TÉCNICO LISBOA

Classifying Geo-Referenced Photos and Segmenting Satellite Imagery for the Assessment of Flood Severity

Jorge Miguel da Silva Pereira

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisors: Prof. Bruno Emanuel da Graça Martins
Prof. Jacinto Paulo Simões Estima

Examination Committee

Chairperson: Prof. Alexandre Paulo Lourenço Francisco
Supervisor: Prof. Bruno Emanuel da Graça Martins
Member of the Committee: Prof. João Miguel da Costa Magalhães

November 2019

Acknowledgements

First, I would like to thank Professor Bruno Martins and Professor Jacinto Estima for their guidance during this last year, through which they contributed very significantly to the work that is reported on this dissertation, with their knowledge and motivation.

Second, I would also like to thank my family, for their constant support and for allowing me to learn in such a distinguished institute as Instituto Superior Técnico.

I would also like to thank the organizers of the Multimedia Satellite Task of the MediaEval competition, for being always available to answer some questions that arose during the development of this work, and for providing the datasets used in the competition. I would also like to thank all the developers responsible for Keras deep learning library, which provides a simplistic, although very powerful, interface for building neural networks. Without the Keras API and the datasets from the Multimedia Satellite Task of the MediaEval competition, it would be extremely difficult to perform all the experiments reported in this work.

It is also important to acknowledge the Fundação para a Ciência e Tecnologia (FCT) for supporting the work reported in this dissertation, through project grants with the references PTDC/EEI-SCR/1743/2014 (Saturn), PTDC/CTA-OHR/29360/2017 (RiverCure), and PTDC/CCI-CIF/32607/2017 (MIMU). I also gratefully acknowledge the support of NVIDIA Corporation, with the donation of the two Titan Xp GPUs used in the reported experiments.

Last, but not least, I also have to thank all my friends and colleagues for all the support given through all the extremely laborious, but fantastic, path in the Instituto Superior Técnico.

Jorge Miguel da Silva Pereira

For all my family and friends,

Resumo

A utilização de *social media* no contexto de desastres naturais tem aumentado rapidamente. Particularmente no contexto de inundações, temos que imagens georreferenciadas partilhadas pelos cidadãos podem fornecer conhecimento sobre a situação aos serviços de emergência, bem como prestar assistência na avaliação de danos materiais, fornecendo informações que, de outra forma, seriam difíceis de recolher com sensores convencionais e/ou deteção remota. Além disso, avanços recentes nas áreas da visão computacional e aprendizagem profunda podem apoiar a análise destes dados. Na minha dissertação, primeiramente focando-me em imagens ao nível do solo recolhidas durante inundações, avaliei o uso de redes neurais convolucionais nas tarefas de (i) selecionar imagens com evidências diretas de uma inundação, e (ii) estimar a gravidade da inundação. As experiências reportadas alcançaram resultados de grande qualidade, demonstrando potencial no complemento de outras fontes de informação (e.g., de deteção remota).

Imagens de satélite podem, por outro lado, ser usadas como alternativa (ou complemento) em tarefas como o mapeamento da extensão de inundações, ou seja, na classificação de pixeis referentes a áreas inundadas em imagens de satélite registadas durante, ou logo após, inundações. Baseando-me em trabalhos recentes, focados em segmentação de imagens, também avaliei o desempenho de uma versão adaptada da rede U-Net na segmentação de áreas inundadas, tendo como entrada múltiplas variáveis de detecção remota. Através de um extenso conjunto de testes, mostro que a arquitetura neuronal considerada é bastante eficaz, produzindo mapas que visualmente contêm as delimitações das inundações ilustradas com grande qualidade, e superando várias abordagens mais simples em termos de métricas padrão.

Abstract

The use of social media in disaster and crisis management is rapidly increasing. Particularly in connection to flooding events, geo-referenced images shared by citizens can provide situational awareness to emergency responders, as well as assistance to financial loss assessment, giving information that would otherwise be hard to collect through conventional sensors and/or remote sensing. Moreover, recent advances in computer vision and deep learning can perhaps support the analysis of these data. In my thesis, firstly focusing on ground-level images taken during flooding events, I evaluated the use of convolutional neural networks for (i) discriminating images showing direct evidence of a flood, and (ii) estimating flood severity. Experiments showed a high accuracy on this task, demonstrating the potential to complement other sources of information (e.g., satellite imagery) related to flooding events.

Overhead imagery can, in turn, be used as an alternative (or complement) for tasks such as flood extent mapping, which concerns with discriminating pixels referring to flooded areas in satellite images recorded during, or shortly after, flooding events. Following recent work, this time focusing on image segmentation, I also evaluated the use of an adapted version of the U-Net neural network architecture, on the specific problem of segmenting flooded areas and taking multiple remote sensing data sources as input. Through an extensive set of evaluation experiments, I show that the proposed neural model is quite effective, producing maps that visually have a high-quality delimitation of the flooded areas, and outperforming simpler/ablated baselines in terms of various metrics.

Palavras Chave
Keywords

Palavras Chave

Classificação de fotos georreferenciadas

Segmentação de imagens de satélite

Identificação e delimitação de cheias

Estimação da severidade de cheias

Redes neuronais convolucionais

Keywords

Classification of geo-referenced photos

Segmentation of satellite imagery

Identifying and delimiