way to optimize the feature representations obtained from different sources to ensure that they are complementary and that the final feature representation is optimal for the task at hand.

For example, loss functions, such as the Compactness Loss [7], help to ensure that the features from the same class are as close as possible in the feature space, meaning that the deep feature representation of the class is distinctive and the intra-class distance is low. This is particularly useful in one-class classification problems, where the goal is to differentiate between the target class and all other classes. By using the compactness loss in conjunction with other loss functions, such as the descriptive loss function, it is possible to achieve an optimal feature representation that is both distinctive and compact. Loss functions can also be used to regularize the feature representations by encouraging them to be more robust to noise and variations in the input data. This can help improve the model's generalisation performance and make it more robust to different scenarios. Another type of loss function for feature fusion is based on Adversarial loss, a technique used in Generative Adversarial Networks (GANs) to measure the difference between the generated and real data [8]. In feature fusion with adversarial loss, the goal is to generate data similar to real data in multiple feature spaces.

Overall, loss functions are a powerful tool that can improve the performance of feature fusion methodologies by optimising the feature representations obtained from different sources, ensuring that they are complementary and optimal for the task at hand.

Compactness Loss

Compactness loss is a function (referred to as $\mathscr{C}$ in the text) that is used to evaluate the distance between deep features of objects from the same class in a one-class classification problem. The goal is to have the features from the same class be as close as possible in the feature space, meaning that the deep feature representation of the class is distinctive and the intra-class distance is low. The compactness loss is used in conjunction with the descriptive loss function (the cross-entropy) denoted as

$\mathscr{D}$ to optimize the deep feature representation for the training data. The optimization problem can be formulated as follows:

$$\hat{g} = \max_{g} \mathscr{D}(g(t)) + \lambda \mathscr{C}(g(t))$$

where g is the deep feature representation for the training data t, λ is a positive constant $\mathscr{D}$ is the Descriptive loss function (within this approach, we used the cross-entropy) $\mathscr{C}$ is the Compactness loss function, which evaluates the batch inter-class deep feature distance to derive objects from the same class. The Compactness loss function, $\mathscr{C}$, evaluates the batch inter-class deep feature distance to ensure objects of the same class are close to each other in the feature space. By combining the features through the Compactness loss applied to the merging layer, it is possible to obtain the most descriptive features of the combined inputs. This function effectively enhances the representation capabilities of the deep learning model, leading to improved accuracy and robustness in its predictions. With this feature composition, the model is able to better capture the relationships between the inputs and produce more meaningful results.

Adversarial Loss

The generator's goal in a GAN is to produce data similar to the real data, as determined by the loss function. The loss function is typically a measure of the distance between the generated and real data in some feature space. One way the loss function in GANs can be used for feature fusion is by incorporating multiple loss functions that measure the difference between the generated and real data.This can help the generator learn to produce data similar to the real data in a combined feature space, leading to more realistic generated data. For example, one loss function could measure the difference between the generated and real data in the pixel space (L_1 o L_2). In contrast, another loss function could measure the distance (Wasserstein distance) in the feature space [9]. In this context, the generator learns to produce data similar to the real data in a combined multimodal feature space by training the generator to minimise the combined loss of multiple loss functions. This approach is called multimodal-GAN [10]. It has been used in many applications where multiple data views are available, like synth-to-real adaptation, image-to-image Translation, text-to-image synthesis, and many more. The combination of loss functions used in a

multimodal-GAN should be carefully chosen based on the specific characteristics of the task and inputs being used. Factors such as the type of data being generated, the desired output, and the overall goals of the model should all be taken into account when selecting the appropriate loss functions.

2.3 The Multimodal tasks

With modality, we mean the nature of the inputs provided to our model. Generally, the inputs are unimodal in a Computer Vision algorithm (raw pixels of the image). However, additional information may be obtained during the gathering process, such as a text description linked to the image caption or information from a different sensor, such as a heatmap or infrared (IR). Adding these inputs of a different modality from the RGB image makes our task a multimodal learning task.

2.3.1 Modalities fusion strategies

After defining what we mean by multimodal, we now face a further problem. How can we merge inputs from different modalities? There are usually two applicable methods: 1) direct concatenation of all inputs before using them for training and 2) late fusion of input embeddings. The first method is the simplest and allows a single model for all the inputs. The drawback is that the inputs must be all the same size, a possible solution perhaps when we want to merge RGB images with IR, but it is not feasible, for example, to combine images with text. For this reason, the second approach is more used and has several advantages. Each input has its dedicated features extraction model, which we call *branch*, and the features are then merged with the techniques and approaches seen in Chapter 2. Having a dedicated branch for each mode has advantages as each branch focuses on learning specific features for that mode. We leave the discrimination of which features to use and which not to the following layers after the block of features fusion. The only disadvantage is that the model suffers from the complexity perspective, as this approach generally requires models with multiple parameters.

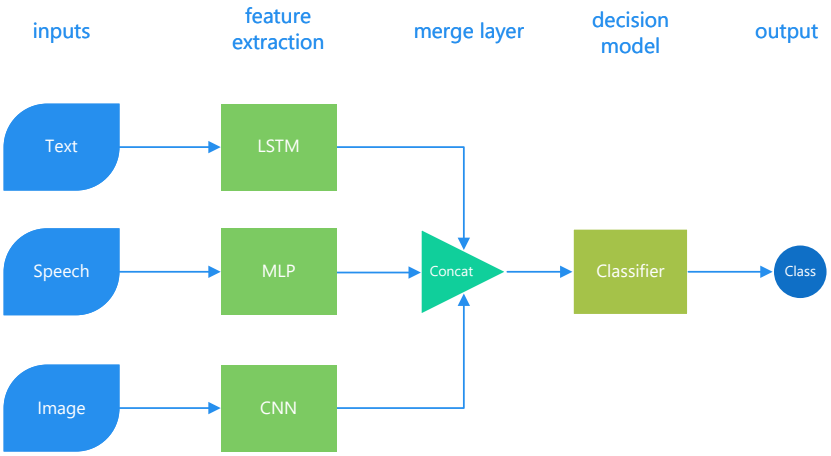


Fig. 2.4 Example of multimodal architecture

2.3.2 The template of a multi-branch architecture

Summarizing what has been defined so far, by multimodal, we mean a machine learning task where the inputs belong to different modalities (text, audio, image, etc ...). While with multi-branch, we want to indicate a model composed of n feature extractors, one for each modality. Each branch will take care of creating a specific embedding for each modality. Feature fusion methodologies will be applied to all the embeddings generated in this way. The merged embeddings will be used to resolve the machine learning task (classification, regression, detection, etc ...) from a decision model. Thanks to Deep Learning, this pipeline can be simultaneously trained end-to-end depending on the loss function defined for the task. An example of a multi-branch architecture for multimodal tasks can be seen in Fig. 2.4.

More formally, we have:

$$F_{ext}^{k}(I_k) = E_k \quad k = \{1,2,3,\ldots,n\} \tag{3.1}$$

$$E_1 \otimes \ldots \otimes E_n = E_{merged} \tag{3.2}$$

$$M_{final}(E_{merged}) = Output \tag{3.3}$$

Where: $F_{ext}^{k}(I_k)$ is the *k-th* feature extractor for the *k-th* input, E_{merged} is the result obtained from the feature fusion strategies and M_{final} is the final decisional model responsible for producing the output.

2.4 What is next

As can be shown in Table 2.1 presents an overview of the feature fusion methodologies that will be examined in the following chapters. In these chapters, we will focus on different feature fusion techniques and how they can be utilized to improve the performance of deep learning models. We will also discuss the advantages and limitations of each method, as well as their potential applications in real-world scenarios. Additionally, we will provide experimental results that demonstrate the effectiveness of these techniques in various benchmark datasets. Overall, this study aims to provide a comprehensive understanding of feature fusion methodologies and their role in deep learning.

Chapter	**Title**	**Fusion Methodology**
4	Estimation of speed and distance of surrounding vehicles from a dashboard camera point-of-view	Direct Fusion
5	AA-SGAN: adversarially augmented Social GAN with synthetic data	Adversarial Loss
7	AI-based flood event quantification using online media and satellite data	Direct Fusion
8	Road passability detection during flood events using social media data	Compactness Loss
9	Emergency scene description using deep learning approaches	Direct Fusion
10	Water segmentation for flood detection and monitoring	Flow Fusion

Table 2.1 Overview of feature fusion methodologies examined in the following chapter

Part I

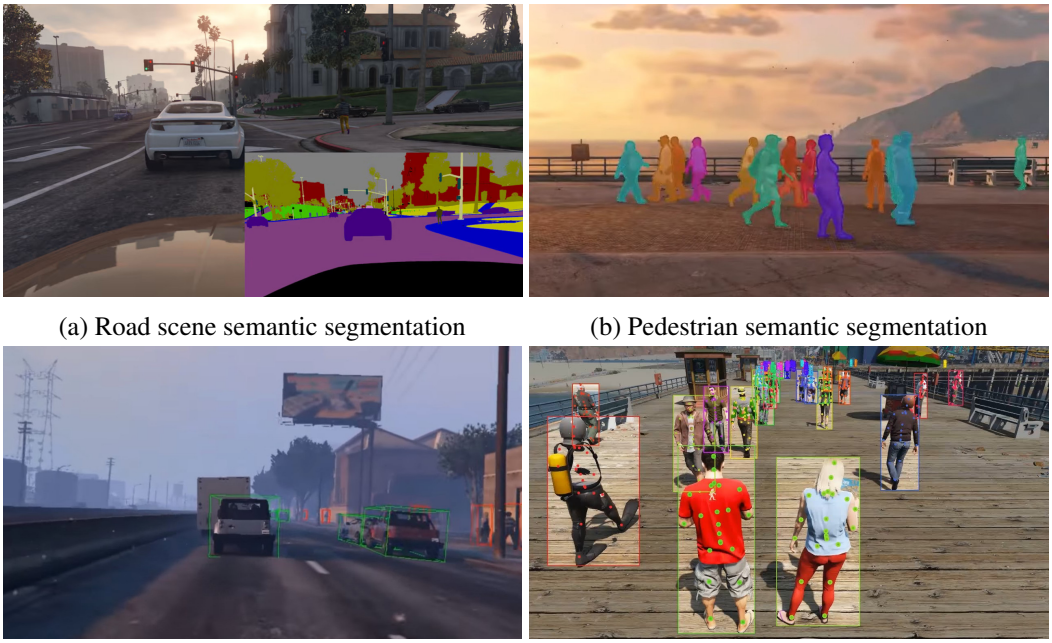
Automotive

Chapter 3

Automotive and synthetic datasets

The automotive industry is experiencing a rapid transformation with the advent of new technologies such as autonomous vehicles, advanced driver-assistance systems (ADAS), and connected cars. These technologies rely heavily on computer vision, responsible for perceiving the environment, understanding the context, and making decisions. One of the key challenges in developing these technologies is the need for large, labelled datasets to train and evaluate the performance of computer vision models. However, collecting and labelling large datasets of real-world data is costly, time-consuming, and may not always be feasible. Computer graphics simulation is emerging as a powerful alternative to real-world data collection, providing a large-sized dataset quickly and with inexpensive resources and allowing the generation of diverse scenarios with different weather and lighting conditions. This makes it a valuable tool for the automotive industry to train and evaluate the performance of computer vision models in the automotive field.

As highlighted, the availability of many indexed and labelled images is key to successfully designing complex vision tasks leveraging powerful DL techniques based on CNNs. Creating a large dataset to represent the target scenario correctly and allowing the trained neural network to generalize in real applications remains a critical design step. Resorting to human visual inspection and manual labelling is not viable in many scenarios. Manual labelling does not scale very well to large datasets, except for simple and repetitive tasks that do not require particular expertise, where one can resort to crowd-sourcing [11]. Moreover, doubts about the collected information's quality and potential unexpected bias may arise. Finally, manual



(a) Road scene semantic segmentation

(b) Pedestrian semantic segmentation

(c) Cars 3D bounding boxes

(d) Pedestrians bounding boxes and body-joints

Fig. 3.1 Examples of synthetic datasets created with the aim of GTAV

labelling is impossible for some tasks, as in the automotive scenario targeted in this work, where physical quantities such as distance and speed must be estimated from images. One option in the automotive field is to use special vehicles with ad-hoc, usually expensive, settings and sensors capable of gathering the information required for training. Such real experiments' setup and maintenance costs may represent a significant barrier. In this context, computer graphics (CG) simulation is emerging as a powerful source of visual information. CG allows for obtaining a large-sized dataset in a short time and with the usage of cheap resources [12]. In addition, modern video games are getting closer and closer to photorealism, thus promising to bridge the gap between visual simulation and reality, which is likely to be the key to training computer vision systems that are effective in real life. Moreover, simulation makes experimental and environmental settings more flexible: i.e., in the automotive field, datasets with heterogeneous driving scenarios can be generated and subjected to different weather and lighting conditions. Higher heterogeneity can significantly improve the trained model's robustness and generalization. As an example, by using a simple 3D rendering technique such as Ray-casting in a virtual environment, one can get a simulation of a LIDAR scanner [13, 14] quickly obtaining information on the distance of the elements within the image. Furthermore,

it is possible to get data that are usually difficult to obtain, such as measurements of the speed of all the surrounding vehicles, which would require a complex setup in the real field. To customize and generate a dataset for visual training, one needs to design a complex CG simulation environment or exploit existing high-quality game engines. The second option is viable if one has access to the source code to quickly extract information on the entities and the elements that make up the gaming environment (bounding box, size, distance from the observer, type of entity, etc...). Nowadays, few open-source simulators can be used to extract synthetic datasets. In the automotive environment, TORCS [15] can be used; however, this tool allows the representation of only a few scenarios with limited photorealism. On the other hand, commercial car video games run very realistic CG and are equipped with intelligent agents to simulate entity actions, e.g. a pedestrian walking. For this reason, the research community has recently got interested in Grand Theft Auto V (GTAV) [16–19], a popular open world videogame that, thanks to the libraries developed by third parties, allows one to extract data from the gaming environment. One of the main advantages of using a game engine like GTAV is the high degree of realism and photorealism that it can provide [16–19]. Additionally, the game's open-world environment allows for a wide variety of scenarios, such as different weather conditions and different types of roads, which can help to improve the robustness and generalization of the trained model. However, it's important to note that the generated synthetic data should be carefully evaluated for its realism and ability to improve the model's performance in real-world scenarios [18]. Combining synthetic data with real-world data through domain adaptation or domain generalization techniques is essential to overcome the lack of realism in synthetic data.

In conclusion, computer graphics simulation, specifically using game engines like Grand Theft Auto V, can provide a powerful source of visual information for training computer vision systems. The high degree of realism and photorealism of these synthetic data, combined with the ability to generate a large-sized dataset quickly and with inexpensive resources, makes it a valuable alternative to manual labelling. However, it's important to carefully evaluate the realism of the synthetic data and combine it with real-world data to improve the model's performance in real-world scenarios.

Chapter 4

Estimation of speed and distance of surrounding vehicles from a dashboard camera point-of-view

In this chapter, we focus on computer vision aids for automotive applications and target to estimate the distance and speed of the surrounding vehicles using a single dashboard camera. We propose two multi-branch network models for distance and speed estimation, respectively. In the first model presented, spatial features are merged with semantic features to obtain an estimate of the vehicle, while in the second, the semantic features are replaced by motion features, represented by the optical flow of the scene, to obtain an estimate of the vehicle speed. For both approaches, feature fusion takes place via direct fusion. Moreover, we show that training them by using synthetic images generated by a game engine is a viable solution that turns out to be very effective in real settings.

4.1 Introduction

In a similar fashion to work done in [17], in this chapter, we propose a work based on two CNN architectures to estimate the distance and speed of the surrounding vehicles from a single camera with windshield view (see Fig. 4.1). Training has been achieved with GTAV simulations and performance validated in real life. The main contributions of this work lie in the following:

- a DL model that uses a pre-trained deep CNN to extract semantic features from vehicles images and uses them to predict the distance of the surrounding vehicles from a single camera with windshield view;
- a model which uses optical-flow information to predict the speed of surrounding vehicles from pixel motion between pairs of video frames.
- a training framework based on videogame simulation for the creation of training and testing datasets in the automotive field;
- we show that training with synthetic but photorealistic images represents a viable alternative to more expensive experimental data collection, with promising results in the estimation of distance and speed of the vehicles on the road observed with a single camera.

Dataset, code, and pre-trained model are publicly available and can be found at: *https://github.com/mirkozaff/DeepGTAPrediction*.

4.2 Methodology

In this section, we will introduce the framework used to collect data, preprocess them, and the models used to accomplish the vision task.

4.2.1 Data collection

Data have been collected from GTAV thanks to Script Hook V library (SHL), which allows access to GTAV native function easily and extract information about the entities (vehicles) from the game environment. Images and corresponding information have been generated by configuring an in-game agent that drives a vehicle and lets it wander the streets; during the simulation, one can collect the required information by querying the game engine through SHL calls. For our goal, we built a dataset by collecting, for every vehicle in the range of 30 metres from the player, the following items:

- *Frame:* 1920×1080 image captured at 30 Hz, gathered by setting the in-game camera on the dashboard.

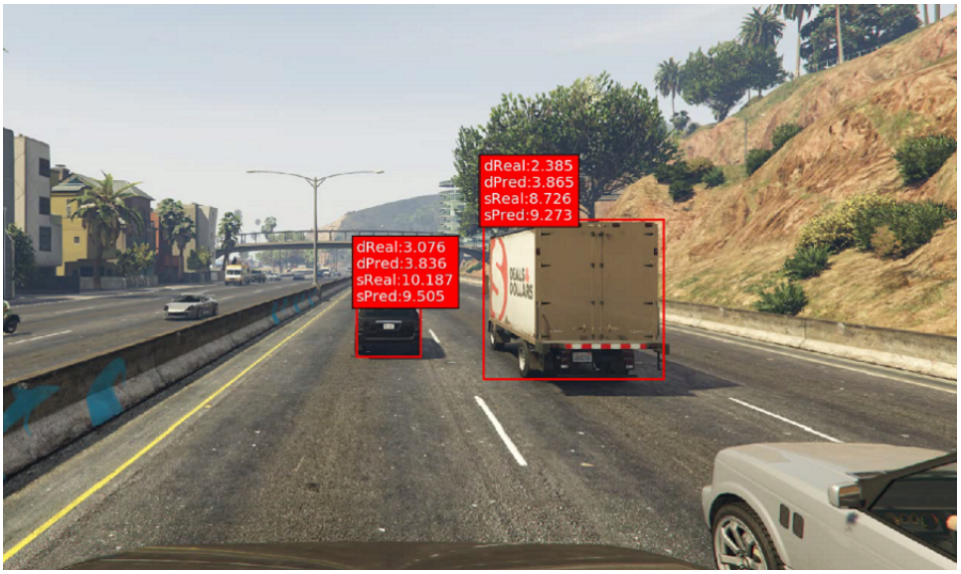


Fig. 4.1 Photorealistic frame extracted from the game environment, featuring an overlay of the prediction

- *Entity ID:* identifier of the vehicle to track it in multiple frames.
- *Entity speed, distance:* speed and distance of the vehicle.
- *Entity bounding box:* pair of coordinates that define the *bounding box* B_1 of the vehicle in the captured frame; this is computed by projecting the 3D bounding box obtained from the game engine into 2D screen coordinates.

We noted that the bounding boxes extracted by SHL are often inaccurate and present a drift caused by the delay in response to each SHL query. To get precise bounding boxes, we use the pre-trained Mask R-CNN [20] model to detect each vehicle in the dumped frame (the same model will be used in the testing phase on real-life images). For each detected vehicle Mask R-CNN output a bounding box B_2. To univocally map B_2 onto previously computed B_1 (and corresponding speed and distance data) we set a threshold on the intersection over union $IoU = \frac{B_1 \cap B_2}{B_1 \cup B_2}$. Only the entities showing $IoU \geq 0.7$ are included in the dataset. The selected threshold also filters out some vehicles that cannot be reliably detected due to poor visual conditions.

Dataset in numbers: we used a dataset consisting of ~250,000 frames captured under various weather conditions. The dataset included a total of 10 different vehicle

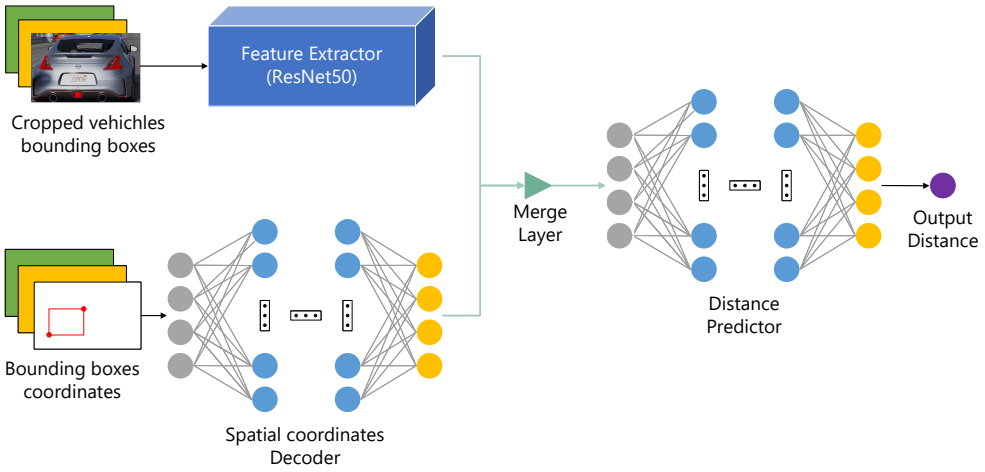


Fig. 4.2 Prototype architecture of our proposed distance model. A CNN extracts semantic features of the cropped vehicles, while the MLP branch is used to learn a spatial representation of the coordinates. Then all this information is merged and decoded into output values through the last MLP.

models and 5 different weather settings, including sunny, cloudy, rainy, snowy, and foggy. GTAV features 348 different vehicles. We only considered frames that contained at least one car, resulting in a dataset of over one million samples of cars.

4.2.2 Models

Distance

The first model we present is designed to estimate the distance from surrounding vehicles using a single camera with a windshield view. The input is a single image from which vehicle bounding boxes are detected, e.g. by using Mask R-CNN.

As shown in Fig. 4.2 the architecture is composed of two branches:

- *First branch:* pre-trained ResNet50 [21] used to extract semantic features of the target vehicle from the corresponding frame. To this end, the classification layers in ResNet50 are removed. The input is a frame crop based on the bounding box B_2. In particular, each vehicle is extracted from the frame by cropping and resizing it at 224×224, which is the input resolution expected by ResNet50.

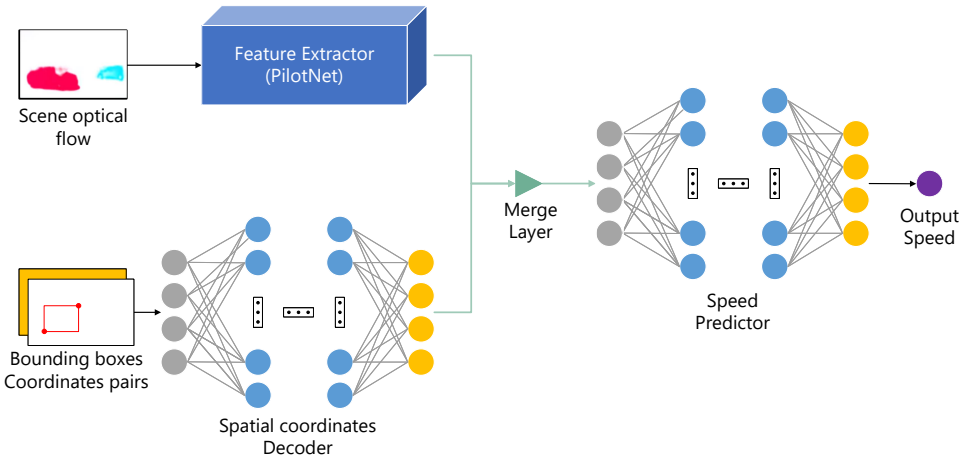


Fig. 4.3 Prototype architecture of our proposed speed model. A CNN extracts optical flow features, while the MLP branch is used to learn a spatial representation of the coordinates pairs of two subsequent frames. Then all this information is merged and decoded into output values through the last MLP.

- *Second branch:* Multi-Layer Perceptron (MLP), used to encode the coordinate of the bounding box B_2 into a higher multidimensional space. We selected the *ELU* activation function [22] for each layer, in place of the classic *ReLU*; we noted that in our scenario *ELU* is very effective in avoiding the dead-neuron problem [23].

These two branches are then concatenated and processed through a final MLP responsible for predicting distance from the fused information produced by image pixels and bounding box coordinates. Semantic features provided by the first branch are important because vehicles appearing at the same scale in the image may represent different classes of an object; clearly, we cannot base the distance estimation on the sole geometric information, i.e., bounding box dimension and position analyzed by the second branch. In other words, as in real life, we must consider that cars are smaller than trucks when guessing the corresponding distance.

Speed

The model proposed to estimate the speed of surrounding vehicles is designed with a similar approach using two branches: (i) semantic based on images, (ii) geometrical based on bounding boxes. When speed is regarded, one has to consider at least two

consecutive images to gather object displacement over time. One option would be to process frames directly. In this chapter, we propose to use as input the *Optical Flow* (OF) estimated from the current (and previous) frame under analysis. OF is a dense vector field representing every pixel's displacement, e.g., computed using the Farneback method [24]. Moreover, we use the two bounding boxes of the same vehicle tracked in two consecutive frames (tracking is obtained in our GTAV dataset using entity IDs, while it will require additional processing in real setting). The structure of the model for speed estimation is as follows:

- *First branch:* PilotNet [25], a CNN proposed to learn salient points of the road for autonomous driving, is used to process the input OF. The OF vector fields are represented as an image with two bands representing vector magnitude and direction, respectively. The obtained OF image is cropped according to B_2 and resized to 200×66 (the resolution expected by PilotNet). We improved the original PilotNet model by adding batch normalization to the convolutional layer in order to speed up convergence and using ELU activation function and *Dropout* on the last fully-connected layers.
- *Second branch:* same MLP structure used in the previous model to encode in a higher multidimensional space the coordinates of bounding boxes; differently from the distance estimation model, now the input is represented by two bounding boxes associated with the same vehicle tracked in two successive frames.

Finally, the features extracted on the two branches are concatenated and processed by the final MLP as shown in Fig.4.2 to estimate speed. We adopted a heuristic similar to the one proposed for distance: the lower branch extracts motion features based on bounding box geometrical information and displacement; the PiloNet branch encodes richer features that depend on the OF of all the pixels corresponding to a vehicle and potentially also extract semantic characteristics.

4.3 Network training

Using the process presented in Sect. 4.2.1 it is possible to generate datasets comprising as many vehicles, labelled with distance and speed, as desired. We employ

training and validation sets with 250,000 and 2,500 samples to train the distance model in this work. Regarding the speed model, we extract from the previous dataset all vehicles visible in two consecutive frames generating a set of 180,000 OF images for training and 1,500 for validation.

The proposed models have been trained using *Mean Squared Error* (MSE) as loss function and *Adam* optimizer with the following parameters: $lr = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$. The training was stopped as soon as the loss computed on the validation set ceased to decrease to avoid over-fitting; in our experiments, this usually occurred after about 15 training epochs.

Training all MLP sub-networks has been done using *Dropout* with parameter $p = 0.4$. In the distance model, ResNet50 weights pre-trained on *ImageNet* have been kept fixed while optimizing only the other MLP sub-networks. In the speed model, all the network has been trained since no pre-trained PilotNet useful in our context was already available.

Training has been run on a PC with Intel(R) Core(TM) i9-7940X CPU, 128 GB RAM and NVIDIA GeForce GTX 1080 Ti (x4). Testing was performed on the same machine and a lighter one with Intel(R) Core(TM) i5-6400, 8GB RAM, and NVIDIA GeForce GTX 1050 Ti. This latter has been selected as representative of the hardware one expects to have on board the vehicle, as opposed to the previous higher-end server.

4.4 Experimental results

This section describes the experimental results obtained in different simulated and real settings.

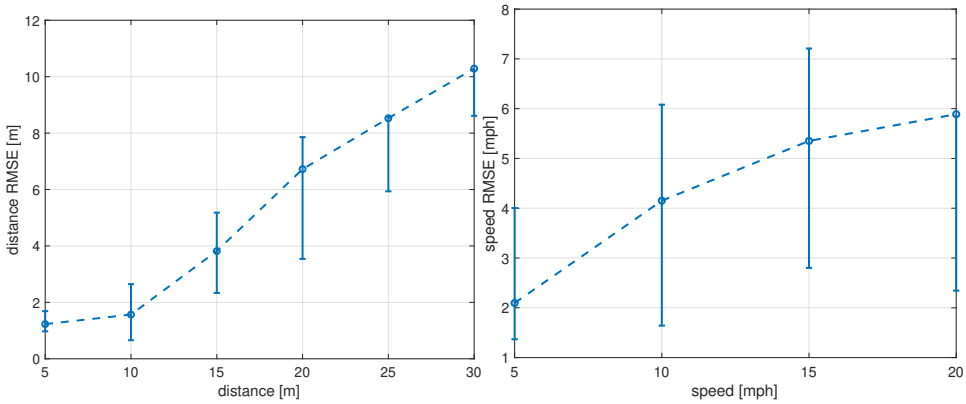
4.4.1 Testing on synthetic dataset

As a first step, the estimation accuracy of the trained models has been evaluated on synthetic datasets of size 2,500 and 1,800 for distance and speed, respectively. These testing sets have been generated using the GTAV simulation described in Sect. 4.2.1. It is worth pointing out that training and testing sets have been generated with different random simulations to make them independent.

The proposed models are able to predict distance with a Root MSE (RMSE) of about **2.46** [m] and speed with RMSE of about **2.75** [mph]. In Fig. 4.1 we provide an example of the obtained visual results. The image shows a car and a truck with labels representing ground truth and predicted distance and speed. For the car, the model predicts a distance of 3.8 m versus a real value of 3 m and 9.5 mph speed versus 10.2 mph.

In Fig. 4.4, we analyze in more detail the estimation accuracy. In particular, Fig. 4.4a shows the RMSE on distance as a function of the actual distance range; to this end, we compute RMSE (the circle marker) and standard deviation of the estimation error (vertical bars) by binning the collected results in increasing distance ranges of 5 m in the interval $(0,30)$ m (the top error bar indicates an overestimate, whereas the bottom segment represents an underestimate). It can be noted that, as one may expect, the RMSE increases for more considerable distances. Overall the distance estimates are pretty accurate and unbiased (almost symmetric error bars) within a range of 15 m: as an example, the RMSE in the range $(0,5)$ m is 1.23 m, and in the range $(5,10)$ m is 1.57 m. For farther vehicles, the predictions are less accurate, and the model underestimates the distance. This can be explained by the fact that at distances greater than 15 m, vehicles are represented in the image by fewer pixels limiting the information extraction capabilities of the convolutional layers.

In Fig. 4.4b we show a similar RMSE analysis on the speed estimate as a function of the speed up to 20 mph, which is the maximum value that can be simulated in



(a) (b)

Fig. 4.4 RMSE on distance (a) and speed (b) estimate.

GTAV. It can be noted that speed RMSE increases as a function of speed. The results show that the proposed network can reasonably guess the speed of the surrounding vehicles by using a single-camera view. As an example, in the speed range $(0,5)$ mph, we get RMSE equal to 2.10 mph, and in the range $(5,10)$ mph, we get RMSE equal to 4.15 mph. We expect to be able to improve such results by increasing the number of temporal frames analyzed by the model and using better OF representations.

4.4.2 Testing on real dataset

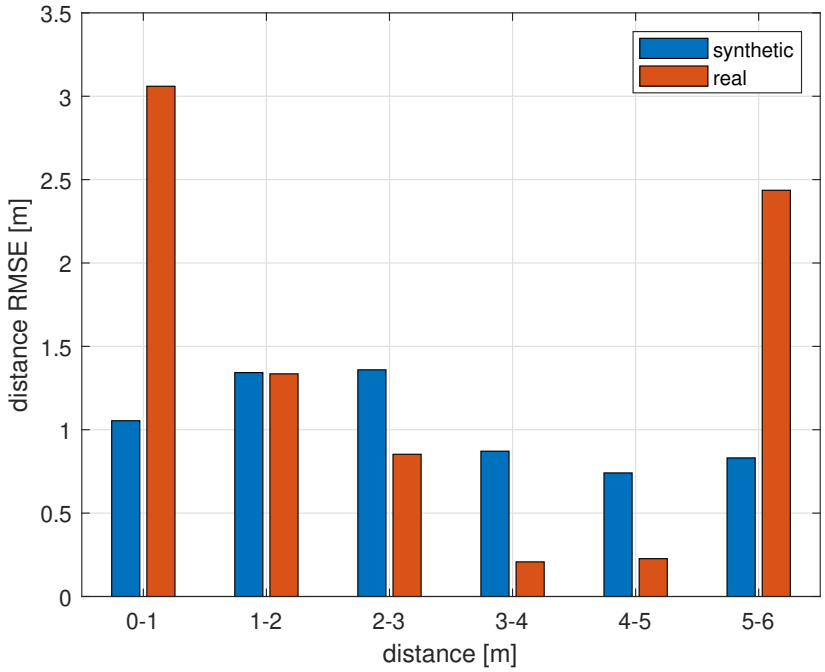


Fig. 4.5 RMSE on distance estimate (synthetic and real images).

As already mentioned in Sect. 4.1 one of the goals of this work is to understand if CG simulation can be used to effectively train DL models employed in real settings. To answer this question, we need vehicle videos with annotated data. To this end, we used the video sequence provided in [26] and corresponding distance estimates as an example of a real dataset. In Fig. 4.5 we compare the RMSE on distance prediction obtained on the real and synthetic datasets subdivided in 2 m ranges (please note that images from [26] are limited to a 6 m range). It can be noted that the proposed model is quite robust and generalizes well in a real-life scenario, even if the actual

environment can be significantly different with respect to GTAV simulation. Indeed it can be noted that, in the experimented distance range, the RMSE of the real dataset increases by less than 0.5 m with respect to the synthetic testing set. Overall the test RMSE was **1.21** on synthetic data and **1.40** on real data. We want to perform a similar experiment with speed prediction, but unfortunately, to the best of our knowledge, no publicly available dataset can be employed to this end. Indeed, the setup of real road experimentation is quite complex.

4.4.3 Testing on the road

Fig. 4.6 Example of distance estimate in real environment.

Finally, the model has been tested on a video recorded on the streets around our city using a Go Pro Hero 6 placed on the car dashboard. This last experiment was accomplished in real road environments (both urban and highway) to check the meaningfulness of the obtained predictions. In this case, we do not have ground truth data. By analyzing the proposed system's operations in real life, we noted that the predicted distances are plausible and coherent, i.e., vehicles appearing at the same distance are assigned the same value, and approaching vehicles exhibit decreasing distance. As in the previous experiment also, in this case, the model performance is not significantly impaired by the road environment; that is very much different with respect the GTAV scenario.

Work Station	
290 ms	Bounding box
15 ms	Distance
45 ms	Speed
120 ms	Latency
500 ms	Total
On Board PC	
2 sec	Bounding box
280 ms	Distance
880 ms	Speed
3 sec	Total

Table 4.1 Execution time of different algorithmic steps.

4.4.4 Computational cost

In this section, we analyze the computational cost of the proposed solutions by measuring the execution time of different algorithmic steps of the two hardware architectures described in Sect.4.3; these are meant to be representative of a workstation performing remote computation and lower-end hardware compatible with the in-vehicle system. In Tab. 4.1, we show the average time taken by the bounding box calculation, speed, and distance estimate for an image with five vehicles (on average). To get acceptable delays (compatible with real-time requirements of advanced driver-assistance systems), it is necessary to use a powerful workstation. As expected, the bounding box computation, so the distance estimate represents the slowest module.

4.5 Conclusions

In this chapter, we proposed two models to accomplish two different tasks: speed and distance prediction using a single camera looking at the road from the driver's perspective. Since, for such tasks, it is either technically difficult or quite expensive to get real video sequences for training CNNs, we proposed using simulated data generated through a popular game engine. Such an approach allowed us to collect photorealistic driving scenes, where all the visible vehicles can be labelled with distance and speed information. We designed two DL models built around similar ideas: one branch extracts feature from the input images, a second maps vehicles' bounding boxes (dimension and position) to higher dimensional space, and the last

MLP network infers distance or speed from all the extracted features. Estimating accuracy has been evaluated on both synthetic and real data showing that the simulated images can train the proposed models effectively. For future work, we plan to improve the models by substituting the bounding box detection network with a lighter version and using a DL approach to estimate OF for speed prediction. Moreover, we plan to enrich the input available to the network by including parameters that can be logged on board a car, such as throttle, brake, and steering data, to mention a few.

4.6 Acknowledgement

The research leading to these results has received funding from the European Union Horizon 2020 research and innovation programme under grant agreement No 713788 ("optiTruck" project).

Chapter 5

AA-SGAN: adversarially augmented Social GAN with synthetic data

Accurately predicting pedestrian trajectories is crucial in applications such as autonomous driving or service robotics. Deep generative models achieve top performance in this task, assuming enough labelled trajectories are available for training. To this end, many synthetically generated, labelled trajectories exist (e.g., generated by video games). However, such trajectories are not meant to represent pedestrian motion realistically and are ineffective at training a predictive model. We propose a method and an architecture to augment synthetic trajectories at training time and with an adversarial approach. In addition to the standard adversarial loss, our proposed method also adds "ad hoc" loss functions responsible for merging the features of the real and synthetic trajectories in a common feature space. This allows the model to learn better from real-world and synthetic data, resulting in improved predictions. Our proposed adversarial loss is specifically designed to address the challenges of merging the two different modalities, real and synthetic, and to achieve better performance in terms of predictions. We show that trajectory augmentation at training time unleashes significant gains when a state-of-the-art generative model is evaluated over real-world trajectories.

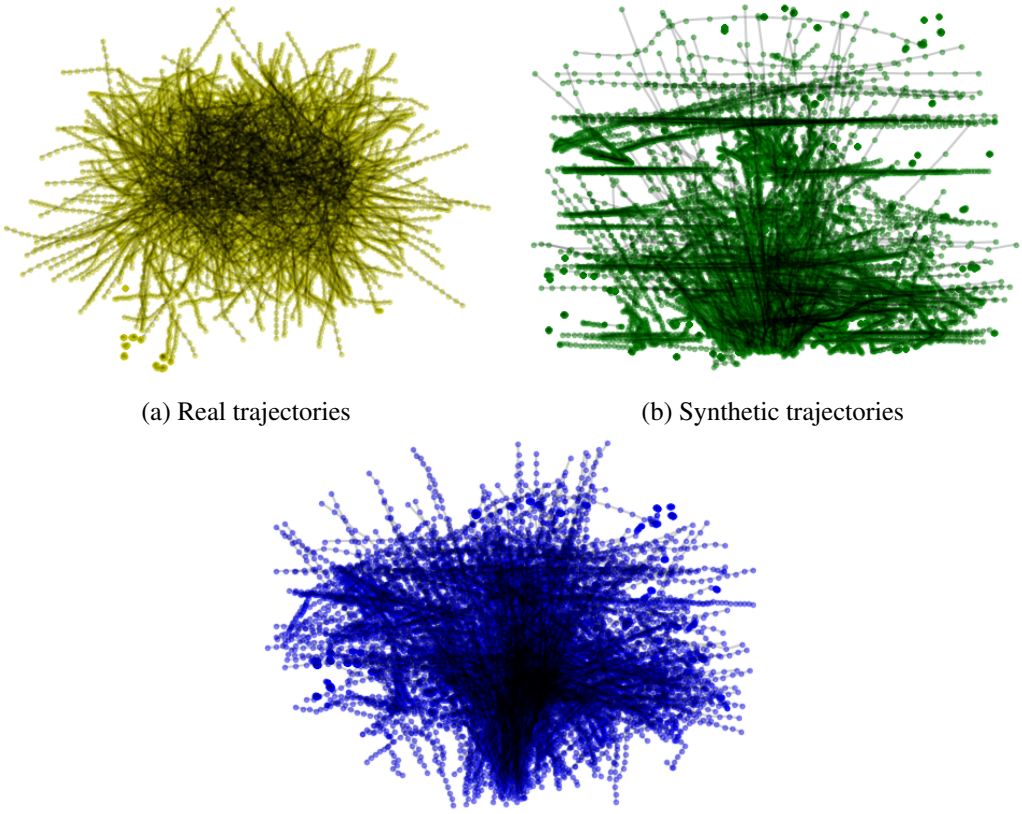
5.1 Introduction

Predicting pedestrian trajectories is of paramount importance in applications where robots must dodge humans, e.g., to avoid collisions, as in the automotive field or robot-human interaction environments. This task is inherently challenging due to the complexity and, to some extent, the unpredictability of human movement patterns. In detail, the task has three main technical challenges. First, given a partial trajectory, multiple options for its continuation are possible, making the problem intrinsically multivariate; second, an individual's motion is conditioned by bystanders' behaviours, especially in crowded spaces; finally, different social and cultural contexts may constrain what can be considered a plausible motion pattern.

Recently, deep generative models showed promising results in plausible human motion prediction. For example, Social GAN [27] trains a trajectories Generator using an adversarial approach where ad-hoc architectural elements and loss terms promote trajectories that are *socially* plausible. This model is trained on pedestrian trajectories extracted from crowds footages from different environments [28, 29]; in Fig. 5.1a sample trajectories extracted from a set of tracked pedestrians in a video scene are shown. However, such datasets exhibit limitations such as no camera settings variability and yield comparatively few trajectories compared to the capacity of some deep models. While collecting more videos to enlarge the training set may bring benefits, data collection and annotation is a daunting, time-consuming activity. Also, releasing datasets of real footage poses some privacy-related issues.

Synthetic training allows for large training sets without the burden of manually collecting and labelling the samples. For example, the Joint Track Auto Dataset (JTA) [30] provides a large body of pedestrian trajectories from the videogame Grand Theft Auto V. However, such computer-generated trajectories lack the realism of real human motion: as an example in Fig. 5.1b many deterministic straight patterns present in the JTA dataset are visually evident. As a result, we experimentally show that such trajectories cannot be directly used to improve learning for path prediction, motivating the present research.

This work proposes AA-SGAN (Adversarially Augmented Social GAN), an end-to-end adversarial approach for predicting trajectories over a combination of real and synthetic trajectories in input. In a nutshell, we introduce a generative *Augmenter* to manipulate synthetic trajectories. Such *synth-augmented* trajectories,



(a) Real trajectories

(b) Synthetic trajectories

(c) Synth-augmented trajectories

Fig. 5.1 Examples of real (a), synthetic (b), and synth-augmented (c) trajectories by the AA-SGAN Augmenter.

interleaved with real ones, are input into a Generator that learns to predict a trajectory continuation as in [27]. Notably, the whole architecture is trained end-to-end, propagating the gradient of an adversarial loss from the Discriminator back towards the Generator and the Augmenter. As a result, the Generator improves its prediction accuracy while, at the same time, the Augmenter learns how to increase the diversity of synthetic trajectories. As a qualitative example, the Augmenter modifies the synthetic trajectories shown in Fig. 5.1b to those reported in Fig. 5.1c that bear more resemblance to real ones. Our experiments over real test sets [28, 29] confirm that our method yields significant gains over a reference SGAN architecture trained either over real trajectories only or a hybrid of real and synthetic trajectories.

The chapter is organized as follows: Section 5.2 introduces relevant background. Section 5.3 illustrates our proposed methodology, while Section 5.4 presents our experimental evidence. Finally, Section 5.5 draws the conclusions of our work.

5.2 Background and Related Works

In this section, we present the background relevant to the understanding of this work. Namely, we discuss existing approaches to pedestrian path prediction and the datasets most commonly used for this task.

5.2.1 Trajectory Prediction Methods

Over the years, pedestrian trajectory prediction has been the subject of many endeavours [31]. Early attempts tried to model this complex task with models borrowed from classic Physics [32–34]. Recently, however, learning-based models have outperformed such early approaches and represent the state-of-the-art in the field. Therefore, our literature review will be limited to learning-based approaches. While some based methods rely on multimodal inputs (e.g., video beside trajectories) to boost performance, the present work relies on unimodal trajectories-based inputs, so our review will also be constrained to such cases.

Given the sequential nature of predicting a future trajectory based on past observations, Recurrent Neural Networks (RNNs) based methods were among the first to yield promising results among learning-based methods. Namely, [35] proposes to rely on Long Short Term Memory (LSTM) RNNs enhanced by an ad-hoc *social pooling* layer. This layer models the interactions between nearby pedestrians. It is responsible for guaranteeing that the predicted trajectories are also plausible from a social perspective (e.g., paths should not interfere at the same time, social conventions such as keeping the right way should be respected, etc.). While modern competitors usually outperform the above architecture, some key ideas were borrowed and put to profit in later, adversarial learning-based approaches.

Social GAN (shortly, *SGAN*) [27] approaches trajectory prediction with an adversarial approach [8]. Figure 5.2 shows the internals of the SGAN: the *Generator* receives in input the first 8 steps (observed trajectory, "*obs*") of a 20 steps long pedestrian trajectory and predicts the following 12 steps (predicted trajectory, "*pred*") of the sequence. Within the Generator, both the encoder and the decoder are implemented as recurrent LSTM networks, connected by a social pooling layer. The *Discriminator* receives in input the 12 steps sequence output by the Generator and the corresponding ground truth sub-sequence of the same length. SGAN is trained

with an adversarial approach where the Generator learns to produce real-looking path traces, whereas the Discriminator learns to tell generated from real trajectories. The architecture is trained to minimize a novel variety loss that encourages diversity among generated predictions. Other works using GAN [36] or synthetic data have been proposed [37, 38]. However, these also integrate visual information into their decision pipeline. We will not deal with these works as our approach is based on using trajectories alone as input to the model. This approach outperforms prior work in terms of accuracy, variety, collision avoidance, and computational complexity.

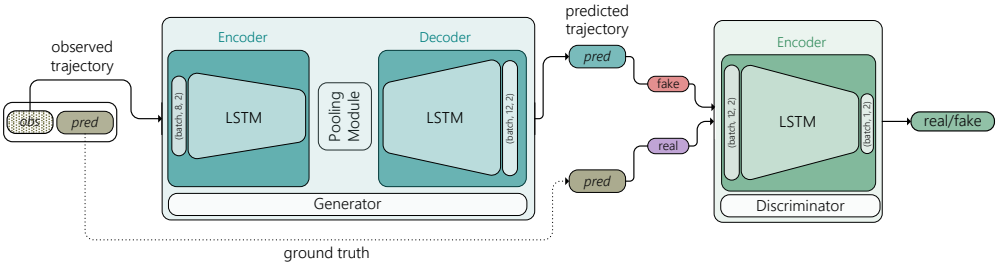


Fig. 5.2 *Social GAN* for pedestrian trajectories prediction [27] adversarial architecture. The Generator learns to generate real-looking trajectories, the Discriminator learns to tell real from generated trajectories. A social pooling module and an ad-hoc loss function enforce the social plausibility of generated trajectories.

5.2.2 Real-World Datasets

Common datasets used for pedestrian trajectory prediction include ETH [28], and UCY [29]. These datasets consist of trajectories extracted from surveillance camera videos and annotated every 0.4 seconds. Each dataset includes multiple trajectories and each trajectory includes multiple pedestrians. Each dataset sample includes a temporal index and, for each pedestrian, the pedestrian's identification code and its position in an *(x,y)*-plane. The ETH dataset includes two different environments called `biwi_eth` and `biwi_hotel`, whereas the UCY dataset includes five different environments called `crowds_zara01`, `crowds_zara02`, `crowds_zara03`, `students001`, `students003` and `uni_examples`. These datasets include a total of 2,205 frames and 6,441 pedestrians. These datasets include crowded environments with challenging scenarios such as group behaviour, people crossing each other,

avoiding collisions, and groups forming and dispersing. Therefore, these datasets are commonly employed for researching pedestrian trajectory prediction.

5.2.3 Data augmentation and synthetic datasets

Classical data augmentation, which is based on the generation of additional data, has become quite standard for visual data where manually designed transformations, e.g. crop, colour jitter, rotation, etc., can be applied to images in order to increase the variability in acquisition settings, thus promoting better generalization on real data. GANs have recently attracted lots of attention in order to overcome the limits of manual augmentation through the direct synthesis of new images. Nonetheless, GAN models need to be trained on real data as well, thus limiting their applicability in many cases. Other works have focused on automating data augmentation policies: see as an example [39] and reference therein.

Data augmentation for path prediction can be very critical: also the authors in [27] reported that synthetic data could potentially lead to worse performance. Therefore, the use of simulated trajectories in this context, while very promising, remains an open issue.

One attempt is based on creating a synthetic dataset starting from real trajectories. This is achieved by randomly sampling a trajectory from real data, adding a small perturbation with a translation, reverting the path by flipping the starting and ending points, and truncating a random number of steps. This process creates synthetic data highly dependent on real data that preserves many of its characteristics, making the crafted trajectories not suitable for proper augmentation [40]. Another work in this direction is based on modelling the underlying physiological, and psychological factors that affect pedestrian movement with agents [41]. This attempts to develop an algorithm based on density-dependent filters to generate human-like crowd flows. The approach borrows deeply from the physical model based on reciprocal velocity obstacles and social forces to create synthetic data.

Promising results were obtained using the synthetic dataset for visual tasks, e.g. pedestrian tracking [42]. Extending this approach to path prediction can be very critical: synthetic trajectories are likely to be simplified and too predictable, e.g., due to game engine scripting. Although they are suitable for learning simple path prediction models, they fail to mimic human behaviour correctly. Synthetic

trajectories poorly represent avoidance paths that people typically and unconsciously adopt when walking with others around. To overcome such limitations in this work we propose to use synthetic data as input to a new Augmenter module that is trained end-to-end along with the path prediction task.

5.3 Proposed Method

In this section, we describe our proposed approach towards learning to predict pedestrian trajectories from both real and augmented synthetic trajectories.

5.3.1 Problem definition

Let us define a pedestrian *trajectory* as a sequence of t_{pred} samples in the temporal order. Each sample is a pair of coordinates in space, where each element $(x_t, y_t)^{(i)}$ ($i = \{1, \ldots, N\}$) represents the position of the i-th pedestrian at time-instant $t \in [1, t_{pred}]$.

We have that $t = \{1, \ldots, t_{obs}, t_{obs}+1, \ldots, t_{pred}\}$, where $t = t_{obs}$ is the number of *observed* samples and $(t_{pred} - t_{obs})$ is the number of following sample to be predicted. As a common practice in the related literature, all the trajectories coordinates are preliminarily normalized to relative coordinates with respect to the starting point.

5.3.2 Architecture

Our proposed AA-SGAN architecture is shown in Fig. 5.3. As for [27], real trajectories r are fed in input to a Generator G, whose task it to predict the future samples. The Generator receives as input the first t_{obs} samples of a trajectory and predicts the next $(t_{pred} - t_{obs})$ samples. However, in the proposed architecture, the Generator receives in input also *synth-augmented* trajectories a. Synth-augmented trajectories are generated by an *Augmenter A* that receives in input a synthetic trajectory s and outputs an augmented trajectory a. Therefore, our Generator learns on a larger variety of inputs than only real trajectories as in [27]. As for [27], a Discriminator D attempts to discriminate if a trajectory is real or it is fake. However, our Discriminator is fed with three different classes of fake trajectories, i.e. it is given the

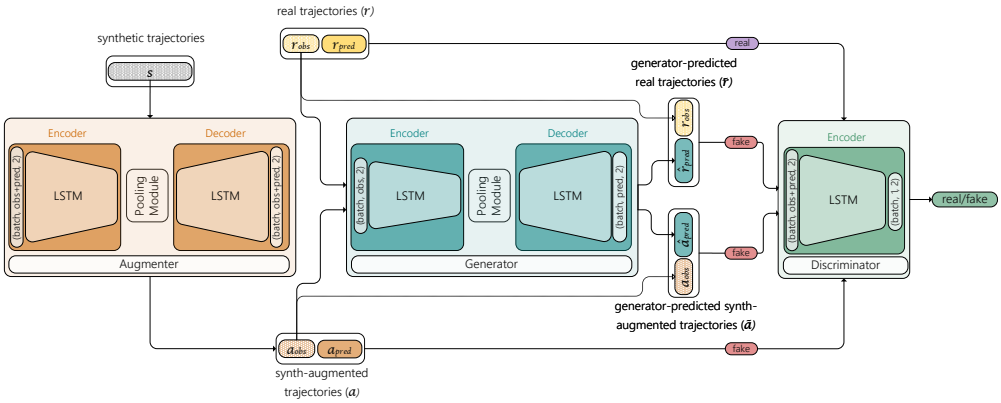


Fig. 5.3 AA-SGAN for adversarially augmented pedestrian trajectories prediction architecture. The Augmenter learns to augment synthetic trajectories into synth-augmented; the Generator learns to generate trajectories prediction; the Discriminator learns to discriminate real from generated and synth-augmented trajectories.

chance to learn over a significantly more challenging problem. Another peculiarity of our proposal is that the above architecture allows us to train the Augmenter not only end-to-end but also over an adversarial loss, rather than just minimizing the difference between its input and output (e.g., MSE). Adversarial Augmenter training is the key towards augmented synthetic trajectories that are practically useful for training the Generator, as we experimentally show later on.

Augmenter The Augmenter A receives in input synthetic trajectories s of length t_{pred} and outputs *synth-augmented* trajectories $a = A(s)$ of identical length. In detail, the Augmenter relies on an encoder-decoder architecture. Thus, sequences with length t_{pred} are embedded into a Multi-Layer Perceptron layer $\phi(\cdot)$ with ReLU non-linearity to get fixed length vectors $e_t^{(i)} = e_t, t \in [1, t_{pred}], \forall i$. Subsequently, these embedding vectors e are used as input to the RNN model - in this case, we use the LSTM cell $f(\cdot)$ - of the encoder ε at time t with the following recurrence:

$$\begin{aligned} e_t &= \phi(x_t, y_t; W_e^{(\varepsilon)}) \\ h_t^{(\varepsilon)} &= f(h_{t-1}, e_t; W^{(\varepsilon)}) \end{aligned} \tag{5.1}$$

where $W_e^{(\varepsilon)}$ are the embedding weights and, $W^{(\varepsilon)}$ are the encoder weights shared between all people in a scene. Once observed trajectories are encoded, we use the

Pooling Module $\rho(\cdot)$ proposed by Gupta *et al.*, 2018 [27] to model human-human interactions: the idea is to obtain a pooled tensor $p_i = p, \forall i$ - consistent with the past - to initialize the hidden state of the decoder δ. Thus, we embed pedestrians through an MLP layer $\gamma(\cdot)$ with ReLU non-linearity and embedding weights W_c.

$$c_t = \gamma(p; W_c) \tag{5.2}$$

Hence, we initialize the decoder state $h_t^{(\delta)}$ as concatenation of c_t and z from $\mathcal{N}(0,1)$:

$$h_t^{(\delta)} = [c_t, z] \tag{5.3}$$

Now, it is possible to generate trajectories for all pedestrian A as follows:

$$\begin{aligned} e_t &= \phi(x_{t-1}, x_{t-1}; W_e^{(\delta)}) \\ p &= \rho(h_{t-1}^{(\delta)}, \ldots, h_t^{(\delta)}) \\ h_t^{(\delta)} &= f(\gamma(p, h_{t-1}^{(\delta)}), e_t; W^{(\delta)}) \end{aligned} \tag{5.4}$$

where $W_e^{(\delta)}$ are the embedding weights and $W^{(\delta)}$ are the decoder weights shared between all people in a scene. Then, the coordinates will be:

$$(\hat{x}_t, \hat{y}_t) = \gamma(h_t^{(\delta)}) \tag{5.5}$$

Generator The Generator G observes the first t_{obs} samples of a trajectory of t_{pred} samples and predicts the next $t_{pred} - t_{obs}$ samples. The architecture of the predictor has an encoder-decoder structure that is entirely analogous to that of the Augmenter A. However, unlike [27], the Generator takes two types of inputs: real trajectories r and synth-augmented trajectories a output by the Augmenter. More in detail, the real trajectory r is divided in two segments called r_{obs} (t_{obs} samples) and r_{pred} ($t_{pred} - t_{obs}$ samples). Then, r_{obs} is input into G that outputs the prediction $\hat{r}_{pred} = G(r_{obs})$, i.e. the predicted continuation of r_{obs}. Similarly, synth-augmented trajectories are split in a_{obs} and a_{pred}. The Generator is similarly fed with a_{obs} and predicts $\hat{a}_{pred} = G(a_{obs})$. The best trajectory is selected by calculating the $L2$-distance for each predicted point, following the method proposed by [27]. Predictions $\hat{r}_{pred}$ and $\hat{a}_{pred}$ are then concatenated with r_{obs} and a_{obs} as $\hat{r}$ and $\hat{a}$, respectively.

Discriminator The Discriminator D consists of a separate encoder with an MLP layer as the last layer to classify if a trajectory is fake or not, i.e., whether a trajectory is socially acceptable. To this end, D is designed as a binary classifier that takes in input trajectories of t_{pred} samples.
In our proposed architecture, the Discriminator has two peculiarities over [27]. The first peculiarity is that the Discriminator is challenged to discriminate between true or fake trajectories over four different classes of inputs rather than just two. Namely, real trajectories r are the only ones that are labelled as real. Generator-predicted real trajectories $\tilde{r} = [r_{obs}, \hat{r}_{pred}] = [r_{obs}, G(r_{obs})]$, Generator-predicted synth-augmented trajectories $\tilde{a} = [a_{obs}, G(a_{obs})] = [a_{obs}, \hat{a}_{pred}]$ and synth-augmented trajectories a that go all under the fake label. This increased variety of false trajectories is expected to provide the Discriminator with more challenging examples at training time. In turn, the Discriminator will propagate better error gradients when the entire architecture is trained as follows. Secondly, the Discriminator receives the entire trajectories as input: in fact, the trajectories predicted by the Generator are concatenated with their observed part ($\tilde{r}$ and $\tilde{a}$) and the real trajectories r and synth-.augmented a are taken in their entirety. In this way, it is possible to use the same Discriminator on both the Generator output and the Augmenter output.

5.3.3 Training Procedure

The above-mentioned is trained end-to-end with an adversarial approach [8].

The Discriminator is trained with the following loss function:

$$\begin{aligned} L_D = \mathbb{E}[&\log(D(r)) + \log(1 - D(a)) + \\ &\log(1 - D(\tilde{r})) + \log(1 - D(\tilde{a}))] \end{aligned} \tag{5.6}$$

with the aim of maximizing the average of the log probability of real trajectories r and the log of the inverse probability for synth-augmented a, Generator-predicted real $\tilde{r}$ and Generator-predicted synth-augmented $\tilde{a}$ trajectories.
Concerning the Augmenter, an $L2$-loss is computed between s and a overall t_{pred} samples, as proposed in [27]. Thus, the Augmenter is trained to minimize the loss:

$$L_A = \mathbb{E}[\log(D(a)) + L2(s,a)] \tag{5.7}$$

Metric	Dataset	**SGAN** [27] real	**SGAN** [27] synthetic	**SGAN** [27] hybrid	**AA-SGAN**
ADE	**ETH**	0.85 ($\pm 9\times10^{-3}$)	1.28 ($\pm 1\times10^{-2}$)	0.93 ($\pm 4\times10^{-3}$)	**0.71** ($\pm 1\times10^{-2}$)
	HOTEL	0.63 ($\pm 4\times10^{-3}$)	0.88 ($\pm 4\times10^{-3}$)	0.63 ($\pm 3\times10^{-3}$)	**0.42** ($\pm 3\times10^{-3}$)
	UNIV	0.67 ($\pm 4\times10^{-4}$)	1.19 ($\pm 8\times10^{-4}$)	0.62 ($\pm 4\times10^{-4}$)	**0.60** ($\pm 8\times10^{-4}$)
	ZARA1	0.42 ($\pm 1\times10^{-3}$)	1.22 ($\pm 2\times10^{-3}$)	0.41 ($\pm 2\times10^{-3}$)	**0.34** ($\pm 2\times10^{-3}$)
	ZARA2	0.40 ($\pm 6\times10^{-4}$)	0.51 ($\pm 8\times10^{-4}$)	0.42 ($\pm 5\times10^{-4}$)	**0.34** ($\pm 7\times10^{-4}$)
Average		0.60	1.02	0.60	**0.48**
FDE	**ETH**	1.63 ($\pm 2\times10^{-3}$)	2.58 ($\pm 2\times10^{-2}$)	1.79 ($\pm 8\times10^{-3}$)	**1.24** ($\pm 3\times10^{-2}$)
	HOTEL	1.36 ($\pm 4\times10^{-3}$)	1.79 ($\pm 1\times10^{-2}$)	1.30 ($\pm 6\times10^{-3}$)	**0.78** ($\pm 6\times10^{-3}$)
	UNIV	1.44 ($\pm 9\times10^{-3}$)	2.36 ($\pm 2\times10^{-3}$)	1.33 ($\pm 9\times10^{-4}$)	**1.25** ($\pm 2\times10^{-3}$)
	ZARA1	0.90 ($\pm 4\times10^{-3}$)	2.55 ($\pm 6\times10^{-3}$)	0.86 ($\pm 3\times10^{-3}$)	**0.68** ($\pm 3\times10^{-3}$)
	ZARA2	0.91 ($\pm 2\times10^{-3}$)	1.06 ($\pm 2\times10^{-3}$)	0.93 ($\pm 2\times10^{-3}$)	**0.71** ($\pm 2\times10^{-3}$)
Average		1.24	2.07	1.24	**0.93**

Table 5.1 Trajectory prediction accuracy in terms of ADE and FDE for a reference SGAN baseline trained on real, synthetic and hybrid (50% real/ 50% synthetic) dataset compared with AA-SGAN.

Indeed, it may seem counterintuitive to introduce an $L2$-loss term when training the Augmenter (we want to make these trajectories different, not identical). However, we observed that, in combination with the Discriminator loss, this yields trajectories that are more useful to train the Generator

Finally, concerning the Generator, two $L2$-losses are computed between $\tilde{r}$ and r and between $\tilde{a}$ and a. However, the Generator shall be obviously trained only over the $t_{pred} - t_{obs}$ predicted samples. Thus, the Generator is trained to minimize the loss

$$\begin{aligned} L_G = \mathbb{E}[\log(D(\tilde{r})) + L2(r_{pred}, \hat{r}_{pred}) + \\ \log(D(\tilde{a})) + L2(a_{pred}, \hat{a}_{pred})] \end{aligned} \tag{5.8}$$

The above-described architecture is trained with the classical GAN training procedure, yet extended to the Augmenter. As a first step, the Discriminator D is first optimised over a real trajectory r, then over the Generator-predicted trajectory $\tilde{r}$, next over synth-augmented trajectory a and finally over the Generator-predicted synth-augmented trajectory $\tilde{a}$. As a second step, the Generator G is optimised first over real trajectories r and then over the synthesised trajectories. As a final step, the Augmenter A is optimised through the synthetic trajectories.

5.4 Experimental results

In this section, we evaluate the performance of our AA-SGAN architecture on the two publicly available real datasets ETH and UCY introduced in Sect. 5.2.2.

The experimental evaluation is based on the leave-one-out methodology used in the related literature. In particular, we consider a total of 5 sets of real trajectories (`eth`, `hotel` from ETH and `univ`, `zara1` and `zara2` from UCY dataset): we train on 4 sets and test on the remaining (left out) set. Other common settings are $t_{obs} = 8$ (3.2 s) and $t_{pred} = 20$ (8 s) which amounts to predicting 4.8 s of the future path followed by each pedestrian (all possible 8 s long trajectories are extracted from datasets using a sliding window with skip equal to 1 frame as in [35, 27, 36]). The trajectory prediction accuracy is measured in terms of Average Displacement Error (ADE) and Final Displacement Error (FDE) [43, 27, 8, 44]. Therefore, given a generic trajectory v, we have:

$$ADE_{v_{pred},\hat{v}_{pred}} = \sum_{t=t_{obs}+1}^{t_{pred}} \frac{\|\hat{v}_t - v_t\|}{(t_{pred} - t_{obs})}, \forall i \tag{5.9}$$

$$FDE_{v_{pred},\hat{v}_{pred}} = \|\hat{v}_{t_{pred}} - v_{t_{pred}}\|, \forall i \tag{5.10}$$

where v_{pred} and $\hat{v}_{pred}$ are, respectively, the ground truth and the model prediction.

As a source of synthetic trajectories for AA-SGAN Augmenter, we use the JTA Dataset [30]: it consists of a vast dataset of trajectories extracted from GTAV (see sample frames in Fig. 5.4), containing 384 full-HD videos of 30 seconds and recorded at 30 fps. Different environments (e.g., airports, stations, squares, and parks), different climatic conditions (e.g., sun and rain), different angles (e.g., from above, from human height, and in motion), and other lighting conditions (e.g., day, night, artificial light) characterize each video. These characteristics make this dataset highly versatile and usable for many purposes. One of the main advantages of synthetic data is the ability to quickly and easily retrieve a large amount of data and related labels. To match the structure of the real pedestrian trajectories used for testing, we rely only on JTA trajectories characterized by a top view. In detail, we track the *head top* skeleton joint to get pedestrian trajectories; also, we maintain the same inter-samples time interval equal to 0.4 s. Overall, the synthetic dataset includes a total of 4,488 frames from 3,834 pedestrians.

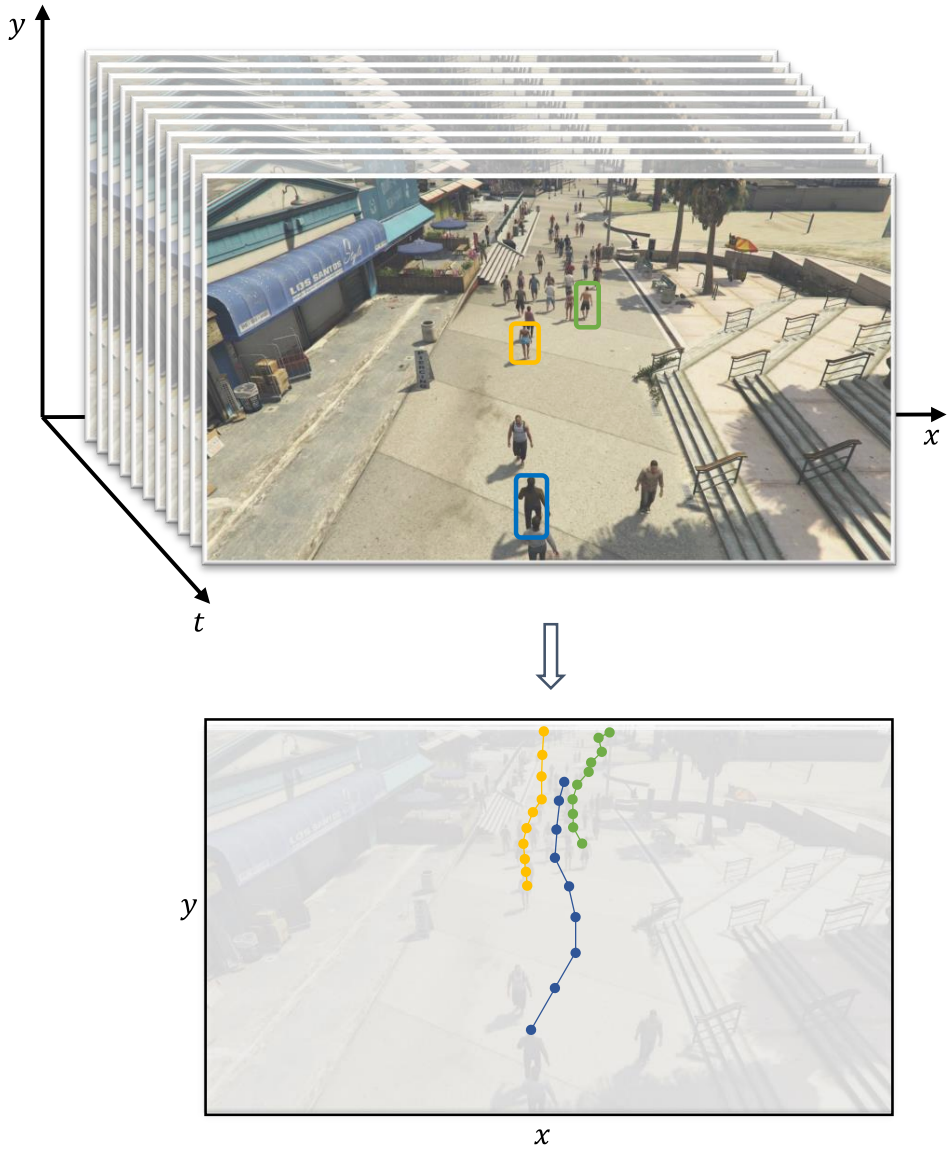


Fig. 5.4 Examples of frames from the JTA dataset (top) and the corresponding trajectories (bottom) we used as a synthetic training set.

5.4.1 Path Prediction Accuracy

Table 5.1 shows the trajectory prediction accuracy obtained by our proposed AA-SGAN scheme and the three baseline SGAN schemes. The *AA-SGAN* scheme refers to the proposed architecture trained as described above. The three baseline schemes refer to a standard SGAN trained, in turn, as follows. The *real* scheme corresponds to the setup [27], where an SGAN is trained over real trajectories from UCY and ETH. The *synthetic* scheme corresponds to the case where an SGAN is trained only over synthetic trajectories from JTA. Finally, *hybrid* refers to the case where SGAN is trained over a 50-50 mix of real and synthetic trajectories from JTA (the size of this hybrid dataset is twice that of the former two). Notice that this mix of real and synthetic trajectories is the same provided in input to our AA-SGAN scheme, as discussed later on. The four schemes share the same training hyper-parameters and configuration suggested in [27].
The results for the *real* scheme reflect those reported in [27], apart from some minor differences on the `zara1` and `zara2`. Please also note that all the results reported in the following are averaged on 3 trials (corresponding standard deviation is shown as well). As expected, performance drops when SGAN is trained over synthetic data only. This performance loss shows that JTA synthetic trajectories are so simple that they are not useful for training an SGAN. For example, on the ETH test, the value for ADE increases from 0.85 to 1.28 when training on synthetic trajectories only. The *hybrid* column in Tab. 5.1 shows that accuracy does not improve over the *real* scheme. For the ETH experiment, we even report ADE equal to 0.93, which is significantly worse than *real*. These preliminary experiments show that synthetic trajectories cannot replace real ones at training times, nor do they bring any benefit if mixed with them.
The last column shows that the proposed AA-SGAN method yields much better accuracy in all cases, both in terms of ADE (20% reduction from 0.6 to 0.48) and FDE (25% reduction from 1.24 to 0.93). We recall that the AA-SGAN scheme is trained over the same mix of real and synthetic trajectories used for the *hybrid* scheme. Such difference in accuracy despite the same training set can be brought down to the job the Augmenter performs, which makes synthetic trajectories eventually useful for training. In the AA-SGAN scheme, we used a 50-50 ratio of real and synthetic trajectories: this is a reasonable choice as the Augmenter and Generator are fed with

Metric	Dataset	**AA-SGAN**	**Independent Augmenter**
ADE	**ETH**	**0.71**	0.74
	HOTEL	**0.42**	0.57
	UNIV	**0.60**	0.61
	ZARA1	**0.34**	0.36
	ZARA2	**0.34**	0.37
AVG		**0.48**	0.53
FDE	**ETH**	**1.24**	1.45
	HOTEL	**0.78**	1.09
	UNIV	**1.25**	1.30
	ZARA1	**0.68**	0.75
	ZARA2	**0.71**	0.78
AVG		**0.93**	1.07

Table 5.2 Trajectory prediction accuracy of AA-SGAN (joint Augmenter training) versus independent training of the Augmenter feeding reference SGAN.

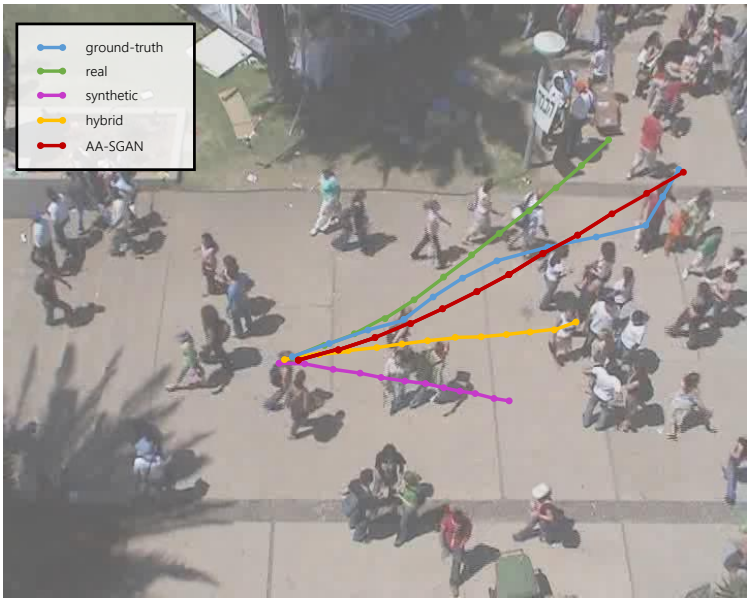
a balanced mix. In the following ablation study, we will discuss whether it is useful to alter such a real-synthetic ratio and to what extent.

Figure 5.5 illustrates some trajectories predicted by the four schemes in Table 5.1. The *synth* scheme predicted trajectories are the worst, diverging the most from the ground truth other than being not acceptable due to collisions (see Fig 5.5b). On the contrary, the results of the AA-SGAN scheme are the closest to the ground truth.

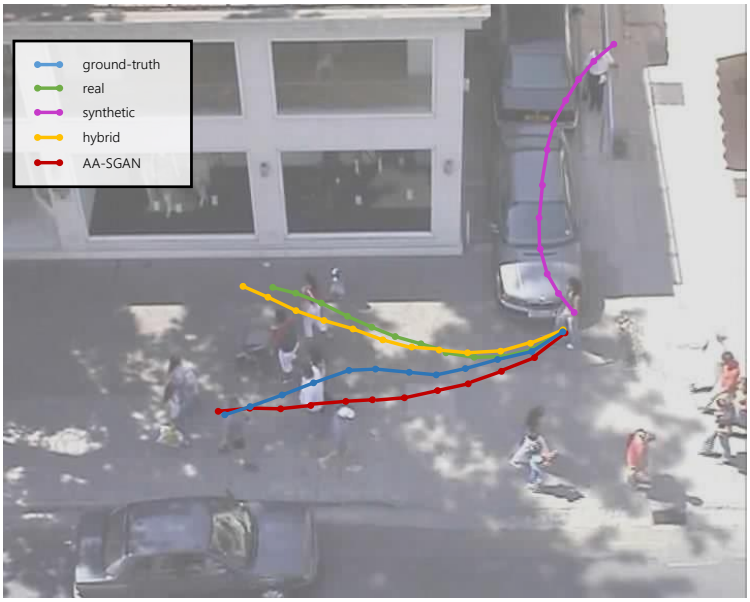
5.4.2 Ablation Study

Before moving further with the experiments, we recall that the experiments with schemes *real*, *synthetic* and *hybrid* in Tab. 5.1 above can be already interpreted as an ablation study. In fact, the standard SGAN architecture trained over a hybrid dataset can be seen as equivalent to AA-SGAN minus the Augmenter, where synthetic trajectories are fed directly into the Generator.

Independent Augmenter training In this first ablation experiment, we investigate the advantage of *jointly* training the Augmenter A and the Generator G in an adversarial framework. We recall that in our architecture, G and A are jointly optimized over the gradient of the adversarial loss function backpropagated by the



(a) `univ`



(b) `zara1`

Fig. 5.5 Visual comparison between the groud-truth (blue) and predicted trajectory by SGAN with real (green), synthetic (magenta) and hybrid (yellow) training. Predictions by the proposed AA-SGAN are in red and are the closest to the ground truth. The selected results are taken from `univ` and `zara1` testset.

		Real-Synthetic ratio	
Metric	Dataset	1-to-1	1-to-10
ADE	**ETH**	0.71	0.70
	HOTEL	0.42	0.54
	UNIV	0.60	0.69
	ZARA1	0.34	0.35
	ZARA2	0.34	0.34
AVG		0.48	0.52
FDE	**ETH**	1.24	1.25
	HOTEL	0.78	1.09
	UNIV	1.25	1.38
	ZARA1	0.68	0.70
	ZARA2	0.71	0.68
AVG		0.93	1.02

Table 5.3 ADE and FDE values for 1-to-1 and 1-to-10 ratio between real and synthetic (R/S) trajectories used in AA-SGAN training (the number of real trajectories does not change).

Discriminator *D*. In order to investigate on this advantage, we removed *G* from the AA-SGAN architecture and we train the resulting GAN, between *A* and *D*, using their corresponding losses. As a consequence, in this ablated architecture the Augmenter will create synth-augmented trajectories without any feedback from *G*, i.e. without being able to appreciate their contribution to solving the prediction task; the obtained synth-augmented are stored offline for the subsequent training. Next, these trajectories are used to train a reference SGAN in the *hybrid* scheme (50% real and 50% synth-augmented trajectories). *G* and *D* are trained according to the same adversary loss functions described in the previous section.

Tab. 5.2 compares the results of this scheme with AA-SGAN. Performance is still above the *real* baseline. However, it is well below AA-SGAN. This experiment confirms the importance of jointly training the Augmenter *A* with the rest of the architecture.

Synthetic to real ratio Another aspect worth investigating is the ratio between the real and the synthetic trajectories used to train AA-SGAN. Tab. 5.3 shows the prediction accuracy when the synthetic trajectories increase by a 10-fold factor (the number of real trajectories remains constant). On average, a drop of about 10% in ADE and FDE is observed, showing the importance of balancing real and

synthetic trajectories. We hypothesize that this predominance of synthetic traces makes adversarial training less stable, explaining the performance drop.

5.5 Conclusions

This work proposed AA-SGAN, a generative architecture for predicting accurate pedestrian trajectories leveraging synthetic trajectories beside real ones. We experimentally showed that computer-generated synthetic trajectories bring no benefit when used to train the state-of-the-art SGAN generative model. However, if synthetic trajectories are first augmented before being fed to the Generator (*synth-augmented* trajectories), they boost the diversity of the training set, improving the accuracy of the predictions. Through an ablation study, we show that joint training of the Augmenter with the rest of the architecture is the first key element towards accurate trajectory predictions. Through ablation, we also show that a balanced ratio between real and synthetic trajectories is another key element of our architecture.

Part II

Natural disasters management

Chapter 6

The emergency scenarios

In the last decades, the frequency and intensity of natural disasters have risen significantly. According to worldwide data from the Centre for Research on the Epidemiology of Disasters [45], about 12 times more natural disasters were registered in 2017 than in 1950. In Figure 6.1, the dramatic increase in these events is shown, which comprise mass movements, volcanic activities, wildfires, landslides, earthquakes, extreme temperatures, droughts, extreme weather, floods, and epidemics. It can be noted that flood events were persistent; in 2017 alone, floods represented approximately 39% of global natural disasters. On top of the tragic loss of human lives and infrastructures, natural disasters come at a high cost to governments. The European Commission (EC) estimated that, since 2005, natural disasters have cost the European Union (EU) close to 100 billion euros. However, this cost can be significantly reduced by investing in risk prevention: the EC stated that for every 1 euro spent on prevention, 4 euros or more could be saved in response. In this respect, with the "cohesion policy", the EU allocated 8 billion euros for climate change adaptation, risk prevention, and management over the 2014–2020 period [46]. Those investments generated several projects and research opportunities from which this work is taken. The ability to detect the insurgence of such problems promptly is a powerful tool for the bodies in charge of protecting and guaranteeing citizens' safety.

A flood occurs when an overflow of water inundates a portion of land that is usually dry, which can happen in several ways. For example, a flood can be caused by excess rainwater in saturated ground, overflow of water bodies such as rivers

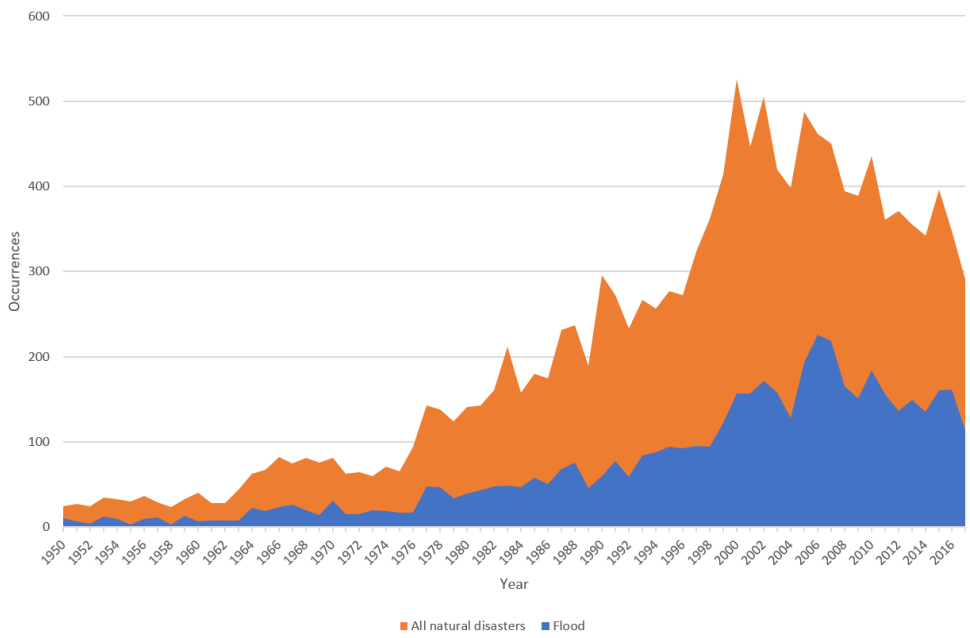


Fig. 6.1 The occurrence of natural disasters and flood events since 1950. Natural disasters are characterized by the following phenomena: mass movements, volcanic activities, wildfires, landslides, earthquakes, extreme temperatures, droughts, extreme weather, floods, and epidemics. The chart shows a moderate increase in this kind of event, which, by 2017, has increased by over a factor of 10 with respect to 1950. Another important aspect is the percentage of incidence of flood events, representing, on average, 30% of the overall natural disasters [47].

and lakes, rapid snow or ice melting, storm surge, or tsunami. Such events are exacerbated by climate change effects, including more intense precipitation and higher temperature variations. Also, lousy water management can cause floods, e.g., an excess discharge from dams can cause surges in rivers downstream; negligent bank maintenance can result in their failure during water discharge peaks, and a poor sewage system will not be able to handle extreme rainfall events in urban areas. Most floods are induced by extreme rainfall and can be predicted to a certain extent. Usually, floods take several days to develop, giving residents time to prepare and follow evacuation plans. Less often, floods can develop in just a few hours, i.e., the so-called flash floods, which are extremely dangerous and more difficult to predict.

During these phenomena, a quiet river can rapidly turn into a flood, bringing debris and rubble along with water in its downstream path. It is not easy to quantify how human activities affect extreme weather events. However, it is increasingly evident that climate change has influenced several variables related to flood events

(a) (b)

(c) (d)

Fig. 6.2 Examples of emergency events related to flooding disasters

[48]. Over the last century, the steady increase in temperatures has changed the hurricanes travelling mechanics, making them slower and consequently letting them cause intense rainfalls. At the same time, the melting of permanent ice zones and glaciers is contributing to the worldwide sea-level rise, creating an increasing threat to coastal areas and cities like Venice (Italy), which was recently hit by a flood event of historic proportions. Annually, floods cause more than $40 billion in damage worldwide [49]. From 2007 to 2016, 5553 people died because of floods, while in 2017 alone, the death toll reached 3331 [50]. The abovementioned figures demonstrate the increasing severity of floods, indicating the need for novel approaches and tools to reduce their impact worldwide. Monitoring water flows is critical to implementing adequate early warnings. At the same time, the analysis of in-field data can contribute to early event detection and the real-time understanding of flood impacts. To achieve these goals, the automated analysis of images and videos through algorithms based on Artificial Intelligence can be of great importance, mainly when applied to heterogeneous data coming from multiple sources, including fixed surveillance cameras installed near river beds or shores, geolocated images taken from drones or other aerial vehicles [51], posts extracted from social media [52], in-field pictures generated by ad-hoc crowdsourced mobile applications [53].

In conclusion, investing in risk prevention and management with computer vision algorithms can play a role in helping with natural disasters such as floods. These algorithms can analyze images and videos from cameras and drones to detect and monitor flood events in real-time. This information can provide early warning alerts, track the progression of the flood, and support rescue and recovery efforts. Computer vision algorithms can also be used to analyze satellite imagery to detect changes in land use, water levels, and vegetation, providing valuable information for flood prediction and management. Overall, computer vision algorithms can be a valuable tool for natural disaster management, providing real-time information and helping to reduce the impact of floods on communities.

Chapter 7

AI-based flood event quantification using online media and satellite data

In this chapter, we study the problem of flood detection and quantification using online media and satellite data. We present three approaches, two based on neural networks and a third based on the combination of different bands of satellite images. The second approach is the most interesting for the thesis topic. We developed a multi-branch model, and as we will see in the chapter, it can learn context and local features and merge them with direct fusion through concatenation to improve baseline predictions. This work aims to detect floods and give relevant information about the flood situation, such as the water level and the extension of the flooded regions, as specified in the three subtasks, for which we propose a specific solution.

7.1 Introduction

The frequency and intensity of natural disasters have risen significantly due to climate change. Flood events alone represent about 39% of the natural disasters occurred worldwide. During this type of natural disaster, emergency responders must have as much information as possible about the magnitude, the areas affected, and the situation and location of people in danger. To extract this information, we consider two sources: online news articles and satellite spectral imagery. Thanks to rapid access to the internet, online news contain information about natural disasters in almost real-time. At the same time, satellite spectral imagery can give information

about the extension of the flood. Using these two information sources, we propose approaches for flood event understanding and quantification:

- An algorithm that determines if an image extracted from an online news article contains relevant information about the flood. For example, images of the flood itself, but also images of emergency responders, people in danger, etc.
- An algorithm that gives an image extracted from online news determines if there is water in the image and, in the case of containing water, if the water level is above or below the knee level of the people in the scene if there are. It also contemplates the use of news text as additional data for inference.
- An algorithm that, given spectral imagery from satellites, segments the images' water regions and gives a flood/no flood prediction and an estimation of the flood extension.

This work has been done in the context of MediaEval 2019 as a participation in the Multimedia Satellite task. Detailed information about the task and data can be found in [54].

7.2 Related Work

Emergency prevention, detection, assistance, and understanding through computer vision and image processing techniques have been an open problem since the early stages of this field [55]. In particular, in the flood detection domain, scientific work mainly focuses on flood detection either in social media or satellite data [56–58]. Among the latest, several approaches are known in the literature and exploit spectral bands and other sensor measurements [59–63] to retrieve good indicators.

This work builds on top of the Multi-modal deep learning approach for flood detection in [64], which used social media images together with their metadata to determine if a social media post contained visual information about a flood, and the deep learning models for passability detection of flooded roads in [65, 66], which went a step further and gave information about the state of the roads during a flood event, information that is of utmost importance during a flood to build a map of accessible roads for rescue and supply operations. Moving in this direction, we aim to estimate the water level in this work.

Fig. 7.1 Sample images extracted from articles from the NITD dataset. This task aims to classify into two classes whether they belong to a flooding event or not.

7.3 Approach

In this section, each stage of the solution will be briefly introduced.

News Image Topic Disambiguation (NITD): during flooding, the media usually updates the information about the situation to keep the reader updated. Due to many online newspapers and media, searching for these relevant articles can be time-consuming. Optimizing the search using natural language processing (NLP) algorithms or keyword searches is possible. Since most of these articles contain images, in this first stage, we want to refine the search using a computer vision algorithm to classify those images as flood event related/not flood event related.

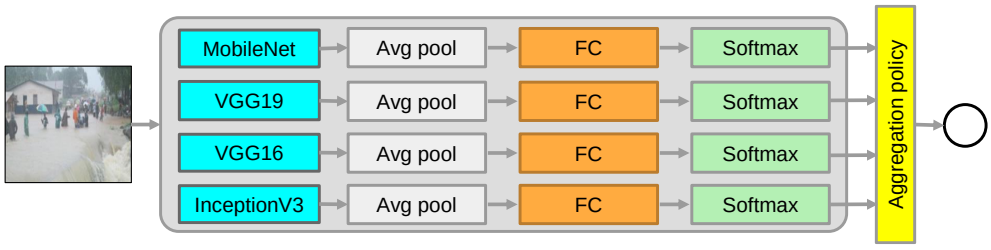


Fig. 7.2 Ensemble architecture used for the NITD task.

To train the classifier, we use the training set for this task, composed of 5145 images retrieved from online news containing information about a flood by an NLP or keyword algorithm and then manually classified, as shown in Figure 7.1. As for the algorithm, we use an ensemble of 4 state-of-the-art networks, as in Figure 7.2, (InceptionV3 [67], MobileNet [68], VGG16 and VGG19 [69]) and cross-validation using two folds. Since the dataset is highly imbalanced, we balance the dataset during training by randomly undersampling the negative class for each epoch. This way, the dataset stays balanced, but we use all the samples from both categories. Finally, we combine the networks by majority voting.

(a) (b)

Fig. 7.3 Sample flood-event related images from articles of MFLE dataset. The goal of this task is to classify images based on text and visual information, whether there are people standing in water that is above knee level.

Multimodal Flood Level Estimation (MFLE): given online articles with visual and textual information, as shown in Figure 7.3, we developed a textual, textual-visual, and only visual model to estimate the flood level by predicting if the water is above or below the knee of the people in the scene. The latter model is composed of two branches: (i) it takes as input image crops of a person's knees extracted by a state-of-the-art pose estimator [70] and predicts if the knee is under or above the water; (ii) it takes as input a full image of the scene and predicts if the image has people with knees underwater.

To create the training data for the first branch, we used the pose estimator algorithm to extract a region around all the knees from the training set. Knees from images labelled as 0 (water below the knee) were labelled as 0 by default. In contrast, the ones belonging to images labelled as 1 were manually labelled since there could be people in the same image with water levels above or below the knee. Both networks use a VGG19 [69] pre-trained on ImageNet [71] to extract deep features

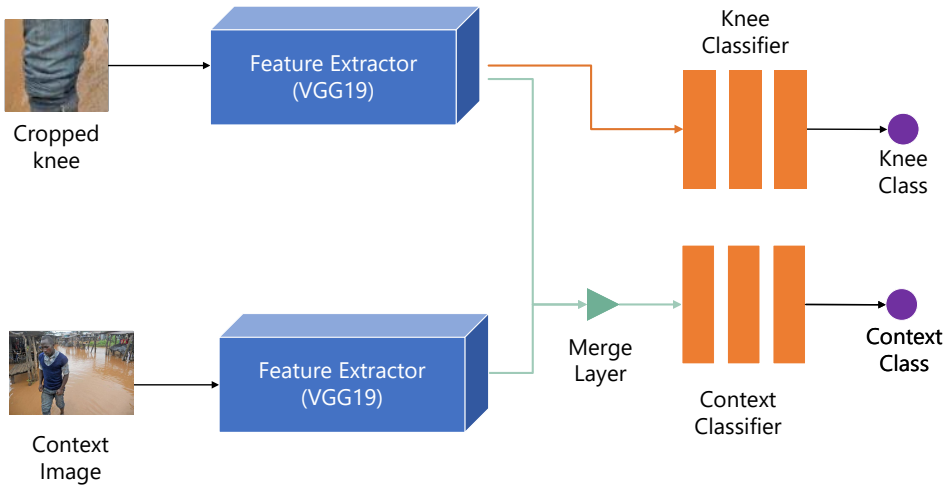
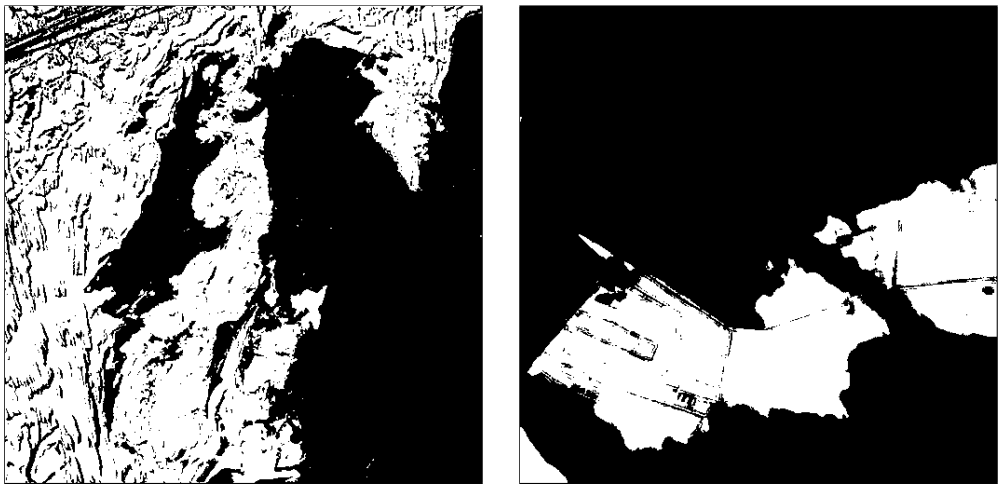


Fig. 7.4 Architecture used for the MFLE task.

of the images, followed by a fully-connected (FC), as in Figure 7.4. Then the information is concatenated to combine the semantic features of the knee with the context information provided by the full-resolution image. This way, the first branch gets information about the context, while the second gets information about the knees. Finally, an FC estimates if the knee is above or below the water, and another FC if the water is above or below the knee level. The two-branch system is proposed because a simple one-branch Convolution Neural Network (CNN) would greedily learn to predict flooded images as the "water above the knee" class since it lacks specific data about the knees in the scene. So it would associate the features of a flooded area as "water above the knee" class because it is solely composed of these examples.

Finally, an image is classified as "water above the knee" if there is at least one knee in the scene that is classified as "water above the knee" by the knee branch, and the context branch also classifies the image as "water above the knee". We also combined textual data from the articles to verify if it could lead to a better predictor. This was achieved by building an ensemble composed of the previous model and an NLP module. This module comprises a bidirectional Long Short-Term Memory (LSTM) network. The result of the LSTM is concatenated with the last FC layer of the image classifier. The only textual model is composed of the module described above alone.



(a) (b)

Fig. 7.5 Sample images for different cities from CCSS dataset. This subtask aims to classify image sequences into two classes, whether they belong to a flooding event or not.

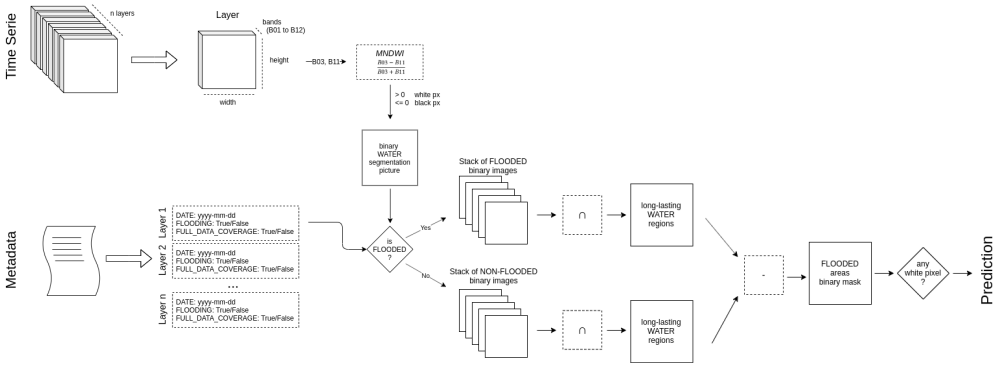


Fig. 7.6 Model used for the CCSS task.

City-centered Satellite Sequences (CCSS): given a sequence of Sentinel-2 satellite images depicting a certain city over a certain length of time, this task aims to classify whether a flooding event was ongoing in that city.

We built an *expert system* which leverages both the spectral and the related metadata information. Firstly, it computes a binary mask for each layer, as in Figure 7.5, in which white pixels represent areas with the presence of water while black pixels represent the other regions. The binary masks are obtained: (i) by computing, for each pixel, the Modified Normalized Difference Water Index (MNDWI) [72] adapted for Sentinel-2 bands (S2), according to Equation (7.1); (ii) by setting to white the

pixels having $MNDWI_{S2} \geq 0$, black the others.

$$MNDWI = \frac{\rho_{green} - \rho_{swir1}}{\rho_{green} + \rho_{swir1}}, MNDWI_{S2} = \frac{B03 - B11}{B03 + B11} \tag{7.1}$$

Assuming that the dataset does not have missing values lasting for the whole time series, we set the pixels related to uncovered areas to white. Then, we performed the pixel-wise intersection among two layers: (i) the computed binary layers marked as FLOODED and (ii) the ones marked as NON-FLOODED in the metadata file.

The two images depict the water persistence in case of flood and non-flood. Finally, a pixel-wise difference between the two sets is computed to discriminate flooded regions from normal water sources (like rivers or lakes). Even if a binary mask representing the residual flood extent is available, to comply with the *CCSS* subtask, the approach returns 1 if there is still any white region in the resulting binary mask and 0 otherwise, as in 7.6.

7.4 Results

The results, split by subtask, are reported in Table 7.1. For the subtasks *NITD* and *MFLE*, the F1-Scores are referred to as the 20 % of the development set, used as the validation set. Conversely, the *CCSS* proposed to approach an expert system, and the whole devset was used. In this latest subtask, the confusion matrix on the devset, TP:108, FP:0, FN:33, TN:127, shows that the approach is robust against false positives, having a precision of 1.0.

Subtask	Data	DevSet F-Score	TestSet F-Score
NITD	Visual	0.8062	0.6628
MFLE	Visual	0.7667	0.5428
	Text	0.5213	0.4956
	Visual & Text	0.5454	0.5284
CCSS	Satellite	0.8850	0.9118

Table 7.1 Results per subtask

7.5 Analysis and conclusions

Conclusions present our insight on the subtasks. As can be seen in Table 7.1: **(NITD)**, balancing the dataset during training and combining different models significantly improves the performance. **(MFLE)** (i) Merging global and local classifiers improve the performance; (ii) the text does not bring any information. It is so noisy that it even degrades the results compared to the visual-only model; (iii) people's water reflection degrades the performance of the pose estimation algorithm. (iv) the importance of the two branches approach is supplied by an ablation study in which the two-branch model achieved 0.79 F1-score on validation, while the entire image branch alone achieved 0.71 and the branch using the cropped knees achieved 0.76. **(CCSS)** (i) B03 and B11 are highly informative for water segmentation; (ii) the approach is an expert system; therefore, there is no need for a training set, and it is computationally fast;

7.6 Acknowledgments

This work was supported by the European Commission H2020 SHELTER project, GA no. 821282, and by the Spanish grant TIN2016-75404-P.

Chapter 8

Road passability detection during flood events using social media data

During natural disasters, situational awareness is needed to understand the situation and respond accordingly. An essential need is assessing open roads for transporting emergency support to victims. This can be done by analyzing photos from affected areas with a known location. This work studies the problem of detecting blocked/open roads from photos during floods by applying a two-step approach based on classifiers: does the image have evidence of a road? If it does, is the road passable or not? We propose a single double-ended neural network (NN) architecture that addresses both tasks simultaneously. Both problems are treated as a single-class classification problem using a compactness loss. The study was performed on tweets posted during flooding events containing (i) metadata and (ii) visual information. We studied the usefulness of each data source and the combination of both. This analysis was carried out using a dedicated loss function on the merging layer. The purpose of the function is to provide a better characterization of the features and a better clustering of similar features in the latent space. The compactness loss, in combination with the merging of the features experimentally, proved to be the best model, as the best scoring model, and won the challenge of MediaEval 2018 for the Flood classification task. Finally, we studied the performance gained from different ensembling networks. Through the experimental results, we prove that the proposed double-ended NN makes the model almost two times faster and the load on memory lighter while improving the results concerning training two separate networks to solve each problem independently.

8.1 Introduction

In this work, we will focus on flood events and, specifically, on assessing the status of roads after floods since knowing the best route to access the affected areas is crucial to transport emergency support to victims. This can be done by analyzing photos from affected areas with a known location. Such photos can be: (i) solicited via dedicated apps, such as UN-ASIGN [73] and I-REACT [74], or (ii) harvested from unsolicited sources, such as social media, as people frequently share pictures during emergencies. Using apps and social media to engage the civil population is of increasing interest and can be helpful for first responders.

Indeed, social media is not primarily known and adopted as an emergency reporting tool, but there is evidence [75] of a large number of posts that provide direct proof of natural disasters, which, if properly processed, could help in the handling of emergencies. The need for sensibility regarding natural disasters and the variety of data to deal with make the research community an active player in those topics and generate numerous important and influential conferences.

The objective is, given a collection of posts (including images) related to floods, to determine whether: (i) there is *Evidence of Roads* and, in a positive case, (ii) there is *Evidence of Road Passability*. In the first case, we are more interested in asserting the presence of a road in the picture: this means the road can be directly visible, or enough elements are justifying its existence, such as the presence of traffic lights or vertical signs. On the other hand, the second goal aims to determine whether the identified road is in good condition to be transited. In the flooding context, the evidence of road passability means that the road is completely clean or can be partially or totally covered by water. However, there must be evidence that vehicles or people can still pass through it.

This work has been inspired by the *Flood classification challenge* from MediaEval 2018, where we presented an algorithm that predicted if there was evidence of road and, if so, if the road was passable. We presented an algorithm that achieved the best results in the challenge [66]. The work has three goals:

- Provide a more detailed explanation of the work presented in [66], which was published as an extended abstract due to the page limitation.
- Contextualize our results with all the results from the challenge participants.

- Introduce two significant modifications to the algorithm, namely a new loss and new architecture, which combines the two problems into a single network, which introduces an almost 10% gain in performance for the passability task while maintaining the road evidence task performance. Moreover, we make the problem end-to-end and the solution almost 90 times faster and lighter, obtaining a model that can be feasibly integrated into a real-life solution.

The chapter is organized as follows. Section 8.2 introduces state-of-the-art techniques, focusing on the ones presented in the same competition. Then, Section 8.3 focuses on the quantitative and qualitative analysis of the available data and how they are used to build the dataset. The approaches developed specifically to deal with textual, and picture information is explained in Section 8.4 and evaluated in Section 8.5, where the results are compared with those of the other techniques presented in the MediaEval 2018 competition. Finally, the conclusions and future improvements are described in Section 8.6.

8.2 Related Work

In the twenty-first century, our social interaction habits mainly revolve around smartphones and IoT devices. The Internet, in general, and social media represent a new way for us to learn and communicate. In a personal Facebook or Twitter profile, it is easy to find personal information about daily activities and news about real-time events. During natural disasters, social media represents a huge source of information from which, if properly processed, it is possible to extract valuable data for emergency management organizations. Indeed, the research literature presents several studies to detect, collect and process valuable information [76–78]. Concerning flood events, general approaches aim to detect flood events [79, 80, 64], to segment water regions [81], or to estimate water level [82]. Other approaches aim to examine details, such as the presence of people [83, 84], the the identification of the most affected areas [77], or the identification of flooded roads and their viability. This last topic is addressed in our work and is thought to be an extension of the approaches presented at the MediaEval 2018 conference. Therefore, this section introduces the methods submitted to the MediaEval challenge. The techniques are developed to deal with the two main kinds of data available from social media—metadata and images. As extensively described in Section 8.3, metadata is composed of textual

information (e.g., the text of the post, title) and punctual information (e.g., coordinates, post creation date, post author reference), while the images are PNG or JPG pictures. The metadata information was approached in many ways. A simple approach was proposed by Zhao et al. [85]. They manually created a set of rules which, leveraging on the textual part of the tweets, look for n-grams (a subset of n contiguous words in the same sentence) representing strings of lexical items they would expect to occur in tweets related to road passability. Other works, such as the ones proposed by Hanif et al. [86] and Moumtzidou et al. [87] started with a pre-processing of the tweet texts—first removing hyperlinks, punctuations and symbols and performing the word tokenization, then removing the stop-words and performing word stemming. The processed information was enriched by adding other metadata features, such as user tags. Another work by Kirchknopf et al. [88] proposed to check the metadata language feature, and it increased the number of English tweets by translating the ones written in other languages. This simple step avoids the need to handle multiple languages simultaneously, which is still an open problem in Natural Language Processing. To properly process by classifiers, words in tweets are then translated into numerical features. This step was made through the use of (i) pre-trained word embeddings, which convert words into numerical vectors, such as fasttext [85, 89], Word2Vec [90] or GloVe [91], and/or (ii) statistical features, like Term Frequency—Inverse Document Frequency (TF-IDF) [92]. Numerical features are then used to train models such as Support Vector Machines (SVM) or Convolutional Neural Networks (CNNs) [93] for the final classification.

Regarding the visual information, two approaches were mainly adopted on pictures: (i) using visual descriptors and (ii) extracting features from pre-trained CNNs. In the first case, the aim was to describe the images through a set of discrete information that could lead the classifier to improve performance on the tasks. Several descriptors were already available from the dataset: Color and Edge Descriptor (CEDD) [94], Color Layout (CL) [95], Fuzzy Color and Texture Histogram (FCTH) [96], Edge Histogram (EH) [97], Joint Composite Descriptor (JCD) [98] and Scalable Color Descriptor (SCD) [99]. In the latter case, hidden layers of CNNs are used as feature descriptors. State-of-the-art CNNs such as AlexNet [1], DenseNet201 [100], InceptionV3[67], InceptionResNetV2 [101], ResNet [102], VGG [103] or YOLOv3 [104] were taken after they had been pre-trained on popular and wide datasets such as ImageNet [2], Places365 [105] or VOC [106] , and then fine-tuned on the dataset of this work. Leveraging on pre-trained networks is a

common practice in deep learning research: training a single model from scratch requires prohibitive computational performances, nearly inaccessible to most research centres or universities. Within the context of this work, using CNNs that were pre-trained on datasets containing a variety of places, environments, and buildings enables them to represent and recognize objects and shapes in their internal layers. Fine-tuning such networks on a smaller dataset for specific tasks, such as the ones used in this work, allows them to reuse the pre-trained knowledge to achieve the goal more effectively. Most of the works proposed during the MediaEval competition used the CNNs mentioned above to extract visual features from the last layers of the networks. Moreover, besides extracting *global* features (pertinent to the whole image), Bischke et al. [107] and Zhao et al. [108] also combined information related to single entities (i.e., cars, boats, persons), named *local* features.

Then, extracted features were used for classification in several manners. One option [85, 87, 109] was to feed them as input for a neural network having few fully connected layers and using softmax for classification. Other approaches used other state-of-the-art machine learning algorithms, such as Support Vector Machine (SVM) [86, 88, 107, 108, 110], Multinomial Naive-Bayes, Random Forest and SRKDA [86]. The best approaches exploited ensemble models, whose final output was determined by majority voting or averaging each model's prediction. Ensemble models were also used for feature extraction, combining features extracted from the same picture by several CNNs [85]. Finally, two strategies were used to merge metadata and visual information: *early* and *late fusion*. The *early fusion* combines the features before being computed by the classifier(s), while *late fusion* averages the prediction of the approaches separately developed for the two domains.

In our work, we introduce a novel lightweight network architecture that achieves comparable results of the winner approach, namely an ensemble model of 45 CNNs per task. The proposed approach leverages a custom loss function and accomplishes both tasks simultaneously, reducing the number of needed parameters.

8.3 Dataset

The dataset used to train, validate and test the algorithms was distributed by MediaEval 2018 for the Multimedia Satellite Challenge [111, 65]. It consists of 7387 tweet ids for the development set and 3683 tweet ids for the test set. By the time the

images were downloaded for this competition, a significant number of tweets were no longer available, which resulted in a development set of 5818 tweets and a test set of 3017 tweets. However, since the work done corresponds to an extension to work done for the Multimedia Satellite Challenge, and we do not have the ground truth for the test set data, we will divide the training set into training (4074 images), validation (872 images) and test (872 images). The images for the test set will only be used to report the final results to make the setup as close as the original challenge.

The tweets have been collected by retrieving all the tweets with images containing the tags *flooding*, *flood* and *floods* during the hurricanes *Harvey*, *Irma*, and *Maria*. Since the image information is crucial to this work and many images were duplicated in the tweets, the dataset distributors carried out a process to remove duplicate images. This process is described in detail in [111].

The provided ground truth for the tweets was manually generated through a crowdsourcing task and consists of a binary class label for the evidence of road presence and only for those images classified as containing a road, a second binary class label for the actual passability of the road. Positive road passability is considered when the road is practicable by conventional means (no boats, off-the-road vehicles, monster trucks, Hummer, Landrover, or farm equipment), and it is, therefore, related to the water level and the surrounding context.

The annotators made the decisions based only on the content of the analyzed images. The dataset is significantly imbalanced towards the non-evidence of road, having only $\sim$36% of the tweet images containing roads. In $\sim$45% of the tweets labelled as containing roads, there is evidence of positive road passability. In Table 8.1, the absolute number of images about each class is displayed.

Table 8.1 Information about the number of images of each set and the number of images with evidence of road and the number of images with passable roads.

Dataset	**# Tot. Imgs**	**# Evid. of Roads**		**# Passable Roads**	
		YES	**NO**	**YES**	**NO**
development set	5818	2130	3688	951	1179
test set	3017	-	-	-	-

8.3.1 Metadata

Each tweet has a set of metadata associated with it, including the user who tweeted it and the text shared by the user. In Table 8.2, we briefly describe each tweet's most relevant fields (metadata). Since many of them are empty or semi-empty, we only report the fields (16 out of 29) without missing values in the MediaEval 2018 tweets.

8.3.2 Images

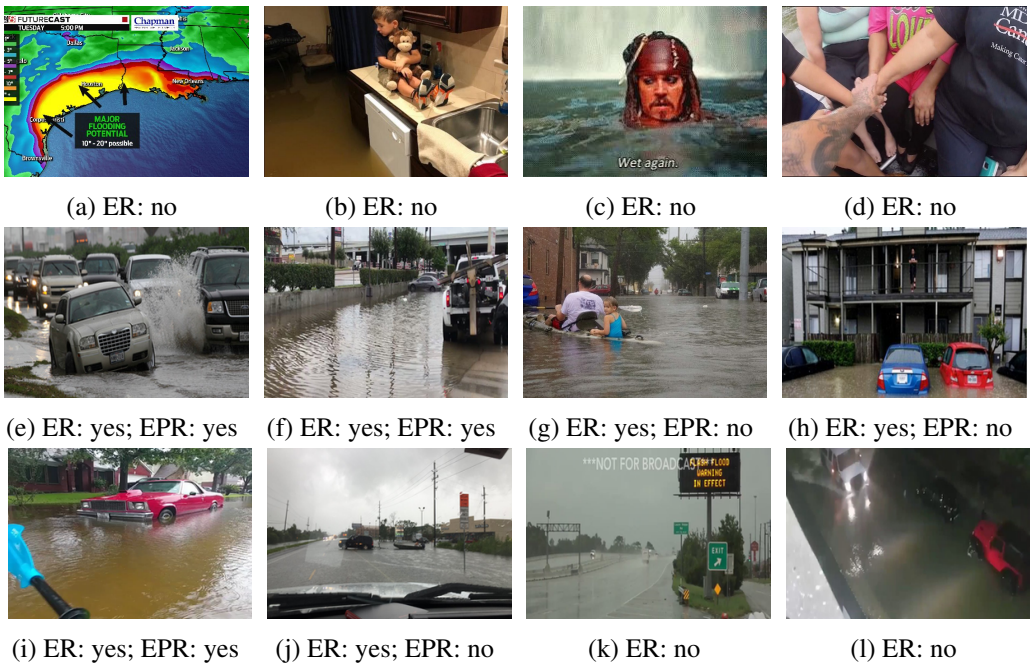
Since the tweets have been retrieved using flood-related tags, most of the images contained in the dataset are related to floods. Among the images that have been classified as not containing roads, some contain charts or weather maps, others contain information about floods unrelated to roads, whereas some images contain no flood information. The images containing evidence of passable roads, in many cases, show cars crossing the road or have enough surrounding contextual information to infer that the water level is not very high. In contrast, the images containing evidence of roads with negative passability contain cars stuck on roads and people crossing the street with boats in many cases. Some examples of the images contained in the dataset are given in Figure 8.1. Sometimes the differences between positive and negative road passability are very subtle and subjective (e.g., see Figure 8.1i,j), while we believe others are wrongly classified (e.g., see Figure 8.1k,l).

8.4 Proposed Solutions

In this section, we describe a solution using only metadata information, a solution using only the tweeted image, and a solution that combines both sources of information.

8.4.1 Algorithm Based on Metadata Only

As explained in Section 8.3, each tweet contains 29 different fields, but only 16 had non-empty values in at least 90% of the tweets. Therefore, the other 13 features were discarded since they do not contain enough information to give any statistically significant information. Moreover, we discarded the following features: (i) *"Created*



(a) ER: no (b) ER: no (c) ER: no (d) ER: no

(e) ER: yes; EPR: yes (f) ER: yes; EPR: yes (g) ER: yes; EPR: no (h) ER: yes; EPR: no

(i) ER: yes; EPR: yes (j) ER: yes; EPR: no (k) ER: no (l) ER: no

Fig. 8.1 Examples of images from the dataset. The *first row* (a-d) contains images classified as not containing Evidence of Roads (ER), while the *second row* (e-h) contains images classified as containing evidence of roads and their corresponding Evidence of Road Passability (ERP). The *third row* (i-l) corresponds to images that were difficult to classify or wrongly classified.

at", which contained the date on which the tweet was posted. Since the tweets were collected during specific hurricane events (namely, Harvey, Irma, and Maria), we considered this field to have minimal time coverage with the risk of being biased and, therefore, useless. Specifically, the development set contains tweets from 38 different days. (ii) *"Extended entities"*, which contains structural information about the tweet, such as the icon and image sizes, their URLs and ids, and, therefore, it does not provide any relevant information; (iii) *"Id"* and *"Id str"* fields are automatically generated to guarantee uniqueness to the tweet; thus, they do not contain any meaningful information; (iv) *"Truncated"* contains a constant value, which is equal for each tweet in the development set; (v) *"Source"* and *"User"*: contained features pertinent to Twitter and the user profile, such as "id", "profile image URL", "friends count", which is information not relevant to our purposes. Additionally, we verified that the development set rarely contained multiple posts from the same user: this lack of information prevented the extraction of data for determining a possible positive (or negative) influence on our goals.

Table 8.2 Brief description of the metadata fields that have non-empty values in at least 90% of the given tweets.

Field	Description	Type
Created at	UTC time when this tweet was created	object
Entities	Dictionary of the entities which have been parsed out of the text, such as the hashtags	object
Extended entities	Dictionary of entities extracted from the media, such as the image size	object
Favorite count	Indicates how many times the tweet has been liked	int64
Favorited	Indicates whether the tweet has been liked	bool
Id	Unique identifier of the tweet	int64
Id str	String version of the unique identifier	object
Is quote status	Indicates whether this is a quoted tweet	bool
Lang	Indicates the language of the text (machine-generated)	object
Possibly sensitive	When the tweet contains a link it indicates if the content of the URL is identified as containing sensitive content	object
Retweet count	Indicates how many times has the tweet been retweeted	int64
Retweeted	Indicates whether the tweet has been retweeted	bool
Source	Utility used to post the tweet	object
text	Text written by the user	object
Truncated	Whether the value of the text parameter was truncated	bool
User	Dictionary of information about the user who posted the tweet	object

As for the *"Lang"* feature, since most of the tweets were in English and all the other languages were very minority, we transformed it into a binary value *"originally_en"* to state whether the language of the tweet was English. To ensure that all features would contribute equally to the loss function used to train our proposed approaches, we normalized the features *"Favorite count"* and *"Retweet count"* between 0 and 1, which we named *"favorited_norm"* and *"retweeted_norm"*, respectively. Finally, we also discarded the features corresponding to *"Favorited"* and *"Retweeted"* since the former ones subsume them.

To determine a correlation between the normalized fields: *"favorited_norm"*, *"is_quote_status"*, *"originally_en"*, *"possibly_sensitive"* and *"retweeted_norm"* and the task at hand, we built a point-biserial correlation matrix between each feature and the "ER" and "ERP" ground truth using the Pearson correlation coefficient. As

seen on the point-biserial correlation matrix from Figure 8.2, none of the features has a robust correlation with the ground truth; however, we decided to keep the fields *"favorited_norm"*, *"originally_en"* and *"retweeted_norm"* since they are the highest correlated features. This result can be correlated to the fact that highly liked and highly shared tweets usually are very informative for the users.

We expected the text written by the user (*"Text"*) and the hashtags of the tweet (*"Entities"*) to be the most informative features, which we concatenated, obtaining a single sentence. To help the training, we translated all the texts into English, tokenized the words, filtered stopwords (i.e., emojis, URLs, special characters, articles, conjunctions), and lemmatized the sentence. Finally, the sentences were transformed into a matrix using a word embedding initialized with GloVe [91] weights, transforming each word into a vector of 200 dimensions. To be processed by a neural network, the matrices generated from *text* and *Entities* have been standardized to have the same number of word vectors—sentences shorter than 30 words (the maximum length of a processed sentence in the dataset) have been filled with zero padding. As other state-of-the-art works [112], the 30x200 matrices have been fed into a Bidirectional Long Short-Term Memory (BiLSTM) network. Then, the output was concatenated with the *extra fields* and fed into two parallel fully-connected (FC) layers with a softmax classifier, one per task. In each FC layer, we used the cross entropy $H(y,\bar{y})$ as loss function, where y is the class annotation and $\bar{y}$ is the model prediction. Denoting by $H_{ER}(y,\bar{y})$, the loss function for the ER task and $H_{ERP}(y,\bar{y})$ the loss function for the ERP task, the overall loss $H_{TOT}(y,\bar{y})$ is set to be the sum of the preceding two. Finally, the outputs from the two FC networks have been thresholded (with the threshold set to 0.5, which is the typical threshold in these contexts since the output ranges between 0 and 1). The first FC layer output is the prediction for the *ER* task, while the second FC layer output, which represents the prediction for the *ERP* task, is combined with the first output through a logical AND operation. This operation prevents the network from predicting inconsistent situations, such as having Evidence of Roads Passability while there is No Evidence of Roads. A representation of the architecture is shown in Figure 8.3.

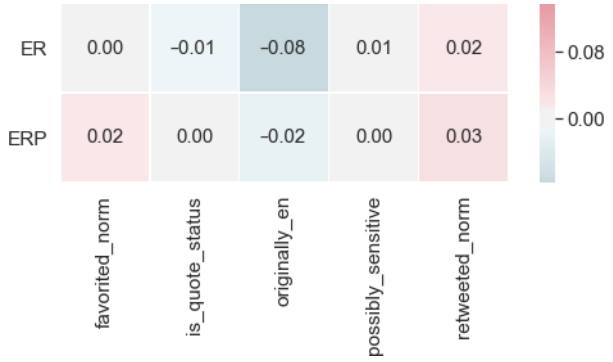


Fig. 8.2 Correlation matrix between the ground truth features, *"ER"* (Evidence of Roads) and *"ERP"* (Evidence of Roads Passability) as two different binary values, with the numerical features that were not discarded.

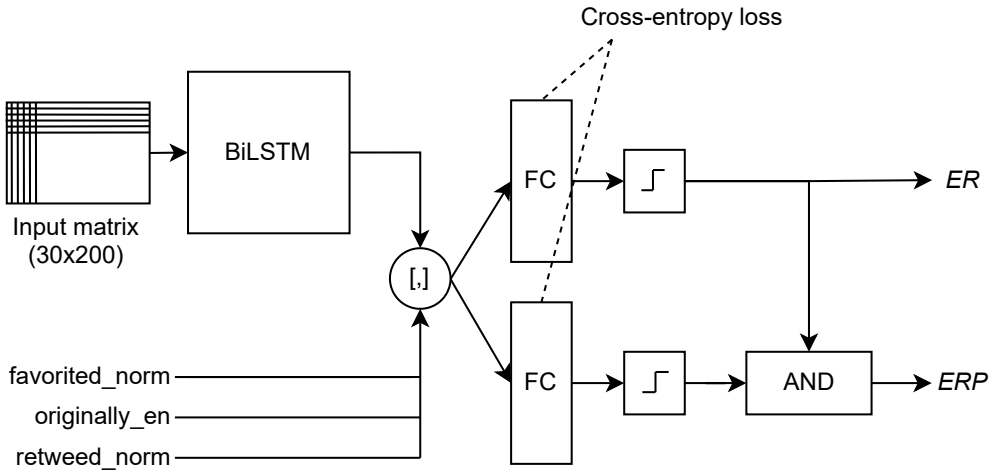


Fig. 8.3 Architecture of the neural network to process metadata. The input matrix, composed of stacked word embeddings representing the tweeted text and hashtags, is processed by a Bidirectional Long Short-Term Memory network (BiLSTM). Its output is concatenated with other metadata information representing whether: (i) the tweet has been favorited, (ii) the tweet was originally written in English, (iii) it was retweeted. Then, two Fully Connected (FC) layers are dedicated to dealing with each task: the FC on the top will determine the Evidence of Roads (ER), while the other one will determine the Evidence of Roads Passability (ERP). The classification is obtained by thresholding their output. Finally, to guarantee consistent classifications, the output of the ERP classifier is combined with the output of the ER classifier by a logical AND operation.

8.4.2 Algorithms Based on Image Only

In this subsection, we first explain the "ensemble image base architecture", which is the solution we presented in the *Flood classification challenge*. Then, we present the new proposed architecture, which will be referred to as a "double-ended network", and finally, we will introduce the extra loss that we have applied to the learning process, which will be referred to as "compactness loss".

Ensemble Image Base Architecture

For this solution, we considered both tasks as two separated, two-class classification problems. Since performance is prioritized over computation and operational time, we created an ensemble of networks, using 9 state-of-the-art networks: InceptionV3, Xception, VGG16, VGG19, InceptionResNetV2, MobileNet, DenseNet121, DenseNet201, NaSNetLarge. Since the dataset was too small to train the networks from scratch, we pre-trained all the networks on ImageNet [1], freezing the parameters from the network's first and fine-tuning the parameters from the second half. All nine models were separately trained for both classification problems: one network for road passability and one for the classification of positive or negative road passability. The networks, trained on detecting positive or negative road passability, were trained using only the images with evidence of road passability according to the ground truth. Moreover, to prevent overfitting, the dataset was randomly divided into training (75%) and validation (25%) sets. The validation set was used to prevent the networks from overfitting to the training set—after training the networks in the training set; we stored the models that performed better in the validation set. Finally, we performed cross-validation using 5 different train-validation folds to prevent overfitting while exploiting the whole dataset. Each fold was generated using a random split of the development set into 75% train and 25% validation; this means some splits overlap. Each network was trained in each fold and for each task separately, resulting in a total of 90 networks, 45 convolutional neural networks for each task (or 5 networks per network architecture and task). The network architecture is shown in Figure 8.4a.

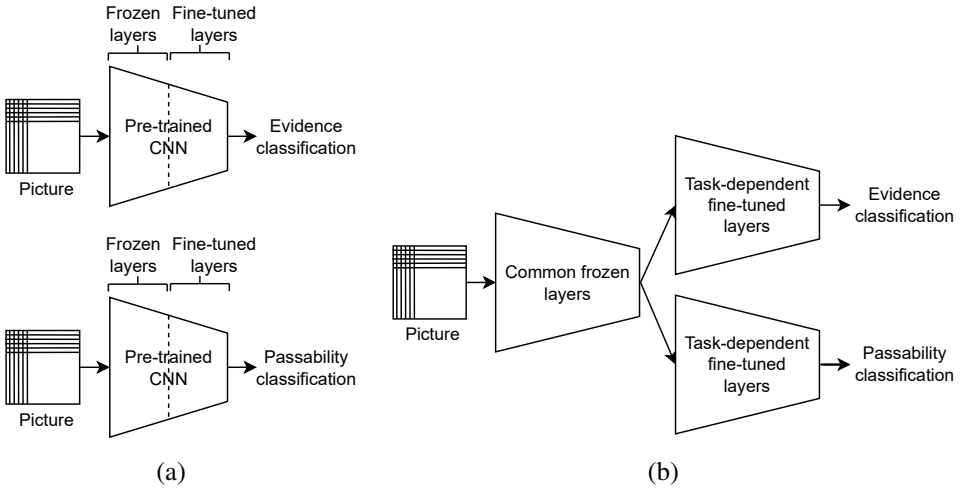


Fig. 8.4 (**a**) Schematic of the "ensemble base image architecture" which is composed of two separate pre-trained networks in which their last layers were fine-tuned on their respective task. (**b**) An equivalent architecture shares the first layers and then diverges into two separate branches for each task.

The output of each network is a number between 0 and 1, representing the probability of the picture containing evidence of road passability and whether the road has a positive or a negative passability, respectively. In order to ensemble the results of all the networks, we decided to allocate the same weight to each sub-model, and we applied two different aggregation methods. On the one hand, we applied a classical average aggregation prediction. On the other hand, we applied a combination of an average aggregation with a majority voting aggregation. These two aggregation methods are defined, respectively, by

$$\text{pred}_1(p_1,\ldots,p_n) = (\overline{p} > 0.5),$$

and $\text{pred}_2(p_1,\ldots,p_n) =$

$$\begin{cases} 1 & \text{if } \left(\overline{p} > 0.45 \text{ and } voting(p_1,\ldots,p_n) \geq \frac{n}{2}\right) \text{ or} \\ & \left(\overline{p} > 0.5 \text{ and } voting(p_1,\ldots,p_n) > \frac{n}{2} - 2\right), \\ 0 & \text{otherwise,} \end{cases}$$

where n is the number of networks, p_i is the probability given by the i^{th}-network of the picture belonging to Class 1, which corresponds to having positive Evidence of Road (ER) for the first task and having positive Evidence of Passable Road (EPR)

for the second task, and $\overline{p}$ is the average of p_i for all $1 \leq i \leq n$ and *voting* is given by $voting(p_1,\ldots,p_n) = \sum_{i=1}^{n}[p_i > 0.5]$.

Thresholding over the average of the predictions $\overline{p}$ or taking the majority vote is two largely adopted approaches to deal with ensemble model predictions [113]. However, their combination through a logical "and" tends to benefit the prediction of negative outcomes (no ER nor ERP) with the result of lowering the number of matches with the ground truth. Therefore, we added two variables x and y to weaken the constraints, defining the following function $\text{pred}_2(p_1,\ldots,p_n,x,y) =$

$$\begin{cases} 1 & \text{if } \left(\overline{p} > 0.5 - x \text{ and } voting(p_1,\ldots,p_n) \geq \frac{n}{2}\right) \text{ or} \\ & \left(\overline{p} > 0.5 \text{ and } voting(p_1,\ldots,p_n) > \frac{n}{2} - y\right), \\ 0 & \text{otherwise.} \end{cases}$$

To determine the best values for the two variables, we applied the aggregation methodology with the grid search [114] approach in the validation set: the variable x was set to range from 0 to 0.5 with a step of 0.05, while y was set to range from 0 to $\frac{n}{2}$ with a step of 1. As a result, the assignments that maximized the number of matches between the model's predictions and the dataset annotations were $x = 0.05$ and $y = 2$.

Despite being a simple and effective model, the winning solution to the challenge, this solution requires a lengthy training process as well as high computation cost and time during testing. Moreover, the solution requires a lot of storage space since parameters trained on 90 different networks are saved.

Double-Ended Network

The ensemble base image architecture relies on two networks that were trained and tested separately to solve each task individually. This architecture is represented in Figure 8.4a. However, since we are using a pre-trained network and freezing half of the model, both tasks share the first parameters of the model. Thus, we reorganized the solution as a single model where the first part of the model has the shared parameters and then diverges into two branches, each one with the specific parameters learned for each task, as represented in Figure 8.4b. This solution is equivalent to the two separate networks in terms of performance, but it is lighter, end-to-end, and computationally less expensive since we do not run the image through

the same layers twice. Starting from this idea, and knowing that in the literature, it has been stated that networks trained to perform two related tasks simultaneously can achieve better performance on both tasks than if they were trained separately [115], we decided to propose the model represented in Figure 8.5. This architecture is similar to the one from Figure 8.4b, but the division of the two branches is at the end of the last convolutional layer. After the last convolutional layer, the network is divided into two branches, one for each task with two consecutive fully connected layers. In this case, the first half of the shared parameters are frozen during training while the other half are fine-tuned jointly.

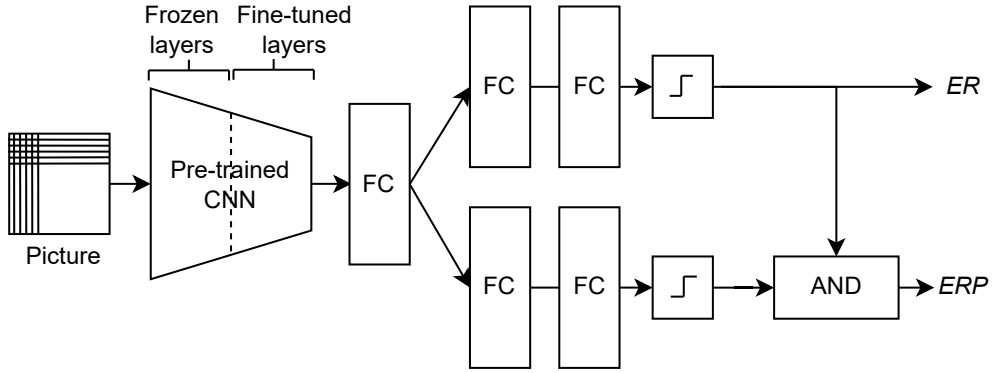


Fig. 8.5 Representation of the double-ended architecture.

To validate that, as hypothesized, both tasks are indeed related, and what has been learned for one task could benefit the other, we decided to check the activation maps triggered by the networks trained on both tasks separately. We used the gradient-weighted class activation mapping (Grad-CAM) to do so. This is a technique proposed in [4] which highlights the regions which triggered the Convolutional Neural Network (CNN) to make its classification by analyzing the activations of a convolutional layer of the network. We extracted the heatmap of the gradient activations from the last convolutional layer of the single network trained on both tasks separately. In Figure 8.6, we present the corresponding activation heatmaps for four images from the validation set. As seen in Figure 8.6, the activation scale ranges from blue to red in the images, where red corresponds to the most significant activation (MAX) and blue to the minimum one (MIN). The upper row of images corresponds to the network's activations trained on evidence of road classification, while the second row corresponds to the activations of the same images on the network trained on road passability classification. As the figure shows,

while the activations from both networks are different, the reddish area is located in a different part of the image because one focuses more on the water in general and the surroundings to determine if it corresponds to a road. The other tends to give more importance to the objects in the water to determine if the road is passable or not, and even though both networks were trained separately, their activations are highly correlated since both tasks are highly correlated. This supports our hypothesis that by training both tasks simultaneously and having both tasks share more parameters, one task could benefit from what the other has learned.

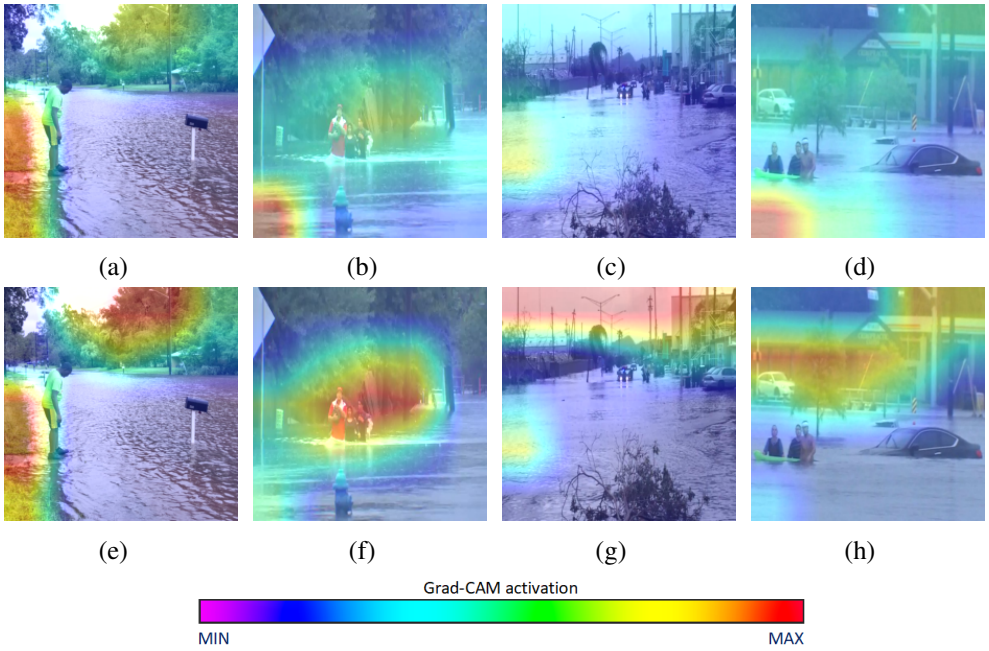


Fig. 8.6 Examples of images created with Grad-CAM. The first row (a-d) corresponds to the activations triggered by the network trained on the evidence task (ER), while the second (e-h) has the activations for the same image, of the network trained on the passability task (ERP). At the bottom of the figure, there is a scale that is used to highlight the activations of the model, where red represents a higher activation and purple means minimal activation. As seen in the first row, the activations for the evidence of the road task are maximal in the water regions. In the second row, which corresponds to evidence of road passability, there is greater activation of the elements in and outside the flooded area. This is because it is necessary to rely on these elements to identify the height of the water.

Compactness Loss

As previously explained, the problem is divided into two tasks: (i) detecting if the image has evidence of roads and (ii) if the image has been classified as containing

a road, determining whether the road is passable or not. The first task could be considered as a *binary classifier* ("evidence of road passability" and "no evidence of road passability"), but the concept of "not having any evidence of road passability" could also be subsumed by "anything which is not contained in the first class". Thus, the problem could also be considered as a *one class classification* or as an *out of distribution problem*, where "evidence of road passability" would be the class to classify (or the in distribution class). The advantage of considering the problem as a one-class classification problem rather than a binary classification problem is that one-class classification algorithms take into account that the out-of-distribution class is not only defined by the images used for the training but that it could be anything that has not previously been seen during the training phase. Similarly, the second task could be considered as a *binary classification problem* ("passable" and "impassable" roads) or as a *one class classification problem* "passable road".

For this solution, we considered both tasks as a one-class classification problem. Taking inspiration from [7], we wanted the features extracted from the first fully connected layer to be as descriptive as possible for the class, meaning that the feature representation of the class will be distinctive from the feature representation of images not belonging to that class. Moreover, at the same time, we would like a low intra-class distance, meaning that features from the same class should be as close as possible in the feature space. This optimization can be described as $\hat{g} = \max_g \mathscr{D}(g(t)) + \lambda \mathscr{C}(g(t))$, where: g is the deep feature representation for the training data t, λ is a positive constant and $\mathscr{D}$ is the *Descriptive loss function* (within this approach, we used the cross-entropy) and $\mathscr{C}$ is the *compactness loss function*, which evaluates the batch inter-class deep feature distance to derive objects from the same class. This compactness loss can be applied to the ensemble base image architecture or the double-ended architecture of the two previous models. In the first case, we would add the compactness loss to the first fully connected layer. For the double-ended architecture, we would replace the last fully connected layer with a fully connected layer followed by two fully connected layers in parallel—one for each task—and add a compactness loss for each task, as shown in Figure 8.7. The outputs of both final fully connected layers are two real values in the range (0,1). They represent the percentage of which the evidence is believed to be of roads and of passable roads, respectively. The two outputs are then rounded according to a threshold of 0.5. Therefore, the first output is the classifier for the ER class. On the other hand, to avoid inconsistent classifications (i.e., ER = false and ERP = true), the

second output is multiplied by the first, determining the ERP class's classifier. Note the decision of which fully connected layer accomplishes the ER and which ERP tasks are taken during the network training phase.

For the compactness loss, we implemented the same loss as the one proposed in [7], which is given by

$$l_c = \frac{1}{nk} \sum_{i=1}^{n} \mathbf{z_i}^T \mathbf{z_i} \tag{8.1}$$

where $\mathbf{z_i} = \mathbf{x_i} - \mathbf{m_i}$, being $\mathbf{x_i} \in \mathbb{R}^k$ the samples of the batch of size n for all $1 \leq i \leq n$ and $\mathbf{m_i} = \frac{1}{n-1} \sum_{\substack{j=1 \\ j \neq i}}^{n} \mathbf{x_j}$, the mean of the remaining samples. As it is proved in [7], this compactness loss is, in fact, a scaled version of the sample variance given by

$$l_C = \frac{1}{nk} \sum_{i=1}^{n} \frac{n^2 \sigma_i^2}{(n-1)^2}, \tag{8.2}$$

where σ_i^2 is the sample variance for all $1 \leq i \leq n$.

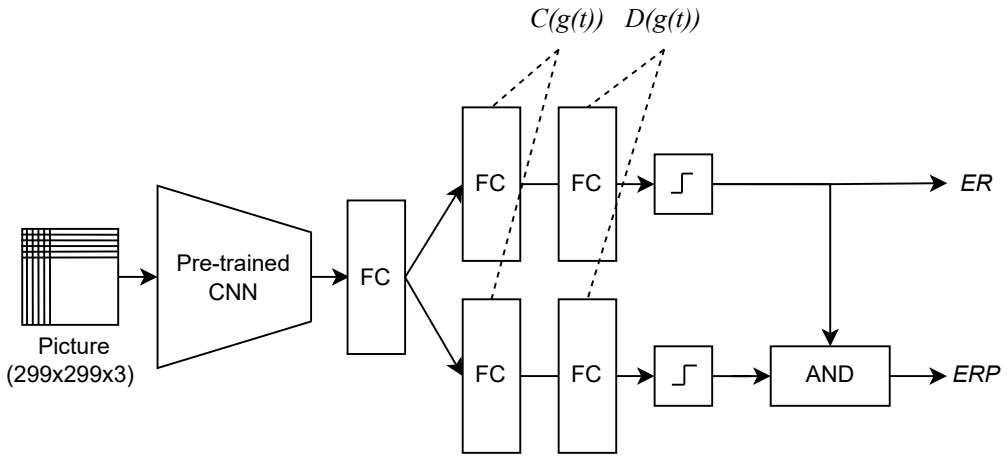


Fig. 8.7 Double-ended classifier with compactness loss. The model is based on the Inception V3 network, replacing the last fully connected layer with a 1024 fully connected layer that extracts the image features and two parallel fully connected layers, one for each task. Two losses are trained simultaneously, a compactness loss to ensure low intra-class feature distance and a descriptiveness loss to ensure a high inter-class feature distance.

In order to implement the backpropagation, we need to compute the gradient of l_c with respect to the input x_{ij}. In [7], the derivation of the backpropagation formula

obtained from the gradient of l_C with respect to x_{ij} contains a mistake. Indeed, in Appendix A in [7], it is stated that the following equation gives the gradient

$$\frac{\partial l_C}{\partial x_{ij}} = \frac{2}{(n-1)nk}\left[n \times \left(x_{ij} - m_{ij}\right) - \sum_{l=1}^{n}\left(x_{il} - m_{il}\right)\right]. \tag{8.3}$$

In Appendix A, we prove that the following equation gives the gradient:

$$\frac{\partial l_C}{\partial x_{ij}} = \frac{2}{(n-1)nk}\left[n \cdot \left(x_{ij} - m_{ij}\right) - \sum_{l=1}^{n}\left(x_{lj} - m_{lj}\right)\right].$$

8.4.3 Algorithm Based on Metadata and Visual Information

To combine the information from the metadata and the images, any of the previously proposed solutions for the image-only architecture can be combined with the metadata-only architecture by concatenating the features extracted from the bi-directional LSTM with the features extracted by the convolutional network, as seen in Figure 8.8.

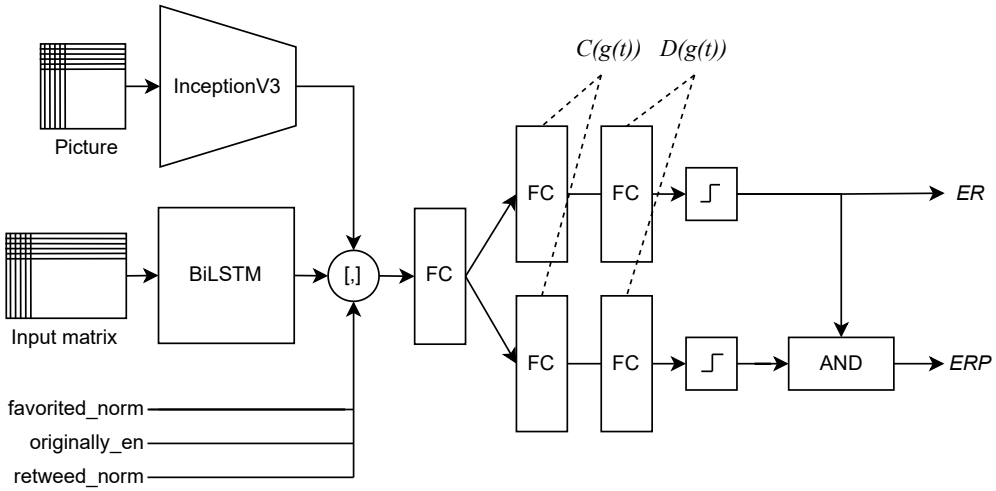


Fig. 8.8 Combination of the *Double-ended classifier with compactness loss* and the metadata system.

8.5 Evaluation and Results

As we have already commented, part of this work has been carried out for a competition in the MediaEval 2018 workshop, in which nine teams participated. In this section, we will compare not only the results of the different methods proposed in this work but we will also compare them with the results of all the other workshop participants. However, since this chapter is an extension of the work done for the competition and we do not have access to the ground truth of the test set, we had to divide the training set into training and test sets, as explained in Section 8.3. Therefore, the results given for our models will be tested on a different set than the ones for the competition. To make the comparison as fair as possible, we created a validation set from our training set to validate the models and tune the hyperparameters. The test set was only used to provide the final results.

In addition, we asked four people to perform the task on a subset of 50 images to have an understandable baseline to compare the results. These persons were external to the project but knew artificial intelligence and computer vision. They received a verbal explanation of the task along the lines of the explanation given by the challenge organizers, and they were not given any examples before starting the annotation.

All the results in this section will be given in terms of F1-score, the harmonic mean of precision, and recall. We will give the results as the average of their F1-score for the human annotators. It is important to note that the second task, the classification between passable and not passable roads, depends on the first task since if an image has been classified as not containing evidence of road passability, it will not be considered for the second task. Therefore, a false negative detection in the first task (an image wrongly classified as not containing evidence of road passability) will also count as a false negative in the second task, regardless of its ground truth. At the same time, a false positive in the first task (an image wrongly classified as containing evidence of road passability) will also count as a false positive for the second task. Due to this error propagation, the performance of the second task cannot be higher than that of the first task.

We will evaluate the results of the proposed models in this work in the same order that we have presented them in the previous section.

8.5.1 Results Using Metadata Only

In Table 8.3, the model's results using metadata information only are provided. As can be seen from the table, the performance, in general, is quite low, even in the case of human annotators. We believe this is because many of the tweets have little information about the tasks in their metadata.

Table 8.3 F1-scores on the challenge test set for both tasks using metadata information only. We understand that results on different test sets are not comparable. However, to make the comparison as fair as possible, we created a validation set from our training set to validate the models and tune the hyperparameters. The test set was only used to provide the final results. The results of the challenge are reported for completeness only. * Results are given on our own test set. ** Results on a subset of 50 images.

Approach\Data	**Evidence of Road [%]**		**Ev. of Road Passability [%]**	
	Validation Set	**Test Set**	**Validation Set**	**Test Set**
Human annotation	51.48 **	-	18.18 **	-
Metadata only	59.93	62.56 *	56.82	57.05 *
Y. Feng et al. [85]	-	-	-	32.8
M. Hanif et al. [86]	-	58.30	-	31.15
Z. Zhao et al. [108]	-	32.60	-	12.86
A. Moumtzidou et al. [87]	-	-	-	30.17
A. Kirchknopf et al. [88]	-	-	-	20

8.5.2 Results Using Images Only

In the case of images where we have proposed several models, first, we will comment and compare the results of the different models we proposed in this article and then compare them with the algorithms proposed by the rest of the participants. Firstly, we presented the "ensemble base image architecture", the algorithm presented for the competition. This algorithm is mainly based on performing iterative cross-validation to train models and ensemble them, for which we proposed a new ensembling technique. Since we have carried out a new training and test split for this work, we have retrained this architecture on the new training set and tested on the new test set to make the comparison as fair as possible. In Figures 8.9 and 8.10, we show how the F1-score evolves for both tasks as we ensemble more models and the difference between the different ensembling techniques. As seen from the curves,

both tasks benefit from the ensembling, particularly for the first networks, then the performance stabilizes. The evidence of road benefits more from this technique than the passability of road task, and the curves for the passability task are less stable. We believe this is due to the difference in the difficulties of both tasks. As can be seen towards the end of the graphs, the results of both tasks begin to worsen slowly. That is because (i) we are adding different architectures, and some of them yield better results on average than others; (ii) we have stacked the networks in order of the architectures' average performance, and thus it gets the point in which adding more architectures starts degrading the results. Given the information from both graphs, the ensemble of more than 30 models (up to 90, in our test case) does not significantly improve the performance. The ensemble of 90 networks (45 per task) was the winning architecture presented in the MediaEval competition, and its performances are presented in Table 8.4. This is the only architecture for which we have results on both the challenge and our test sets. The results of the MediaEval test set are very similar to the results obtained for our own test set, which indicates that the difficulty of both sets is quite similar, allowing us to make a fair comparison with the results of the other participants. Some differences might be since we have had to retrain all the networks to fit them to the new training, validation, and test set. As for the ensembling technique, voting seems to have slightly worsened the results in the evidence of road task, while averaging led to worsening results in the road passability task. For this reason, we decided to use the ensembling technique that we proposed in this work for the final results since it is the technique that has the most stable results.

The ensemble models make sense for competition; however, they might not be suitable for real-life applications since they tend to be computationally expensive and time-consuming. Therefore, we focused our analysis on comparing the best available model, obtained without considering computational limitations, with a lightweight version proposed in this work. We started using an "ensemble of one model per task" that we named "single-network image base image architecture", and then compared it with the "double-ended architecture" presented in the previous section. To reduce the randomness associated with the training process, both architectures were initialized with the same weights and used the same hyperparameters and stopping criterion. As seen from Table 8.4, the improvement is quite significant, particularly in the passability task. We believe that this is because the passability task has significantly fewer images to train when trained separately, so it is more difficult

for the model to generalize to new data, and when trained together, the passability task can benefit from the evidence task has learned. On top of obtaining better results than the single network base image architecture", the "double ended architecture" has fewer parameters, making it lighter and computationally less expensive, and it is an end-to-end architecture.

Then, we proposed to use the "compactness loss" to make the model more robust to unseen data. We retrained the previous model with the compactness loss on each branch, as shown in Figure 8.7. The results of both the validation and test set are in Table 8.4. Although we cannot extract direct evidence from this table that the compact loss improves the results, it does seem to generalize better to the test set since the results from validation to test are more similar than the ones without compactness loss.

Finally, by combining the improvements using the double-ended classifier and the compactness loss, we can reach almost the same performance as we had obtained using the ensemble of 30 models, meaning that we have a model almost 30 times lighter and faster with a comparable performance.

Table 8.4 F1-scores on the challenge test set for both tasks using only the images from the tweets. We understand that results on different test sets are not comparable. However, to make the comparison as fair as possible, we created a validation set from our training set to validate the models and tune the hyperparameters. The test set was only used to provide the final results. The results of the challenge are reported for completeness only. * Results on a subset of 50 images.

Approach\Data	**Evidence of Road [%]**			**Ev. of Road Passability [%]**		
	Validation Set	**Test Set** (MediaEval)	**Test Set** (Own)	**Validation Set**	**Test Set** (MediaEval)	**Test Set** (Own)
Human annotation	87.32 *	-	-	47.71 *	-	-
Ensemble image base architecture (90)	90.14	87.79	90.17	64.33	68.38	65.91
Ensemble image base architecture (30)	88.91	-	89.45	70.18	-	65.28
Single network image base architecture	86.48	-	84.88	62.84	-	59.99
Double-ended architecture	88.73	-	85.00	67.51	-	67.91
Double-ended with compactness loss	87.78	-	86.42	67.49	-	68.53
Y. Feng et al. [85]	-	-	-	-	64.35	-
M. Hanif et al. [86]	-	74.58	-	-	45.04	-
Z. Zhao et al. [108]	-	87.58	-	-	63.13	-
A. Moumtzidou et al. [87]	-	-	-	-	66.65	-
A. Kirchknopf et al. [88]	-	-	-	-	24	-
N. Said et al. [110]	-	-	-	-	65.03	-
D. Dias [109]	-	-	-	-	64.81	-
B. Bischke [107]	-	87.70	-	-	66.48	-

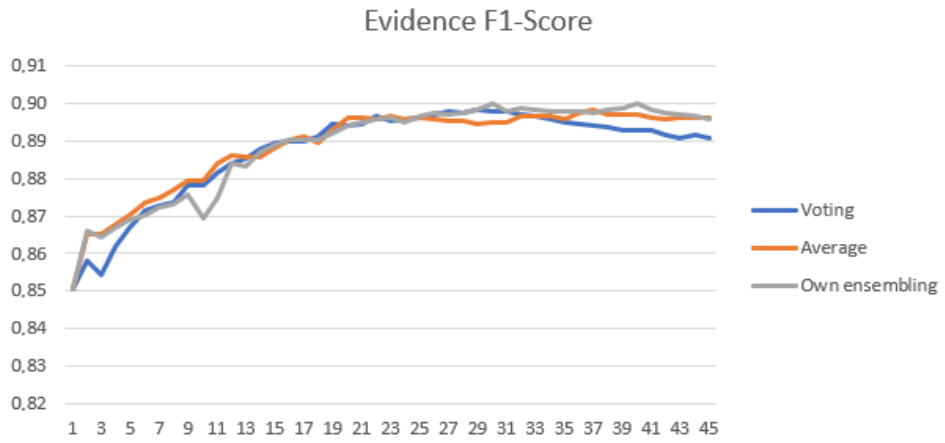


Fig. 8.9 Evolution of F1-score on the road evidence task as we ensemble more networks and compare the three different ensembling techniques.

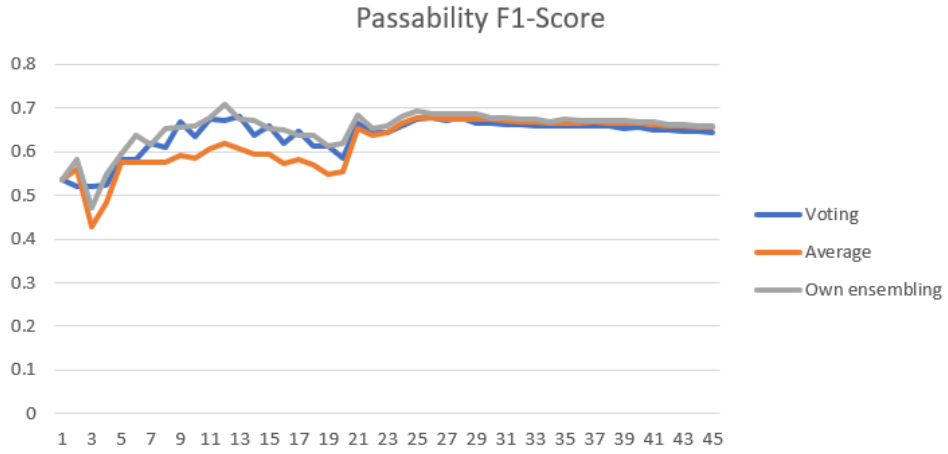


Fig. 8.10 Evolution of F1-score on the road passability task as we ensemble more networks and compare the three different ensembling techniques.

Remarkably, the results using images are considerably better than the ones using metadata, not only in our case but also for humans or other participants. That is because the dataset has been built and annotated using only visual information. Thus, we know that the images should have enough information to solve the problem, but the metadata does not always have this specific information.

8.5.3 Results Using Images and Metadata

As a final step, we combined the previous best model with the metadata information. As it was not clear from the previous results if the compactness loss provided a significant boost in performance, we tried combining the metadata with the double-ended classifier with and without compactness loss. The results are given in Table 8.5. In this case, we can notice a considerable improvement in the model with compactness loss relative to the one without it. In fact, this model achieves only 3% below the best score in the evidence of road task, while it obtains almost a 10% improvement in evidence of road passability compared to the second best participant. Finally, it seems like adding the metadata information improves the road passability task. To understand how the metadata information can help to improve the results, in Figure 8.11, some tweet examples are given that were incorrectly classified by the image-only model but correctly classified by the model, which combined visual and metadata information. These tweets contain some very informative keywords, such as flooded streets, stalled cars, and drive-through.

Table 8.5 F1-scores on the challenge test set for both tasks using the metadata and image information. We understand that results on different test sets are not comparable. However, to make the comparison as fair as possible, we created a validation set from our training set to validate the models and tune the hyperparameters. The test set was only used to provide the final results. The results of the challenge are reported for completeness only. * Results given on our own test set.

Approach\Data	**Evidence of Road [%]**		**Ev. of Road Passability [%]**	
	Validation Set	**Test Set**	**Validation Set**	**Test Set**
Double-ended architecture	78.96	86.99 *	61.06	62.96 *
Double-ended with compactness loss	77.85	84.56 *	73.61	75.93 *
Y. Feng et al. [85]	-	-	-	59.49
M. Hanif et al. [86]	-	76.61	-	45.56
Z. Zhao et al. [108]	-	87.58	-	63.88
A. Moumtzidou et al. [87]	-	-	-	66.43
A. Kirchknopf et al. [88]	-	-	-	35

Fig. 8.11 Examples of tweets within the dataset allowed to disambiguate the visual content for a correct ERP prediction. In (a-e), the text of the tweets allowed to resolve ambiguous ERP images thanks to keywords such as "flooded" concerning roads and "drive through" about vehicles.

8.6 Conclusions

We extended the work conducted for the *Flood classification challenge* in MediaEval 2018, in which we presented a winning architecture based on fine-tuning and ensembling 45 models for each task. This work, evaluates the algorithm's performance evolution as we ensemble more models. It is determined that by ensembling models in order of average performance, we can improve F1-score for both tasks. However, as we ensemble networks, at some point, adding more networks to the ensemble starts worsening the results. Thus, by reducing the number of ensembled

models to 30, we can reduce the number of ensembled networks, making the solution faster and lighter while improving the results. Then, we proposed a double-ended architecture that trains both tasks simultaneously to further reduce the number of parameters composing the solution and to let the network share knowledge between tasks. We also propose the usage of a compactness loss. Said loss is proposed to convert a binary classifier network into a one-class classification network. In the original work, this loss's derivation contained a mistake we corrected. Through experiments, we conclude that, by combining the double-ended architecture and the compactness loss, we can obtain a single network that solves both problems, achieving comparable results for the evidence of road task and better results for the road passability estimation compared to the ensemble models. Since this solution does not rely on ensemble networks, it is almost 90 times faster and lighter than the originally proposed architecture, making it a viable solution for real-life applications.

Chapter 9

Emergency scene description using deep learning approaches

This chapter presents our proposed model for the Disaster Scene Description and Indexing Challenge of TRECVIDI2020. We analyzed the best approach to use the features extracted from the images and the best blending mode. The strategies we have implemented are 1) common backbone and common classifier, 2) common backbone but dedicated classifier for each dataset macro category, and 3) dedicated backbone and classifier. Direct fusion through concatenation was the feature fusion strategy of choice. This involved combining the features extracted from the images by directly joining them. This allowed the model to learn specific features for each category and improved the overall performance of the model.

9.1 Introduction

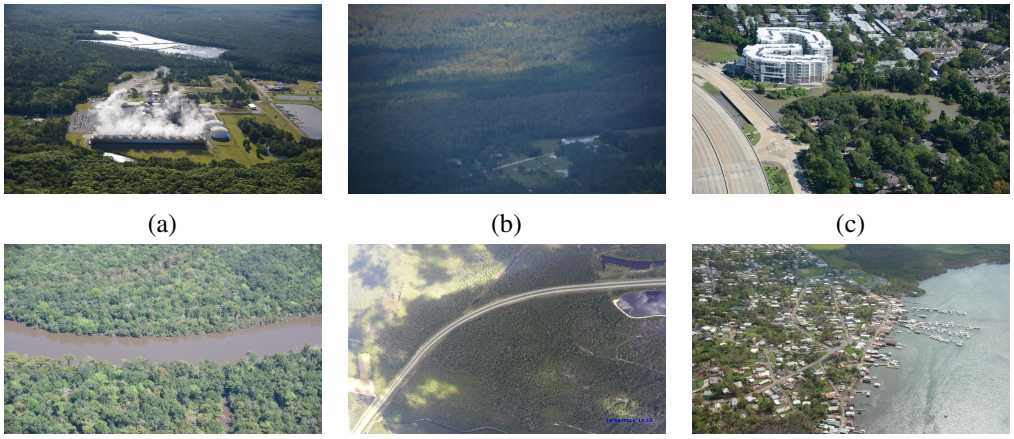
The TRECVID2020 [116] Disaster Scene Description and Indexing (DSDI) Challenge involves using a newly developed dataset to solve a multi-labelling task. The task consists of detecting all the labels detected as feasible for an emergency scenario. Labels can be grouped into five macro-categories: *Damage, Environment, Infrastructure, Vehicles, and Water*; each category is composed of different elements, each of which distinguishes a concept or an object. The peculiarity of this challenge lies in the fact that not all the classes that must be predicted consist of entities, as *infrastructure* and *vehicle* others concern more extensive elements that can also cover

all the area of the image, like the elements of the *environment* or *water* categories. Finally, in *damage*, there are primarily abstract classes that are difficult to define at the local level (it is difficult to define a bounding box that envelops all the interesting areas) and can be evaluated by a person only by analyzing the image in its entirety. These conceptual labels are also the most difficult to predict because their features can often be heterogeneous. These are the challenges this dataset poses, but if they can be overcome, the dataset can become an essential element in classifying and analyzing images of natural disasters and a helpful tool for first responders. This challenge is, therefore, the first step to better understanding the dataset and what work must be done to improve the first intervention in emergency scenarios, thanks to open data now available online.

9.2 LADI Dataset

The dataset developed for the challenge comprises images gathered by Civil Air Patrol during missions for various natural disasters. It presents various challenges due to the angle from which the images were collected. The change in the angulation is a problem because if it differs too much from the one used to train state-of-the-art neural networks, there is a great chance they will perform poorly. Usually, the images they are trained on are collected from land, not from an aerial point of view. Furthermore, the type of label of the dataset is not homogeneous. The data are divided into entities, environmental elements, and concepts. This represents another of the challenges for this dataset. Suppose some entities (house, road, bridge, truck, car, etc ...) can be easily described with features. In that case, it is slightly complicated for environmental elements that can extend over the whole image (tree, river, ocean, etc ...) to be identified since, being background elements, they can contain within them entities that are still part of the set of features for the environmental element. While in the *damage* category, there are all elements of damage to structures or other elements. However, the damage is conceptual, so even a person would find it difficult to classify and define it with a set of features. This type of label is the most difficult to classify within the dataset. Figure 9.1 presents examples of the dataset's images.

While in Figure 9.2, the distributions of the dataset are presented, in the first image 9.2a, the distributions of the dataset provided for the challenge are presented. In the second image 9.2b, the distributions for the extended dataset we have collected



(a) (b) (c)

(d) (e) (f)

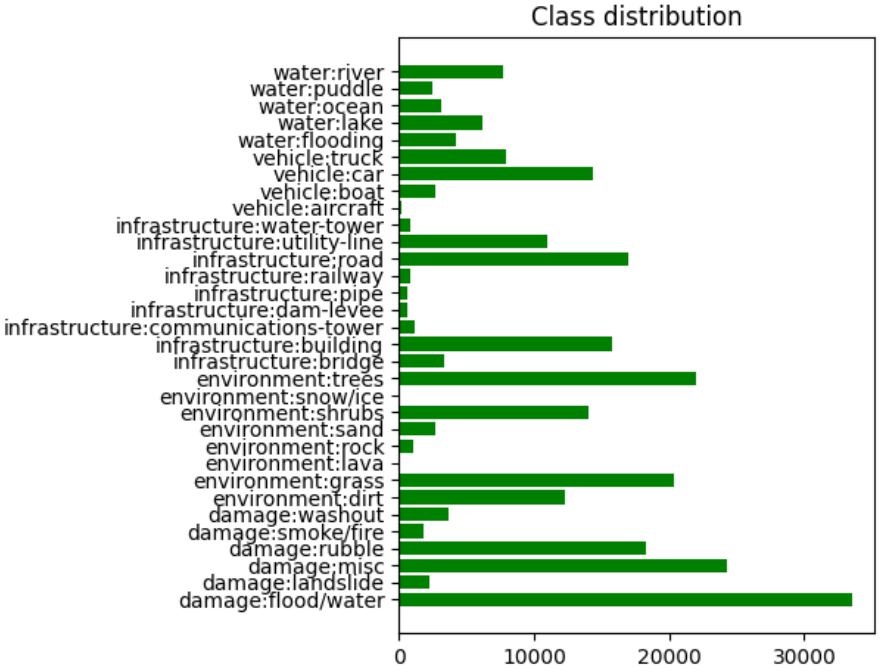
Fig. 9.1 Examples of images in the dataset. It can be noted that the lighting, orientation, perspective, and resolution vary across the examples. These changes are a key component of the LADI Dataset and are part of this dataset's main challenges.

are represented by taking a non-labelled subset of the LADI Dataset and having it labelled using Amazon Mechanical Turk as a service.

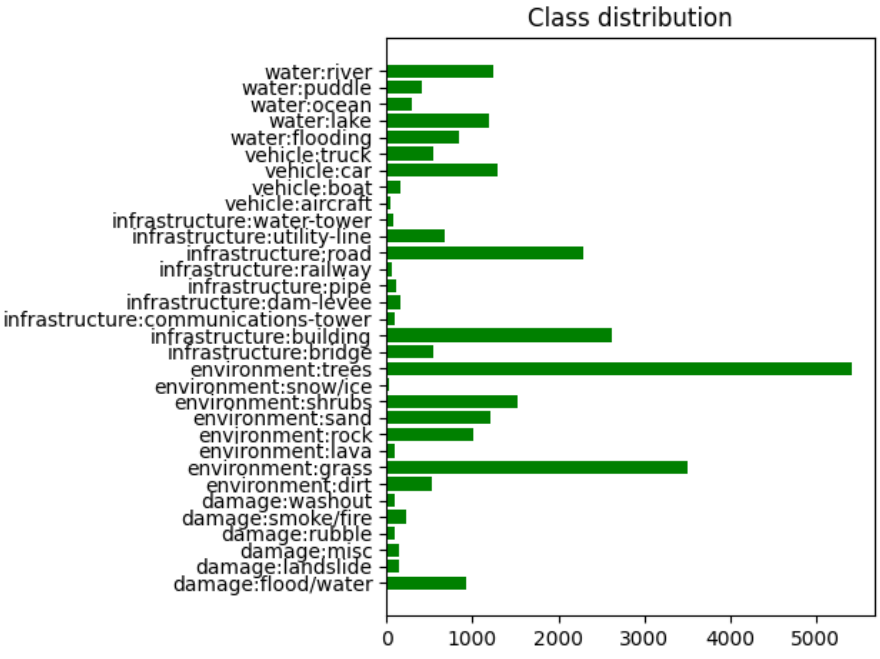
As seen from Figure 9.2, the distributions for the dataset and its extension are unbalanced in the classes. Some are predominant (trees, grass, roads), while others have few examples (lava, snow/ice, aircraft). For this reason, given the scarcity of examples for specific classes, it will not be easy to get good detections for those with very few examples.

9.3 Models

We used ResNet152 [21] as a feature extractor. ResNet allows the extraction of the features trained on Imagenet [71]. The features learned from the model with the pre-training on Imagenet are mostly features of images taken from the ground. Some of these features, although very similar to aerial ones, in some cases, may be different. For example, a tree can have similar leaf features when viewed from above, but a house or any other building can change a lot. For this reason, we trained our models from scratch. A first vanilla model, represented in Figure 9.3, consisted of using a single backbone network to extract the features of all the classes and then passing these features to a classifier capable of identifying the presence or absence of a certain class. However, this approach obtained the worst results in the validation



(a)



(b)

Fig. 9.2 We propose using Amazon Mechanical Turk as a labelling service for the different class distributions for the LADI Dataset and the dataset extension. As seen from the images, the classes are very unbalanced, and for some classes, the examples are very few.

phase. Most likely, the complexity of the dataset is too high for the model to handle. Although numerous, about 30k, the images provided during the training step were insufficient to ensure correct learning of all the dataset's features. The model used was Resnet152, a fairly complex model, able to learn features for the 1000 classes of Imagenet, so if it fails to learn the correct features for the LADI Dataset, this is most likely due to the need to have a much larger dataset. The other proposed model, in 9.4, consists of a single feature extractor and five different classifiers, one per category. This model, under validation, was able to achieve better results. Finally, as the last step, we decided to try a different approach for each category and combine the results. This model, which is represented in Figure 9.5, is the one that obtained the best results in the validation phase and is the one we have selected as the final model. In the experiments section, the implementation choices and the experiments carried out using these models will be presented in more detail.

9.4 Experiments

We tested different classifier configurations before proposing the optimal variant for the challenge. A first experiment was performed using a single network as a features extractor. In our case, we used ResNet152, followed by a classifier, a simple Multi-Layer Perceptron (MLP). Our study used an MLP as the classifier. The MLP was composed of 4 layers, with the first layer taking as input the 2048 features extracted from the ResNet model. These features were then passed through the subsequent layers of the MLP. The output of the final layer consists of all the possible labels of the prediction. Even in the testing phase, this model has yielded very unsatisfactory results. However, our purpose was to evaluate the tasks' complexity and have a baseline from which to advance our work. Following these results, we thought that maybe the problem was in the classifier, which is too simple and unable to find a correct division between classes. The inability of the model is due to its failure to both memorize and accurately discern the features of the dataset caused by its dimension and the simplicity of the model. For this reason, we have separated the classifier into five distinct networks, one for each category. Although they are very heterogeneous, within the same category, the classes are very similar; therefore, with this configuration, we have tried to make sure that each classifier is specific for the features of that category. The experimental results showed that this approach

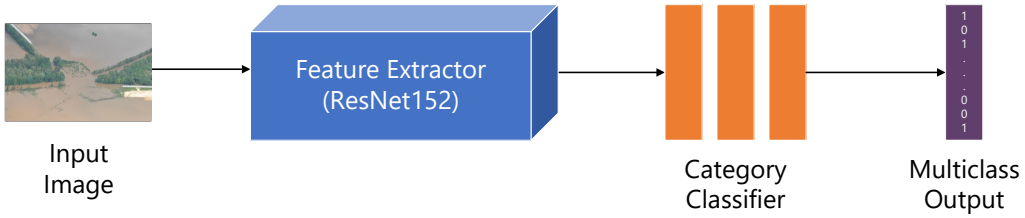


Fig. 9.3 Single classifier

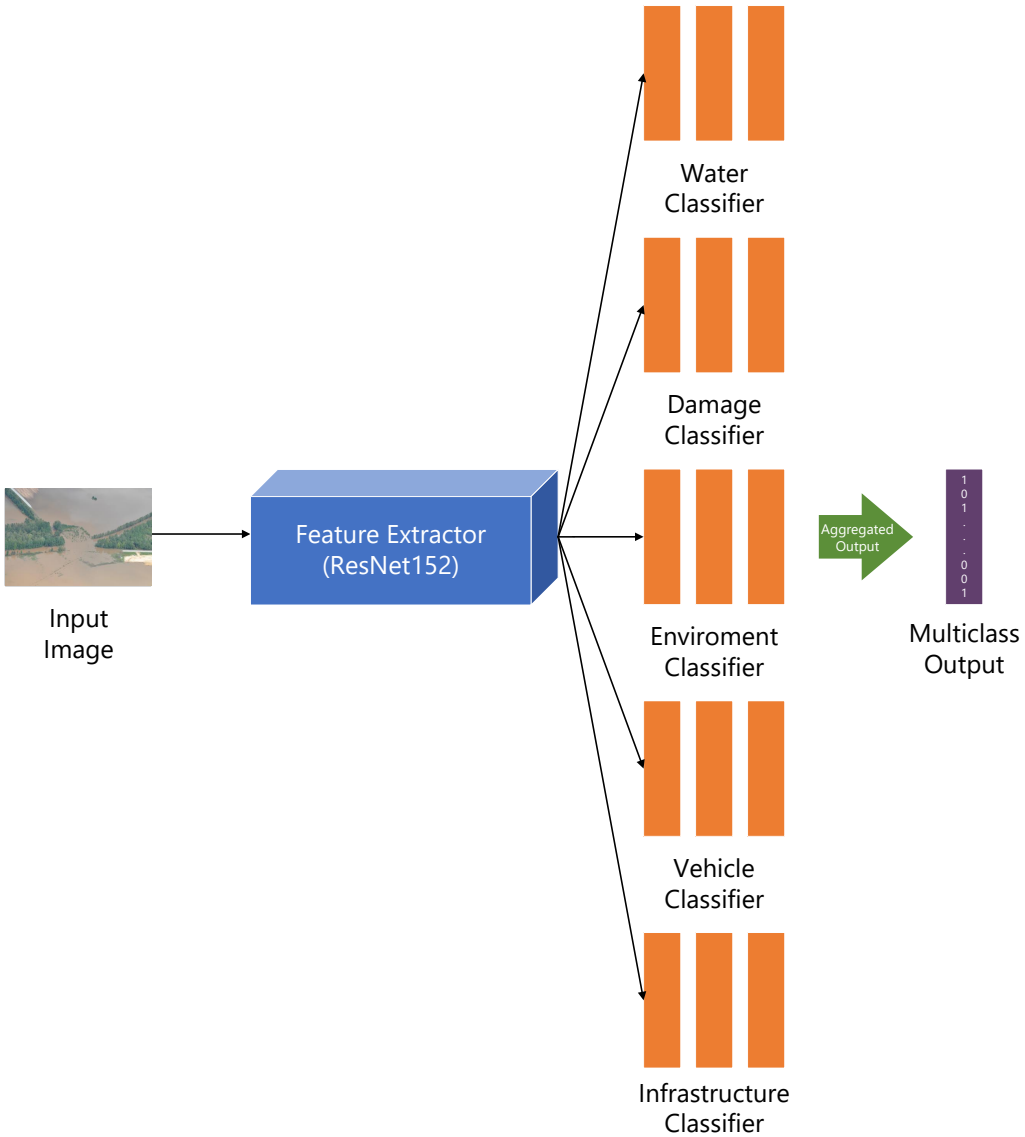


Fig. 9.4 Five classifiers

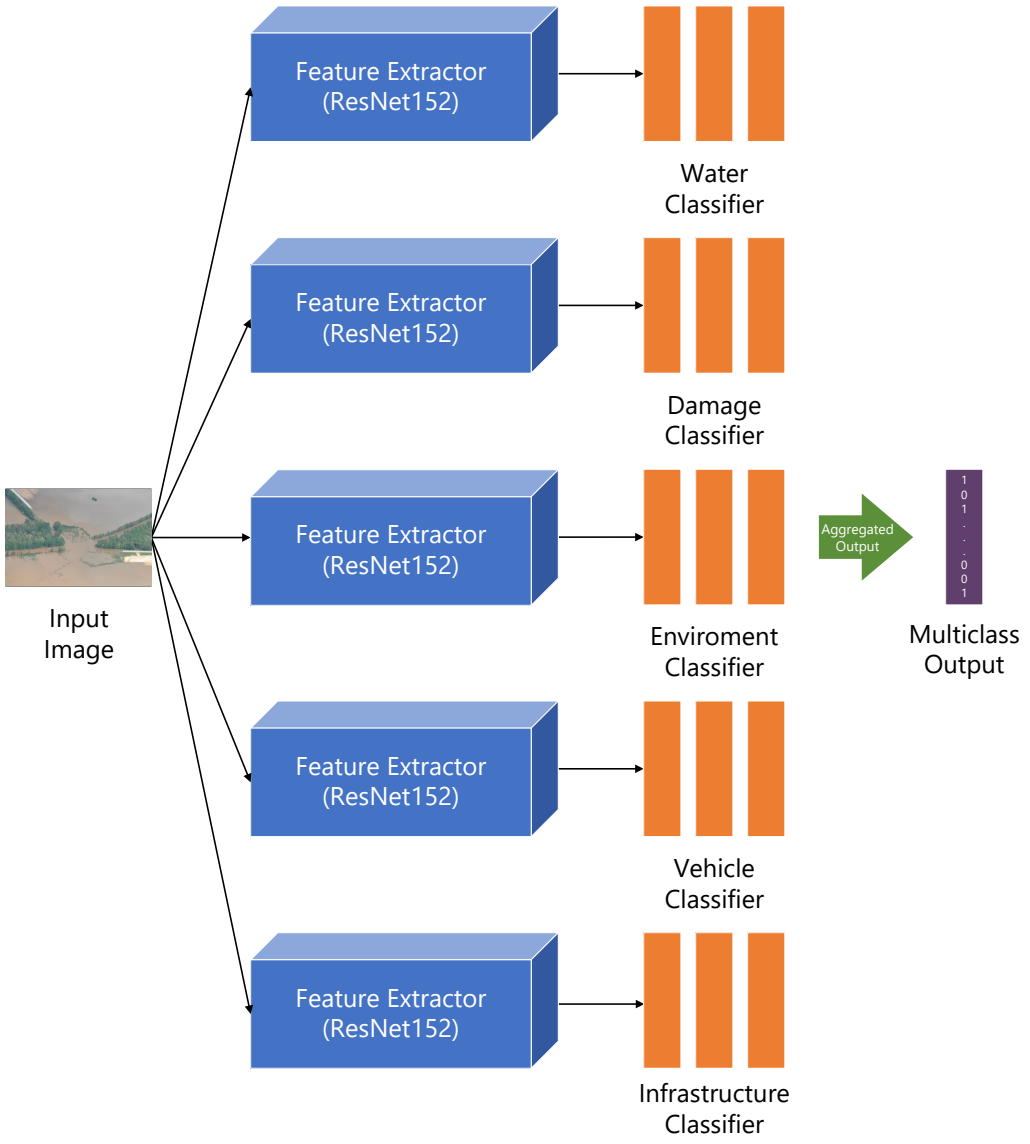


Fig. 9.5 Five Networks

was better than the previous one since the scores obtained were improved. These results prompted us to try a further split of the model, also dividing the feature extraction part, dedicating one for each category. The idea was very similar to that of the previous case. If the model benefits from having a dedicated classifier for each feature, it could also benefit from having more specific features learned for that category. This would lead the classifier to have the possibility to specialize further for that specific set of features of the category. The experimental results showed that this

configuration based on backbones and separate classifiers was the best, as the scores improved further. Having defined the best model, we wanted to try some variations in the dataset. The first variation consisted of extending the dataset with an additional 6k examples. To obtain this data, we used the Amazon Mechanical Turk platform [117] on a random set of examples that had not already been labelled. We have seen that an improvement can be achieved by extending the dataset. This suggests that given the complexity of the task, the 35k examples provided by the challenge are not enough and, therefore, that the algorithms would undoubtedly benefit from a more significant amount of data. This observation is valid for this case, and it has been shown several times that the performance of convolutional networks is also often correlated with the number of examples available during training. Another experiment that we tried, which did not produce significant benefits, but we report for the record and experimental interest, consisted in carrying out a pre-training phase using images taken by Google. The images were collected using an automatic system based on a keyword query, and the first 1000 images were downloaded per keyword. This was done using the label name as a keyword. After downloading the 32k images (1k for each class), the model backbone was pre-trained on a classification task on the 32 classes of the LADI Dataset. In this way, the network should have had a pre-training of the features of the target dataset. But, the perspective problem persists. Images perspective is from the ground and the air. Following this pre-training, the backbone network was reinserted into the five separate network models to see if this pre-training phase had given any benefit in identifying better features for the task. However, from the experimental results, this process does not influence the final scores. The results are presented in terms of Mean Average Precision (MAP) as requested by the challenge organisers. In Table 9.1, it is possible to see the experimental results obtained on the challenge test set. Figure 9.6 shows the scores obtained by each team in the challenge. Our team results for each run are highlighted in red. For each run, the training approach was the same. The only difference was the aggregation mode of the label for the test set and the last run based on a pre-training on Google Images. From left to right, the runs are the following: *1)*: 1000 video sequences, aggregated by max correspondence in the sequence; *2)*: sequences were aggregated with *mean* but filtered out with a threshold t=0.25; *3)*: the same as the previous run, but with *max* as aggregation function and t=0.5; *4)*: done with a pre-train on Google Images and by using *max* and the first 1000 sequences.

We perform the training and the testing on a desktop workstation with an Intel(R) Core(TM) i9-7940X CPU, 128 GB RAM, and 4 NVIDIA GeForce GTX 1080 Ti.

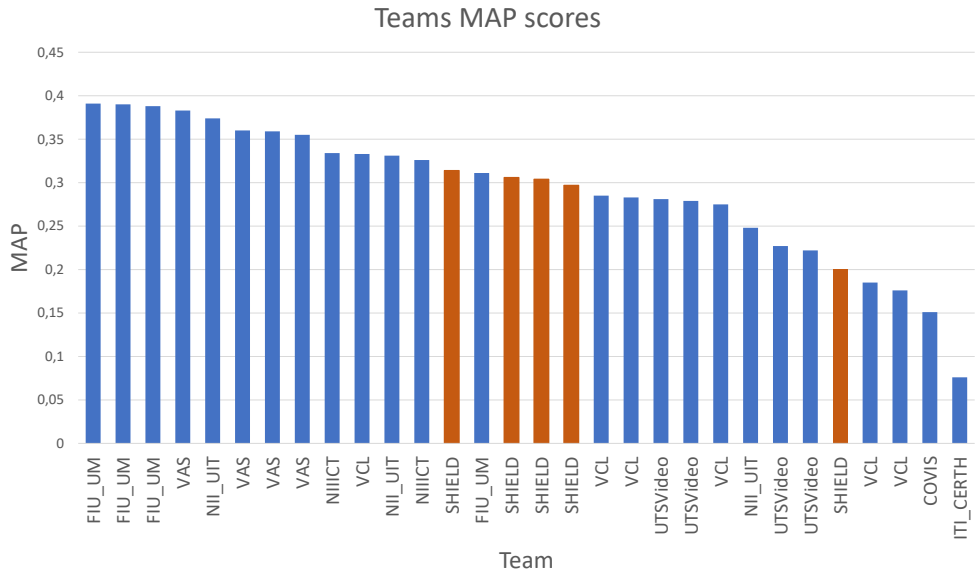


Fig. 9.6 Team scores in terms of MAP, our team results for each run are highlighted in red.

Model	Training Dataset	MAP score
Single Classifier	LADI + MTurk LADI	0.232
Five Classifiers	LADI + MTurk LADI	0.258
Five Networks	LADI	0.306
Five Networks	LADI + OTHER	0.297
Five Networks	**LADI + MTurk LADI**	**0.314**

Table 9.1 Best scores obtained during training

9.5 Conclusion

This work presents the solutions we have used to address the multi-labelling problem on the LADI dataset and an extension of the dataset that we created using crowd-sourcing platforms. After trying various settings, we found that the best solution

consisted of a model based on five different classifiers. This analysis pipeline is the best we analyzed because it allowed us to divide the features by categories and analyze them correctly for multi-labelling. Furthermore, we found that more samples provided by our dataset extension allowed the solutions to perform better. We think that future work on this dataset could have two directions; first: it is to improve the dataset labels by adding more refined ones, which could cost more resources to create, but at the same time, would allow using more sophisticated algorithms. One type of these labels could be segmentation maps. This would enable learning the areas within the image and which semantic features belong to which area. If this task proves too expensive, the segmentation maps could be limited to background elements, such as water and environment. At the same time, simple bounding boxes could be used for the other categories, which in any case, would allow for localized learning of the features for that particular class.

9.6 Acknowledgements

The European Commission partially funded this work through the FASTER project, grant agreement n.833507, SHELTER project, grant agreement n.821282, and SAFERS project, grant agreement n.869353.

Chapter 10

Water segmentation for flood detection and monitoring

In this chapter, we compare recent Deep Learning algorithms' accuracy and prediction performances for the pixel-wise water segmentation task. Many of these models have a standard feature extractor combined with subsequent segmentation layers. These layers use a flow feature fusion methodology to connect information from different resolution scales. Thanks to this feature fusion approach, it was possible to obtain good results in the segmentation phase, and we also presented the model that best suits our scenario. Moreover, we release a new dataset that enhances well-known benchmark datasets for multi-class segmentation with specific flood-related images taken from drones, in-field observations, and social media.

10.1 Introduction

In this work, we study Deep Learning models' accuracy and prediction performances for detecting water in images. Specifically, we target the pixel-wise semantic segmentation of images to separate water elements from everything else, which we consider as background. This approach cannot only understand if a given image contains a water element but also quantify the number of water pixels and delineate the shape of water elements. These capabilities enable multiple use cases, span river monitoring through a fixed camera, flood mapping from drones, data validation in case of crowdsourced data collection activities, and social media analysis. Note that

image water segmentation can be effectively used within early warning systems, which can be triggered to send alerts when a certain threshold of water is reached, as well as in the emergency response phase, to monitor the flood evolution. However, existing machine learning models trained for the semantic segmentation task may perform poorly in detecting water in a heterogeneous scenario. The images to be analyzed could be produced at different resolutions, angles, lighting conditions, altitudes, and distances from the observed phenomena.

The contribution of this work is two-fold. First, we introduce a new dataset that increases the water segmentation performances when images from different sources are considered. As highlighted [118], the more data the model is trained on, the better it can learn the patterns and features relevant to the task. This can lead to increased water segmentation performances when images from different sources are considered, as the model can better generalize to new examples it hasn't seen before; second, we study the accuracy and prediction performances of the most recent and advanced Deep Learning models for the semantic water segmentation tasks with and without the use of our dataset.

We first review the related works in the following sections and then introduce our new dataset. Next, we describe the methodology we applied in our analysis, briefly explaining the selected Deep Learning algorithms. Then, we present the evaluation outcomes, comparing the performances of the selected algorithms when using previous datasets and ours. Finally, we outline the main conclusion of our study.

10.2 Related Works

Social media platforms offer the possibility of quickly obtaining many data about natural disasters- However, information extracted from open systems such as social media platforms would require validation before being trusted and operationally used in emergency management contexts. For this reason, previous works focused on using Artificial Intelligence to analyze and classify this kind of data [119–121]. However, most such works mainly use the text of social media posts to implement a binary classifier that retains only informative content. Another kind of previous work can be found in the computer vision domain [122, 123], where the use of a machine learning algorithm with handcrafted features is coupled with image

processing techniques such as background subtraction, morphological operators, and color probability, light intensity, texture or color clustering. The shortcomings of such approaches are mainly linked to the use of handcrafted features, which usually tend not to generalize well and to perform poorly when applied to data coming from different data sources or contexts. Moreover, comparing the majority of such works is often impossible because they were evaluated in particular scenarios and with non-publicly available datasets.

Our goal is different, and we want to study the performances of recent Deep learning models for the pixel-wise water segmentation task in order to allow the automatic analysis of images coming from multiple sources, namely from social media, surveillance cameras placed near the water beds, drones, and from in-field operators. Moreover, we do not rely on handcrafted features but apply a data-driven semantic segmentation approach with automatic feature selection. Semantic segmentation is a common task in image processing and analysis, and it consists in assigning to each pixel a label, thus obtaining a set of regions in output. Image segmentation can be used to separate the foreground from the background or cluster pixels regions based on common properties, such as color or shape. Therefore, we compare the performances of recent Deep Learning models used for multi-class semantic segmentation [124–129] trained on well-known benchmark datasets, showing how their performances can be increased through a dedicated dataset designed to fit the water segmentation task. We also include the Tiramisu model in our performance analysis, which resulted in the best algorithm for the water segmentation task when trained a specific dataset containing riverine flood images [130]. We compare the classification accuracy and the prediction performances of four state-of-the-art Deep Learning algorithms using different combinations. We train such algorithms on a novel dataset we create by extending a collection of previous benchmark multi-label datasets with a set of new labelled images covering the abovementioned data sources. Our new dataset, as well as the tested algorithms, are presented in the following sections.

10.3 The Water Segmentation Open Collection Dataset

This section introduces the Water Segmentation Open Collection (WSOC), a new dataset that effectively trains deep Learning water segmentation algorithms. WSOC

Table 10.1 Water Segmentation Open Collection (WSOC) key metrics.

Metric	*Value*
Min size of image	[147, 150] px
Max size of image	[2448, 3264] px
Mean size of image	[612, 465] px
Min percentage of water	10%
Max percentage of water	95%
Mean percentage of water	35%
Number of images	120061
Number of images with water presence	11900

is composed of a collection derived from pre-existent publicly available datasets for image segmentation, which have been binarized to be used for the water segmentation task and enhanced with an additional dataset specifically created to increase the performance of the water delineation task. The public collection is composed by well-known benchmark multi-class segmentation datasets, namely COCO [131], the Semantic Drone Dataset [132], the Microsoft Research in Cambridge v2 (MSRC v2) [133], as well as by other open datasets containing water-related classes (e.g., sea, shore, river, lake, etc..), namely the Video Label Propagation [134] and the River Dataset [135]. We extend this aggregated dataset with additional 490 images, which we gather from Twitter and online news using "floods" as a keyword for the search query. We label the new images using human annotators with the aid of a segmentation tool as well as by using crowdsourcing online annotation services. For each image, we select three human annotators: the first to realize the initial annotation, the second to refine it, and the third to validate it. In case the final validation fails, the labelling is discarded and re-assigned. Overall, the WSOC dataset comprises 120061 images and their labelled ground truth, consisting of binary masks where zero is assigned to the background pixels and one to the water pixels. We openly publish WSOC at the following link: https://zenodo.org/record/3642406 so that it can be further extended and used in future works.

In Table 10.1 we report the key WSOC metrics, including the amount of water in the images and the variation in image size. The biggest images are the drone aerial images, while the smallest ones are the pictures shared on social media.

10.4 Methodology

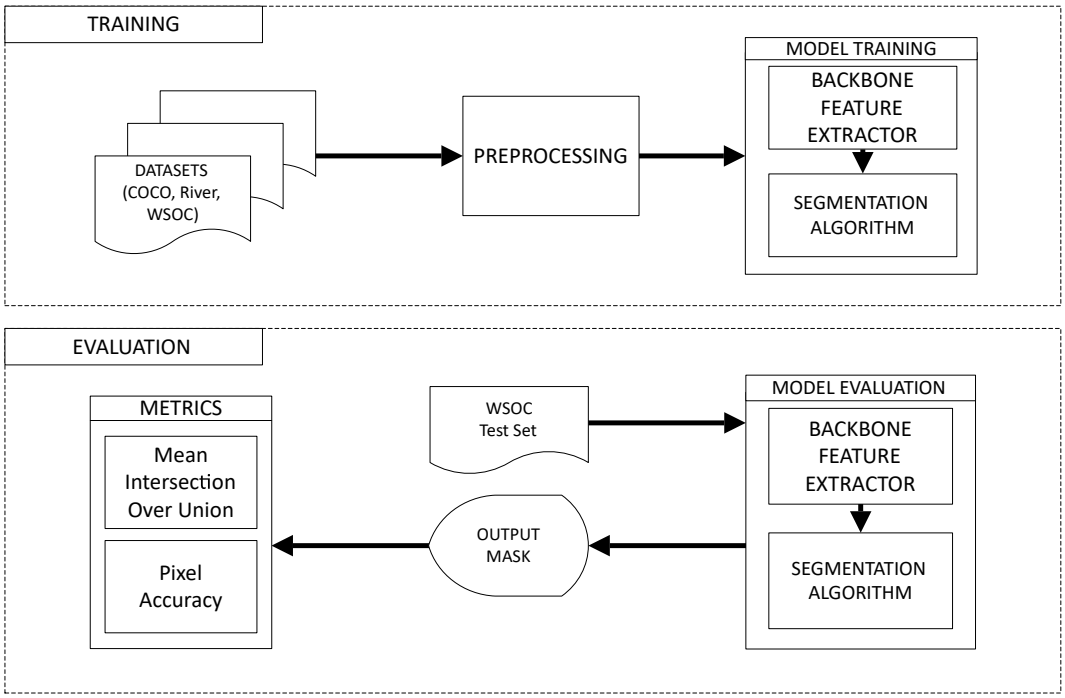


Fig. 10.1 A graphical example of our proposed methodology: images inside the dataset are preprocessed and then used to train the model, composed of a backbone network responsible for the features extraction, and a segmentation algorithm uses the features to create a segmentation mask. Then the trained model is evaluated on Mean Intersection over Union (MIoU) and Pixel Accuracy (PA). Those metrics are used to score the model performances.

We decided to test WSOC with four different Deep Learning algorithms for image segmentation, testing each of them with different backbones, namely: VGG16 [136], ResNet50 [137] and MobilNet [138]. A backbone is the initial part of a deep learning model for image segmentation and is usually used for feature extraction. A suitable feature extractor is crucial because it has been demonstrated that high-quality features enable better performance on many machine learning tasks [139]. We selected VGG16, ResNet50, and MobilNet because they are among the most recent and performing models for image classification that allows being re-trained and specialized for semantic the segmentation task by substituting the last classification layers with other Deep Learning models. Therefore, we use such backbones to exploit the features they learned on another dataset. VGG16 is a classic CNN, one of the first Deep learning networks that outperformed classic machine learning algorithms on image classification. Instead, ResNet50 is a more recent model that

uses a new kind of connection between subsequent layers called *residual block*, which has been proven to increase the performance on very deep models by allowing better information transfer through layers. MobileNet is a simplified CNN that has been designed to be used in embedded systems at the cost of a slight performance loss. The backbone is usually pre-trained on well-known benchmark dataset such as ImageNet [140], COCO [131] or PascalVOC [141]. We used all the backbones mentioned above pre-trained on ImageNet, because it contains 1000 classes, and therefore it is able to create very rich and different features. For this reason, models pre-trained on ImageNet are considered good feature extractors. We evaluate each backbone in combination with recent Deep segmentation models designed for semantic segmentation, namely: SegNet [125], PSPNet [126], UNet [124] and FCN32 [142]. SegNet is a fully convolutional autoencoder [143], where each encoding layer is connected with his decoding layer with which pooling information is shared. This particular architecture allows the model to share higher-resolution information between layers, leading to better segmentation accuracy. The UNet was developed for biomedical image segmentation, and its architecture was created to work with few training images without sacrificing segmentation accuracy. Its name is due to its U-shaped architecture, where the network uses connections between layers to propagate context information to higher resolution layers. PSPNet uses a pyramid pooling module to transfer information from the higher layer to the lower one. The pyramidal structure aims at passing additional contextual information between layers. FCN32 uses a deep, fully convolutional network that combines semantic information from coarse layers with outputs of fine layers to produce accurate and detailed segmentations. Our goal is to test and evaluate all the aforementioned segmentation networks with different backbones and compare them with Tiramisu, which has been previously trained and evaluated for the water segmentation task using only the River Dataset. We evaluate each model combination based on accuracy, prediction speed, and memory usage (size and VRAM).

10.5 Evaluation metrics and configurations

We divided the WSOC dataset between training and test sets that we use to train the models and evaluate their performances. The training set is composed of all WSOC images, including at least one water pixel (11900 images), excluding the test

Table 10.2 Model accuracy comparison in terms of average and standard deviation of Mean Intersection over Union (MIoU) and Pixel Accuracy (PA) with different training datasets (COCO, River, and WSOC) and with WSOC test set.

	COCO				River				WSOC			
	MIoU [%]		PA [%]		MIoU [%]		PA [%]		MIoU [%]		PA [%]	
	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
Tiramisu	0.35	0.20	0.65	0.22	0.33	0.19	0.63	0.22	0.38	0.17	0.73	0.24
SegNet-M	**0.80**	**0.10**	**0.90**	**0.05**	0.64	0.13	**0.75**	**0.17**	**0.85**	**0.07**	0.92	0.05
UNet-M	0.76	0.12	0.86	0.08	0.64	0.13	0.75	0.20	0.83	0.09	**0.92**	**0.03**
FCN32-M	0.62	0.15	0.73	0.17	**0.67**	**0.13**	0.74	0.17	0.77	0.11	0.86	0.08
PSPNet-M	0.75	0.12	0.88	0.08	0.62	0.13	0.74	0.19	0.81	0.08	0.90	0.07
SegNet-R	**0.82**	**0.10**	0.90	0.05	**0.74**	**0.12**	**0.83**	**0.13**	0.85	0.08	**0.94**	**0.01**
UNet-R	0.81	0.10	**0.92**	**0.01**	0.55	0.14	0.69	0.23	**0.88**	**0.06**	0.94	0.03
FCN32-R	0.79	0.12	0.75	0.12	0.59	0.13	0.74	0.19	0.79	0.10	0.87	0.08
PSPNet-R	0.81	0.10	0.90	0.04	0.62	0.12	0.74	0.22	0.83	0.08	0.90	0.07
SegNet-V	**0.71**	**0.16**	0.87	0.05	**0.69**	**0.13**	**0.79**	**0.15**	0.82	0.09	0.90	0.05
UNet-V	0.49	0.27	0.68	0.13	0.30	0.03	0.59	0.31	0.50	0.26	0.67	0.15
FCN32-V	0.67	0.19	0.90	0.03	0.58	0.16	0.70	0.19	0.76	0.13	0.91	0.02
PSPNet-V	0.61	0.23	**0.91**	**0.02**	0.12	0.10	0.61	0.38	**0.82**	**0.09**	**0.92**	**0.02**

X: Best score model-wise X: Best score metric-wise

Table 10.3 Models performance comparison in terms of VRAM, size and prediction speed.

	ResNet50			MobileNet			VGG16		
	Size [Mb]	VRAM [Mb]	Prediction time [ms]	Size [Mb]	VRAM [Mb]	Prediction	Size [Mb]	VRAM [Mb]	Prediction time [ms]
SegNet	**57**	**2461**	27	**21**	**2665**	**9**	**44**	**1193**	22
UNet	62	2563	29	24	2674	10	47	1792	21
FCN32	1732	4188	250	867	3255	115	516	3334	56
PSPNet	114	2557	**24**	37	4352	11	65	1824	**16**

ones, which we select as the 10% (139 images) of a WSOC subset composed of the following categories: the River, the Semantic Drone Dataset and the newly added images. We select this subset as the test set because such images represent the flood monitoring scenarios we are considering most. We use only images containing at least one water pixel because this configuration leads to the best accuracy, although the test set contains images with no water element. The models have been trained using Tversky loss function [144] with $\alpha = 0.2$ and $\beta = 0.8$. Note that a high value for β leads to higher sensitivity (recall) and to lower specificity [145], which is the desired condition for our task because we want to predict as many water pixels as possible while accepting lower performance on the background ones. As optimizer was used *Adadelta* with the following parameters: $lr = 0.001$ and $\rho = 0.95$ [146]. In order to avoid over-fitting, we stop the training as soon as the loss computed on the validation set ceases to decrease for five epochs. We evaluate the results of each

algorithm on the test set with the two most commonly used metrics for segmentation tasks [147, 148], namely the Mean Intersection over Union (MIoU):

$$MIoU = \frac{\frac{1}{k}\sum_i^k n_{ii}}{t_i + \sum_j^k (n_{ij} - n_{ii})}$$

and the Pixel-wise Accuracy (PA):

$$PA = \frac{\sum_i^k n_{ii}}{\sum_i^k t_i}$$

where n_{ij} corresponds to the number of pixels from class i that have been wrongly classified as belonging to class j, n_{ii} represents the pixels from class i that have been correctly classified, k is the total number of classes (k=2 in our case, because we are only trying to predict water pixels in the scene) and $t_i = \sum_j n_{ij}$ is the total number of pixels belonging to class i. We calculated the mean value and the standard deviation for each of these metrics. We perform the training and the testing on a desktop workstation with an Intel(R) Core(TM) i9-7940X CPU, 128 GB RAM, and 4 NVIDIA GeForce GTX 1080 Ti.

10.6 Results

In Table 10.2 we report the results in terms of prediction accuracy, where we highlight the best model both metric-wise and model-wise. Such results are obtained by scoring all the models on the same test set while varying the training set. Each model is scored in terms of Pixel-wise Accuracy and Mean Intersection over Union. Model-wise, SegNet has the higher number of best scoring entries, although sometimes it was slightly worse than the UNet. Metric-wise, the models with ResNet50 as backbone result consistently as the best performing. This increase in performance is due to the residual layers of ResNet50, which, as reported in the original paper, allow a better backpropagation of the gradient during training, thus increasing the model's accuracy. Moreover, model-wise and metric-wise, the models trained with WSOC achieve increased accuracy against the training done with COCO and the River Dataset. Moreover, the Tiramisu model, which was our reference benchmark, achieved by far the worst performance.

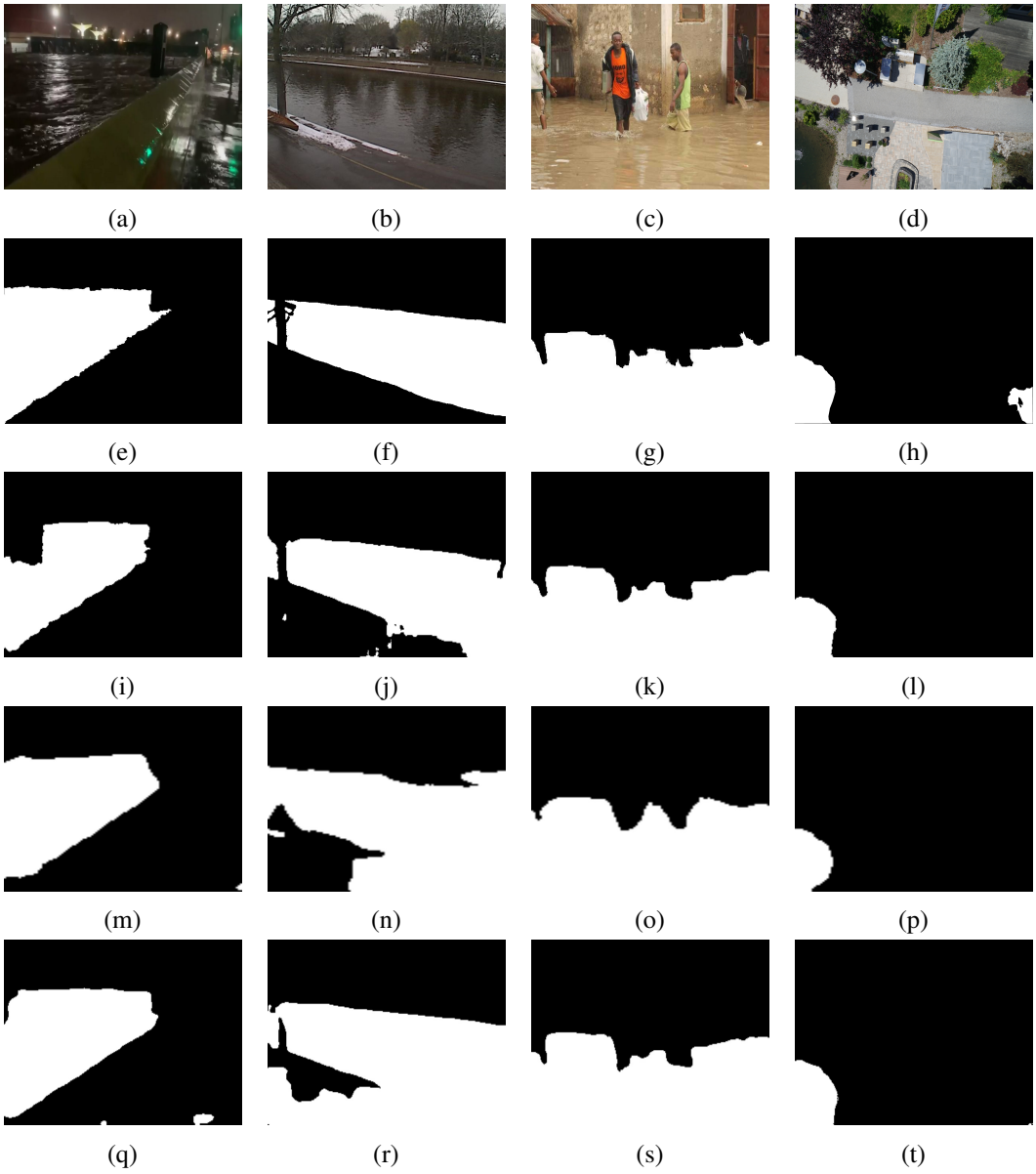


Fig. 10.2 Results of the best-performing algorithms on the best sample of the validation set, where (a-d) corresponds to the original images, (e-h) are the ground truths, (i-l) SegNet with ResNet50 as the backbone, (m-p) SegNet with MobileNet as the backbone, (q-t) UNet with ResNet50 as the backbone.

In Table 10.3 we present the model's performance in terms of model size and prediction time for one image. Such results depict a different situation, where SegNet is clearly the lightest model in terms of size (memory usage) and is also the fastest when coupled with MobilNet as the backbone. This combination (MobileNet plus

SegNet) is the winning one in size, VRAM, and speed, being three times faster compared with ResNet50 as the backbone.

Overall, we can state that the ResNet50 backbone is the best in terms of accuracy. While we do not have a unique winner model-wise, UNet and Segnet are the best options, provided that there are no particular needs regarding memory and prediction speed. Whenever the application requires faster prediction and lighter models, able to get good results in a shorter time, SegNet with MobileNet is the best choice. Moreover, we note that the introduction of WSOC increased the performances of all models in the considered test scenario.

Figure 10.2 shows representative images from the test set (first row) with their relative ground truths (second row) and the predicted binary masks (third, fourth, and fifth rows) of the following backbone-model combinations: SegNet-ResNet50, SegNet-MobileNet, and Unet-ResNet50. In Figure 10.2 (a) is taken from Twitter, (b) from the River dataset, (c) from online news and (d) from the drone dataset. As it can be seen, the models that use ResNet give more fine-grained masks, whereas results with MobileNet are slightly coarser. Specifically, it can be observed that in (k) and (s), people's legs are clearly defined; the same applies in (j) and (r), where the shape of the structure of the tree is at least sketched, although it is not completely identified.

10.7 Conclusion and Future Works

In this chapter, we presented and released a new dataset for water segmentation (WSOC), enhancing a previous collection of publicly available datasets composed of water-related labels (river, sea, waves, waterdrops, etc...). This collection of data is helpful for the task of water segmentation, in particular for implementing services aimed at flood event monitoring and information validation. Moreover, we studied several state-of-the-art Deep Learning models for semantic segmentation, combining different backbones used for image labeling with semantic segmentation models. We trained and tested all models with WSOC, comparing their performances against training done with previously available datasets. *SegNet* ad UNet, both with ResNet as the backbone, performed slightly better than the other algorithms in terms of accuracy, while SegNet was the best performing in terms of prediction speed and memory usage, VRAM, and size. Considering accuracy, ResNet was the best

performing backbone overall, but it was also the slowest. Therefore, the model used for the water segmentation task should be selected according to the application requirements regarding accuracy and prediction speed.

Future works will include creating an operational service to monitor flood-prone areas and support social media analysis. Moreover, we plan to improve the dataset by evaluating the possibility of using synthetically generated data.

10.8 Acknowledgements

The European Commission partially funded this work through the FASTER project, grant agreement n.833507, and SHELTER project, grant agreement n.821282.

Chapter 11

Conclusions

In this thesis, we analyzed some possible methodologies of feature fusion and then explored some application use cases to evaluate their effectiveness. In *Chapter 2*, we have introduced the advantages of Deep Learning over classic Machine Learning in the context of Computer vision. We compared the difference between data-driven and hand-crafted features, highlighting how the former is much easier to obtain thanks to modern technologies and how they fit perfectly into the learning pipeline to obtain an end-to-end trainable method that includes feature fusion from different modalities. Subsequently, we introduced the concept of deep multimodal learning, explaining how it is necessary for some applications and highlighting how feature fusion techniques are essential for constructing an architecture for solving multimodal tasks.

In *Chapter 3*, we introduce the automotive field and related tasks in computer vision. Specifically, we delve into synthetic datasets for training and evaluating models for tasks such as vehicle speed and distance estimation and pedestrian trajectory prediction in the context of autonomous vehicles. We discuss the benefits and limitations of synthetic data and the various methods and techniques used to generate these datasets. Specifically, in *Chapter 4*, a multimodal model was presented that combines the semantic features contained within the bounding box of a vehicle with those of its spatial location. This combination of spatial and semantic features made it possible to estimate the distance of the vehicles. Similarly, the vehicle's speed was obtained, replacing the semantic input, providing as data no longer the content of the RGB image but the optical flow of the same area. In

Chapter 5, we proposed our method for augmenting synthetic trajectories at training time with an adversarial approach and custom loss function as a promising solution for addressing the challenges of using synthetic data for training predictive models in pedestrian trajectory prediction. It allows the model to learn better from real-world and synthetic data, resulting in improved predictions, and it can address the limitations of using only real-world data and provide a diverse set of training data. Our proposed adversarial loss allows synthetic data to be used effectively by merging it with real-world data in a common feature space, resulting in improved predictions for pedestrian trajectory prediction.

In *Chapter 6*, we have introduced the emergency scenarios field, and we have analyzed some real case scenarios, often linked to the emergency contest, carried out during my time within the Links Foundation research team. In these works, we have shown how Deep Learning can be a valuable tool for analyzing and solving real problems and how feature fusion techniques, often combined with multimodal architectures, can obtain excellent results in identifying flooded areas or in identifying objects and their properties. Encouraged by the good results obtained in *Chapter 4*, in *Chapter 7*, we also applied a similar methodology to an emergency case to get an estimate of the water level, to assess whether it was above or below the knee of the individuals on the scene. For this problem, we have combined two images, one local of the knee area and one global of the whole scene. The union of global and local contexts, through the fusion feature, has allowed us to improve the performance of our classifier, obtaining good results in the prediction phase compared to a model trained only with images of the entire scene. In *Chapter 8*, we extended the same type of architecture by adding a loss function, the compactness loss, which would improve the quality of the features created in the merging layer. The purpose of the function is to provide a better characterization of the features and a better clustering of similar features in the latent space. The compactness loss, in combination with the merging of the features experimentally, proved to be the best model, as the best scoring model, and won the challenge of MediaEval 2018 for the Flood classification task. A final work based on direct fusion but with a different type of application was explored in *Chapter 9*, where we analyzed the best approach to use the features extracted from the images and the best blending mode. The strategies we have implemented are 1) common backbone and common classifier, 2) common backbone but dedicated classifier for each dataset macro category, and 3) dedicated backbone and classifier. All features have been combined through concatenation and direct fusion. In this case,

we have not limited ourselves to the fusion feature alone. Still, we have also focused on analyzing which type of architecture for the multi-tagging task is the best when combining features from macro categories with micro categories. Experimentally, the last approach was the best, dividing the features for each macro category and leaving the multi-tagging to a dedicated model. Finally, in *Chapter 10*, we made a segmentation model analysis, trying to identify the best from an architectural and feature extraction point of view. Many of these models have a standard feature extractor combined with subsequent segmentation layers. These layers use a flow feature fusion methodology to combine information from different resolution scales. We have therefore seen with this analysis the possibilities of extracting and merging features for models with flow fusion approach. This type of blending mode remains one of the best, mainly when applied to the segmentation task where combining information from different levels of resolution scale is necessary.

To conclude, we have represented various machine algorithms in which the development of feature fusion and multimodal learning techniques go hand in hand. The most significant result obtained during this research path is represented by the evolution of a baseline multi-branch architecture to a multi-branch, multi-task, and multimodal learning architecture with a dedicated loss function that ensures good quality compact features. We started from a multi-branch structure, with synthetic inputs extracted from GTAV. We have designed an architecture to resolve two regression tasks, such as estimating the distance and speed of the surrounding vehicles. We obtained good results for both thanks to the combination of semantic and spatial location features for the distance and movement and spatial location features for the speed. Following these results and moving from the automotive context to the emergency one, we have extended the design of the architecture to handle multiple outputs by adding more loss functions, one for each output. Thanks to a careful choice and combination of the loss functions, it was possible to obtain an excellent result by forcing the feature extraction to propose more representative features of higher semantic quality. Worth noting is also the importance that approaches based on Adversarial Functions can have when merging different feature modalities. Our proposed method includes a custom adversarial loss function specifically designed to merge and align the features of real and synthetic trajectories in a common feature space, allowing the model to learn better from both real-world and synthetic data. This results in improved performances for pedestrian trajectory prediction, but it can also be easily generalized to many other tasks where merging different modalities is

necessary. The use of the adversarial loss function allows the model to handle better the variations and discrepancies between the training and unseen data, resulting in a more robust model that can perform well thanks to the augmentation of synthetic data.

Finally, as the last follow-up, in the evolution process of designing a general template for multi-task and multi-branch multimodal learning. We have enriched the previous architecture with a dedicated loss function for the features in the merged layer. We have proposed the *Compacteness Loss*. Thanks to this function, we can ensure that a more compact representation is obtained within the latent space for similar features. As stressed several times in this thesis, extracting good features and merging them correctly to generalize for the task is the key to success in designing a good machine learning model. The definition of a loss function that improves the mapping and the generalization of the features within the latent space and the design of a multimodal feature fusion methodology that can be applied regardless of the number of inputs or tasks represent the most significant contributions of my research work to the multimodal feature fusion task. Future works can start from this template and improve it further, especially by testing other pre-existing functions in the field of Metric Learning (Contrastive Loss, Cosine Loss, etc ...) and applying them to the merging layer. Further improving the quality and representation of the merged features in the latent space is the key to designing good predictive models for multimodal tasks.

References

[1] 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.

[2] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. IEEE, 2009.

[3] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[4] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-CAM: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.

[5] Jiang Deng, Sun Bei, Su Shaojing, and Zuo Zhen. Feature fusion methods in deep-learning generic object detection: A survey. In *2020 IEEE 9th Joint International Information Technology and Artificial Intelligence Conference (ITAIC)*, volume 9, pages 431–437, 2020.

[6] Tsung-Yi Lin, Piotr Dollár, Ross B. Girshick, Kaiming He, Bharath Hariharan, and Serge J. Belongie. Feature pyramid networks for object detection. *CoRR*, abs/1612.03144, 2016.

[7] Pramuditha Perera and Vishal M Patel. Learning deep features for one-class classification. *IEEE Transactions on Image Processing*, 28(11):5450–5463, 2019.

[8] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Y. Bengio. Generative adversarial networks. *Advances in Neural Information Processing Systems*, 3, 06 2014.

[9] Xiaodong Liu, Zhi Gao, and Ben M. Chen. Mlfcgan: Multilevel feature fusion-based conditional gan for underwater image color correction. *IEEE Geoscience and Remote Sensing Letters*, 17(9):1488–1492, 2020.

[10] Haoliang Li, Sinno Jialin Pan, Shiqi Wang, and Alex C. Kot. Domain generalization with adversarial feature learning. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5400–5409, 2018.

[11] Michael Buhrmester, Tracy Kwang, and Samuel D. Gosling. Amazon's mechanical turk: A new source of inexpensive, yet high-quality, data? *Perspectives on Psychological Science*, 6(1):3–5, 2011.

[12] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*. Springer, 2014.

[13] Xiangyu Yue, Bichen Wu, Sanjit A. Seshia, Kurt Keutzer, and Alberto L. Sangiovanni-Vincentelli. A lidar point cloud generator: From a virtual world to autonomous driving. In *Proceedings of the 2018 ACM on International Conference on Multimedia Retrieval*, ICMR '18, pages 458–464, New York, NY, USA, 2018. ACM.

[14] Bichen Wu, Alvin Wan, Xiangyu Yue, and Kurt Keutzer. SqueezeSeg: Convolutional neural nets with recurrent CRF for real-time road-object segmentation from 3d LiDAR point cloud. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, may 2018.

[15] Torcs, http://torcs.sourceforge.net/, 2007.

[16] Matthew Johnson-Roberson, Charles Barto, Rounak Mehta, Sharath Nittur Sridhar, Karl Rosaen, and Ram Vasudevan. Driving in the matrix: Can virtual worlds replace human-generated annotations for real world tasks? In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 746–753. IEEE, 2017.

[17] Andrea Palazzi, Guido Borghi, Davide Abati, Simone Calderara, and Rita Cucchiara. Learning to map vehicles into bird's eye view. In *International Conference on Image Analysis and Processing*. Springer, 2017.

[18] Mark Martinez, Chawin Sitawarin, Kevin Finch, Lennart Meincke, Alex Yablonski, and Alain L. Kornhauser. Beyond grand theft auto V for training, testing and enhancing deep learning in self driving cars. *arXiv*, 2017.

[19] Chenyi Chen, Ari Seff, Alain Kornhauser, and Jianxiong Xiao. Deepdriving: Learning affordance for direct perception in autonomous driving. In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, ICCV '15, pages 2722–2730, Washington, DC, USA, 2015. IEEE Computer Society.

[20] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask r-cnn. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988, 2017.

[21] Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[22] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv*, 2015.

[23] Raniah Zaheer and Humera Shaziya. Gpu-based empirical evaluation of activation functions in convolutional neural networks. In *2018 2nd International Conference on Inventive Systems and Control (ICISC)*, pages 769–773, 2018.

[24] Gunnar Farnebäck. Two-frame motion estimation based on polynomial expansion. In *Proceedings of the 13th Scandinavian Conference on Image Analysis*, SCIA'03, page 363–370, Berlin, Heidelberg, 2003. Springer-Verlag.

[25] Mariusz Bojarski, Philip Yeres, Anna Choromańska, Krzysztof Choromanski, Bernhard Firner, Lawrence D. Jackel, and Urs Muller. Explaining how a deep neural network trained with end-to-end learning steers a car. *ArXiv*, abs/1704.07911, 2017.

[26] Vasileios Karagiannis. *Distance Estimation between Vehicles Based on Fixed Dimensions Licence Plates*. PhD thesis, UNIVERSITY OF PATRAS, 2017.

[27] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi. Social gan: Socially acceptable trajectories with generative adversarial networks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2255–2264, Los Alamitos, CA, USA, jun 2018. IEEE Computer Society.

[28] Stefano Pellegrini, Andreas Ess, and Luc Van Gool. You'll never walk alone: Modeling social behavior for multi-target tracking. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 261–268, 09 2009.

[29] Alon Lerner, Yiorgos Chrysanthou, and Dani Lischinski. Crowds by Example. *Computer Graphics Forum*, 2007.

[30] Matteo Fabbri, Fabio Lanzi, Simone Calderara, Andrea Palazzi, Roberto Vezzani, and Rita Cucchiara. Learning to detect and track visible and occluded body joints in a virtual world. In *European Conference on Computer Vision (ECCV)*, 2018.

[31] Raphael Korbmacher and Antoine Tordeux. Review of pedestrian trajectory prediction methods: Comparing deep learning and knowledge-based approaches. *IEEE Transactions on Intelligent Transportation Systems*, 11 2021.

[32] Dirk Helbing and Péter Molnár. Social force model for pedestrian dynamics. *Physical Review E*, 51(5):4282–4286, May 1995.

[33] Stefano Pellegrini, Andreas Ess, and Luc Van Gool. Improving data association by joint modeling of pedestrian trajectories and groupings. In *ECCV*, 2010.

[34] Hema Swetha Koppula and Ashutosh Saxena. Anticipating human activities using object affordances for reactive robotic response. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38:14–29, 2016.

[35] Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. Social lstm: Human trajectory prediction in crowded spaces. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 961–971, 2016.

[36] Amir Sadeghian, Vineet Kosaraju, Ali Sadeghian, Noriaki Hirose, Hamid Rezatofighi, and Silvio Savarese. Sophie: An attentive gan for predicting paths compliant to social and physical constraints. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[37] Junwei Liang, Lu Jiang, and Alexander Hauptmann. Simaug: Learning robust representations from simulation for trajectory prediction. In *Proceedings of the European Conference on Computer Vision (ECCV)*, August 2020.

[38] Junwei Liang, Lu Jiang, Kevin Murphy, Ting Yu, and Alexander Hauptmann. The garden of forking paths: Towards multi-future trajectory prediction. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[39] Xinyu Zhang, Qiang Wang, Jian Zhang, and Zhao Zhong. Adversarial autoaugment. In *International Conference on Learning Representations*, 2020.

[40] Cyrus Anderson, Xiaoxiao Du, Ram Vasudevan, and Matthew Johnson-Roberson. Stochastic sampling simulation for pedestrian trajectory prediction. *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4236–4243, 2019.

[41] Sahil Narang, Andrew Best, Sean Curtis, and Dinesh Manocha. Generating pedestrian trajectories consistent with the fundamental diagram based on physiological and psychological factors. *PLOS ONE*, 10(4):1–17, 04 2015.

[42] Matteo Fabbri, Guillem Brasó, Gianluca Maugeri, Aljoš an Ošep, Riccardo Gasparini, Orcun Cetintas, Simone Calderara, Laura Leal-Taixé, and Rita Cucchiara. Motsynth: How can synthetic data help pedestrian detection and tracking? In *International Conference on Computer Vision (ICCV)*, 2021.

[43] Rohan Chandra, Uttaran Bhattacharya, Christian Roncal, Aniket Bera, and Dinesh Manocha. Robusttp: End-to-end trajectory prediction for heterogeneous road-agents in dense traffic with noisy sensor inputs. *ACM Computer Science in Cars Symposium*, 2019.

[44] Namhoon Lee, Wongun Choi, Paul Vernaza, Christopher Bongsoo Choy, Philip H. S. Torr, and Manmohan Chandraker. Desire: Distant future prediction in dynamic scenes with interacting agents. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2165–2174, 2017.

[45] EM-DAT. The international disaster database, 2019.

[46] EU-Commission. Funding opportunities to support disaster risk prevention in the cohesion policy 2014-2020 period, 2014.

[47] EM-DAT Data. The international disaster database - data access, 2019.

[48] S. I. Seneviratne, N. Nicholls, D. Easterling, C. M. Goodess, S. Kanae, J. Kossin, Y. Luo, J. Marengo, K. McInnes, M. Rahimi, M. Reichstein, A. Sorteberg, C. Vera, and X. Zhang. *Changes in climate extremes and their impacts on the natural physical environment: An overview of the IPCC SREX report*, page 12566. Provided by the SAO/NASA Astrophysics Data System, April 2012.

[49] OECD. *Financial Management of Flood Risk*. OECD, 2016.

[50] Below R. and Wallemacq P. Annual disaster statistical review 2017, 2018.

[51] A. Restas. Drone applications for supporting disaster management. In *World Journal of Engineering and Technology*, pages 316–321. Scientific research, 2015.

[52] Flavia Sofia Acerbo and Claudio Rossi. Filtering informative tweets during emergencies: A machine learning approach. In *Proceedings of the First CoNEXT Workshop on ICT Tools for Emergency Networks and DisastEr Relief*, I-TENDER '17, pages 1–6, New York, NY, USA, 2017. ACM.

[53] Q. N. Nguyen, A. Frisiello, and C. Rossi. Co-design of a crowdsourcing solution for disaster risk reduction. In *Proceedings of the First CoNEXT Workshop on ICT Tools for Emergency Networks and DisastEr Relief*, I-TENDER '17, pages 7–12, New York, NY, USA, 2017. ACM.

[54] B. Bischke, P. Helber, S. Brugman, E. Basar, Zx Zhao, M. Larson, and K. Pogorelov. The multimedia satellite task at mediaeval 2019: Estimation of flood severity. In *Proc. of the MediaEval 2019 Workshop*, Sophia Antipolis, France, 2019.

[55] L. Lopez-Fuentes, J. van de Weijer, M. González-Hidalgo, H. Skinnemoen, and A. D. Bagdanov. Review on computer vision techniques in emergency situations. *Multimedia Tools and Applications*, 77(13):17069–17107, 2018.

[56] B. Bischke, P. Bhardwaj, A. Gautam, P. Helber, D. Borth, and A. Dengel. Detection of flooding events in social multimedia and satellite imagery using deep neural networks. In *Proc. of the MediaEval 2017 Workshop*, Dublin, Ireland, 2017.

[57] K. Avgerinakis, A. Moumtzidou, S. Andreadis, E. Michail, I. Gialampoukidis, S. Vrochidis, and I. Kompatsiaris. Visual and textual analysis of social media and satellite images for flood detection@ multimedia satellite task mediaeval 2017. In *Proc. of the MediaEval 2017 Workshop*, Dublin, Ireland, 2017.

[58] B. Bischke, P. Helber, Z. Zhao, J. De Bruijn, and D. Borth. The multimedia satellite task at mediaeval 2018. In *Proc. of the MediaEval 2018 Workshop*, Sophia Antipolis, France, 2018.

[59] S. Qiu, Z. Zhu, and B. He. Fmask 4.0: Improved cloud and cloud shadow detection in landsats 4–8 and sentinel-2 imagery. *Remote Sensing of Environment*, 231:111205, 2019.

[60] Z. Zhu and C. E. Woodcock. Object-based cloud and cloud shadow detection in landsat imagery. *Remote sensing of environment*, 118:83–94, 2012.

[61] A. Farasin and P. Garza. Perceive: Precipitation data characterization by means on frequent spatio-temporal sequences. In *ISCRAM*, 2018.

[62] K. Osumi. Detecting land cover change using sentinel-2. *Abstracts of the ICA*, 1, 2019.

[63] C. Rossi, F. S. Acerbo, K. Ylinen, I. Juga, P. Nurmi, A. Bosca, F. Tarasconi, M. Cristoforetti, and A. Alikadic. Early detection and information extraction for weather-induced floods using social media streams. *International journal of disaster risk reduction*, 30:145–157, 2018.

[64] L. Lopez-Fuentes, J. van de Weijer, M. Bolanos, and H. Skinnemoen. Multi-modal deep learning approach for flood detection. In *Proc. of the MediaEval 2017 Workshop*, Dublin, Ireland, 2017.

[65] B. Bischke, P. Helber, Z. Zhao, J. de Bruijn, and D. Borth. The multimedia satellite task at mediaeval 2018: Emergency response for flooding events. In *Proc. of the MediaEval 2018 Workshop*, Sophia-Antipolis, France, 2018.

[66] L. Lopez-Fuentes, A. Farasin, H. Skinnemoen, and P. Garza. Deep learning models for passability detection of flooded roads. In *Proc. of the MediaEval 2018 Workshop*, Sophia Antipolis, France, 2018.

[67] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[68] A. G Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *ArXiv*, 2017.

[69] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *ICLR*, 2015.

[70] Z. Cao, T. Simon, S. Wei, and Y. Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7291–7299, 2017.

[71] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[72] G. Donchyts, J. Schellekens, H. Winsemius, E. Eisemann, and N. van de Giesen. A 30 m resolution surface water mask including estimation of positional and thematic differences using landsat 8, srtm and openstreetmap: a case study in the murray-darling basin, australia. *Remote Sensing*, 8(5):386, 2016.

[73] AnsuR Technologies AS. UN-ASIGN. App: https://play.google.com/store/apps/details?id=ansur.asign.un&hl=en_US, FP7 Project: https://cordis.europa.eu/project/rcn/94375/factsheet/en, 2019.

[74] Istituto Superiore Mario Boella (ISMB). I-REACT. App: https://play.google.com/store/apps/details?id=it.ismb.iReact&hl=en_US, H2020 Project: https://cordis.europa.eu/project/rcn/203294/factsheet/en, 2019.

[75] Clayton Wukich et al. Social media use in emergency management. *Journal of Emergency Management*, 13(4):281–294, 2015.

[76] Anita Saroj and Sukomal Pal. Use of social media in crisis management: A survey. *International Journal of Disaster Risk Reduction*, page 101584, 2020.

[77] Nayomi Kankanamge, Tan Yigitcanlar, Ashantha Goonetilleke, and Md Kamruzzaman. Determining disaster severity through social media analysis: Testing the methodology with south east queensland flood tweets. *International journal of disaster risk reduction*, 42:101360, 2020.

[78] Nayomi Kankanamge, Tan Yigitcanlar, and Ashantha Goonetilleke. How engaging are disaster management related social media channels? the case of australian state emergency organisations. *International Journal of Disaster Risk Reduction*, page 101571, 2020.

[79] Cornelia Ferner, Clemens Havas, Elisabeth Birnbacher, Stefan Wegenkittl, and Bernd Resch. Automated seeded latent dirichlet allocation for social media based event detection and mapping. *Information*, 11(8):376, 2020.

[80] Anna Kruspe, Jens Kersten, and Friederike Klan. Detection of informative tweets in crisis events. *Natural Hazards and Earth System Sciences Discussions*, pages 1–18, 2020.

[81] Mirko Zaffaroni and Claudio Rossi. Water segmentation with deep learning models for flood detection and monitoring. In *Proceedings of the 17th ISCRAM Conferen*, pages 66–74, 2020.

[82] Laura Lopez-Fuentes, Claudio Rossi, and Harald Skinnemoen. River segmentation for flood monitoring. In *2017 IEEE International Conference on Big Data (Big Data)*, pages 3746–3749. IEEE, 2017.

[83] Dan Bînă, George-Alexandru Vlad, Cristian Onose, and Dumitru-Clementin Cercel. Flood severity estimation in news articles using deep learning approaches. *MediaEval 2019*, 2019.

[84] Mirko Zaffaroni, Laura Lopez-Fuentes, Alessandro Farasin, Paolo Garza, and Harald Skinnemoen. Ai-based flood event understanding and quantification using online media and satellite data. *MediaEval 2019*, 2019.

[85] Yu Feng, Sergiy Shebotnov, Claus Brenner, and Monika Sester. Ensembled convolutional neural network models for retrieving flood relevant tweets. In *Proc. of the MediaEval 2018 Workshop (Oct. 29-31, 2018).*, Sophia-Antipolis, France, 2018.

[86] Muhammad Hanif, Muhammad Atif Tahir, and Muhammad Rafi. Detection of passable roads using ensemble of global and local features. In *Proc. of the MediaEval 2018 Workshop (Oct. 29-31, 2018).*, Sophia-Antipolis, France, 2018.

[87] Anastasia Moumtzidou, Panagiotis Giannakeris, Stelios Andreadis, Athanasios Mavropoulos, Georgios Meditskos, Ilias Gialampoukidis, Konstantinos Avgerinakis, Stefanos Vrochidis, and Ioannis Kompatsiaris. A multimodal approach in estimating road passability through a flooded area using social media and satellite images. In *Proc. of the MediaEval 2018 Workshop (Oct. 29-31, 2018).*, Sophia-Antipolis, France, 2018.

[88] Armin Kirchknopf, Djordje Slijepcevic, Matthias Zeppelzauer, and Markus Seidl. Detection of road passability from social media and satellite images. In *Proc. of the MediaEval 2018 Workshop (Oct. 29-31, 2018).*, Sophia-Antipolis, France, 2018.

[89] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.

[90] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *Proceedings of the International Conference on Learning Representations (ICLR 2013)*, 2013. Workshop poster. Available at https://arxiv.org/pdf/1301.3781.pdf.

[91] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[92] Max Bramer. *Principles of data mining*, volume 180. Springer, 2007.

[93] Yoon Kim. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar, October 2014. Association for Computational Linguistics.

[94] Savvas A Chatzichristofis and Yiannis S Boutalis. CEDD: color and edge directivity descriptor: a compact descriptor for image indexing and retrieval. In *International Conference on Computer Vision Systems*, pages 312–322. Springer, 2008.

[95] Hamid A Jalab. Image retrieval system based on color layout descriptor and Gabor filters. In *2011 IEEE Conference on Open Systems*, pages 32–36. IEEE, 2011.

[96] Savvas A Chatzichristofis and Yiannis S Boutalis. FCTH: Fuzzy color and texture histogram-a low level feature for accurate image retrieval. In *2008 Ninth International Workshop on Image Analysis for Multimedia Interactive Services*, pages 191–196. IEEE, 2008.

[97] Dong Kwon Park, Yoon Seok Jeon, and Chee Sun Won. Efficient use of local edge histogram descriptor. In *Proceedings of the 2000 ACM workshops on Multimedia*, pages 51–54. ACM, 2000.

[98] Konstantinos Zagoris, Savvas A Chatzichristofis, Nikos Papamarkos, and Yiannis S Boutalis. Automatic image annotation and retrieval using the joint composite descriptor. In *2010 14th Panhellenic Conference on Informatics*, pages 143–147. IEEE, 2010.

[99] Bangalore S Manjunath, J-R Ohm, Vinod V Vasudevan, and Akio Yamada. Color and texture descriptors. *IEEE Transactions on circuits and systems for video technology*, 11(6):703–715, 2001.

[100] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[101] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[102] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[103] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations, 2015*, 2015. Available at https://www.robots.ox.ac.uk/~vgg/publications/2015/Simonyan15/simonyan15.pdf.

[104] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement, 2018.

[105] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *IEEE transactions on pattern analysis and machine intelligence*, 40(6):1452–1464, 2018.

[106] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[107] Benjamin Bischke, Patrick Helber, and Andreas Dengel. Global-local feature fusion for image classification of flood affected roads from social multimedia. In *Proc. of the MediaEval 2018 Workshop (Oct. 29-31, 2018).*, Sophia-Antipolis, France, 2018.

[108] Zhengyu Zhao, Martha Larson, and Nelleke Oostdijk. Exploiting local semantic concepts for flooding-related social image classification. In *Proc. of the MediaEval 2018 Workshop (Oct. 29-31, 2018).*, Sophia-Antipolis, France, 2018.

[109] Danielle Dias and Ulisses Dias. Flood detection from social multimedia and satellite images using ensemble and transfer learning with cnn architectures. In *Proc. of the MediaEval 2018 Workshop (Oct. 29-31, 2018).*, Sophia-Antipolis, France, 2018.

[110] Naina Said, Konstantin Pogorelov, Kashif Ahmad, Michael Riegler, Nasir Ahmad, Olga Ostroukhova, Pål Halvorsen, and Nicola Conci. Deep learning approaches for flood classification and flood aftermath detection. In *Proc. of the MediaEval 2018 Workshop (Oct. 29-31, 2018).*, Sophia-Antipolis, France, 2018.

[111] Mediaeval 2018 multimedia satellite task. http://www.multimediaeval.org/mediaeval2018/multimediasatellite/, 2018. Data released: 31 May 2018.

[112] Peilu Wang, Yao Qian, Frank K Soong, Lei He, and Hai Zhao. Learning distributed word representations for Bidirectional LSTM recurrent neural network. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 527–533, 2016.

[113] Thomas G Dietterich. *Ensemble Methods in Machine Learning*. Springer, 2000.

[114] PM Lerman. Fitting segmented regression models by grid search. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 29(1):77–84, 1980.

[115] Xuebo Liu, Ding Liang, Shi Yan, Dagui Chen, Yu Qiao, and Junjie Yan. Fots: Fast oriented text spotting with a unified network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5676–5685, 2018.

[116] George Awad, Asad A. Butt, Keith Curtis, Yooyoung Lee, Jonathan Fiscus, Afzal Godil, Andrew Delgado, Jesse Zhang, Eliot Godard, Lukas Diduch, Jeffrey Liu, Alan F. Smeaton, Yvette Graham, Gareth J. F. Jones, Wessel Kraaij, and Georges Quénot. Trecvid 2020: comprehensive campaign for evaluating video retrieval tasks across multiple application domains. In *Proceedings of TRECVID 2020*. NIST, USA, 2020.

[117] Kevin Crowston. Amazon mechanical turk: A research tool for organizations and information systems scholars. In Anol Bhattacherjee and Brian Fitzgerald, editors, *Shaping the Future of ICT Research. Methods and Approaches*, pages 210–221, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[118] Alon Halevy, Peter Norvig, and Fernando Pereira. The unreasonable effectiveness of data. *IEEE Intelligent Systems*, 24:8–12, 2009.

[119] Muhammad Imran, Carlos Castillo, Fernando Diaz, and Sarah Vieweg. Processing social media messages in mass emergency: A survey. *ACM Comput. Surv.*, 47(4):67:1–67:38, June 2015.

[120] Muhammad Imran, Carlos Castillo, Ji Lucas, Patrick Meier, and Sarah Vieweg. Aidr: Artificial intelligence for disaster response. In *Proceedings of the 23rd International Conference on World Wide Web*, WWW '14 Companion, pages 159–162, New York, NY, USA, 2014. ACM.

[121] C. Rossi, F.S. Acerbo, K. Ylinen, I. Juga, P. Nurmi, A. Bosca, F. Tarasconi, M. Cristoforetti, and A. Alikadic. Early detection and information extraction for weather-induced floods using social media streams. *International Journal of Disaster Risk Reduction*, 30:145 – 157, 2018. Communicating High Impact Weather: Improving warnings and decision making processes.

[122] C. L. Lai, J. C. Yang, and y.h. Chen. A real time video processing based surveillance system for early fire and flood detection. *2007 IEEE Instrumentation and Measurement Technology Conference IMTC 2007*, pages 1–6, 2007.

[123] Alexander Filonenko, Wahyono, Danilo Cáceres Hernández, Dongwook Seo, and Kang-Hyun Jo. Real-time flood detection for video surveillance. In *IECON 2015 - 41st Annual Conference of the IEEE Industrial Electronics Society*, pages 004082–004085, 2015.

[124] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015.

[125] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *CoRR*, abs/1511.00561, 2015.

[126] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. *CoRR*, abs/1612.01105, 2016.

[127] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *CoRR*, abs/1606.00915, 2016.

[128] Huikai Wu, Junge Zhang, Kaiqi Huang, Kongming Liang, and Yizhou Yu. Fastfcn: Rethinking dilated convolution in the backbone for semantic segmentation. *CoRR*, abs/1903.11816, 2019.

[129] Towaki Takikawa, David Acuna, Varun Jampani, and Sanja Fidler. Gated-scnn: Gated shape cnns for semantic segmentation. *CoRR*, abs/1907.05740, 2019.

[130] L. Lopez-Fuentes, C. Rossi, and H. Skinnemoen. River segmentation for flood monitoring. In *2017 IEEE International Conference on Big Data (Big Data)*, pages 3746–3749, December 2017.

[131] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014.

[132] Christian Mostege, Michael Maurer, Nikolaus Heran, Jesus Pestana Puerta, and Friedrich Fraundorfer. Semantic drone dataset, 2019.

[133] A. Criminisi. Msrc-v2 image database, 2005.

[134] A. Y. C. Chen and Corso J. J. Propagating multi-class pixel labels throughout video frames. In *Proceedings of Western New York Image Processing Workshop*, 2010.

[135] Laura Lopez-Fuentes and Claudio Rossi. River segmentation dataset, October 2017.

[136] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv 1409.1556*, 09 2014.

[137] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[138] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *CoRR*, abs/1704.04861, 2017.

[139] Simon Kornblith, Jonathon Shlens, and Quoc V. Le. Do better imagenet models transfer better? *CoRR*, abs/1805.08974, 2018.

[140] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Fei-Fei Li. Imagenet large scale visual recognition challenge. *CoRR*, abs/1409.0575, 2014.

[141] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results, 2012.

[142] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3431–3440, June 2015.

[143] Pierre Baldi. Autoencoders, unsupervised learning, and deep architectures. In Isabelle Guyon, Gideon Dror, Vincent Lemaire, Graham Taylor, and Daniel Silver, editors, *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*, volume 27 of *Proceedings of Machine Learning Research*, pages 37–49, Bellevue, Washington, USA, July 2012. PMLR.

[144] Seyed Sadegh Mohseni Salehi, Deniz Erdogmus, and Ali Gholipour. Tversky loss function for image segmentation using 3d fully convolutional deep networks. *CoRR*, abs/1706.05721, 2017.

[145] Alan G. Glaros and Rex Bryan Kline. Understanding the accuracy of tests with cutting scores: the sensitivity, specificity, and predictive value model. *Journal of clinical psychology*, 44 6:1013–23, 1988.

[146] Matthew D. Zeiler. ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701, 2012.

[147] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, June 2010.

[148] Martin Thoma. A survey of semantic segmentation. *CoRR*, abs/1602.06541, 2016.

Appendix A

Compactness Loss equation derivation

In Appendix A from [7], it is stated that the gradient results from the following equation

$$\frac{\partial l_C}{\partial x_{ij}} = \frac{2}{(n-1)nk}\left[n \times \left(x_{ij} - m_{ij}\right) - \sum_{l=1}^{n}\left(x_{il} - m_{il}\right)\right]. \tag{A.1}$$

However, there are some mistakes in that equation. The first mistake is within the summation since the samples $\mathbf{x_i}$ have k components, not n. However, as we will prove, this is not a unique mistake.

Let us compute the gradient of l_C with respect to x_{ij}. Using the definition of the inner product, we have that $\mathbf{z_i}^T\mathbf{z_i} = \sum_{t=1}^{k} z_{it}^2$. Thus, l_C can be written as

$$l_C = \frac{1}{nk}\sum_{l=1}^{n}\sum_{t=1}^{k}\left(x_{lt} - m_{lt}\right)^2.$$

Now, taking partial derivatives of l_C with respect to x_{ij} for all $1 \leq i \leq n$ and $1 \leq j \leq k$, we obtain

$$\frac{\partial l_C}{\partial x_{ij}} = \frac{2}{nk}\sum_{l=1}^{n}\left(x_{lj} - m_{lj}\right) \cdot \left(\frac{\partial\left(x_{lj} - m_{lj}\right)}{\partial x_{ij}}\right).$$

This first step is already incorrect in [7]. The rest of the proof follows similarly. Let us check it. Note that

$$\frac{\partial(x_{lj}-m_{lj})}{\partial x_{ij}} = \begin{cases} 1 & \text{if } l=i, \\ -\dfrac{1}{n-1} & \text{otherwise.} \end{cases}$$

Thus, we obtain that

$$\begin{aligned} \frac{\partial l_C}{\partial x_{ij}} &= \frac{2}{nk}\left[x_{ij}-m_{ij}-\frac{1}{n-1}\sum_{\substack{l=1\\ l\neq i}}^{n}\left(x_{lj}-m_{lj}\right)\right] \\ &= \frac{2}{nk}\left[\frac{n}{n-1}\cdot(x_{ij}-m_{ij})-\frac{1}{n-1}\sum_{l=1}^{n}\left(x_{lj}-m_{lj}\right)\right] \\ &= \frac{2}{(n-1)nk}\left[n\cdot(x_{ij}-m_{ij})-\sum_{l=1}^{n}\left(x_{lj}-m_{lj}\right)\right], \end{aligned}$$

retrieving finally

$$\frac{\partial l_C}{\partial x_{ij}} = \frac{2}{(n-1)nk}\left[n\cdot\left(x_{ij}-m_{ij}\right)-\sum_{l=1}^{n}\left(x_{lj}-m_{lj}\right)\right].$$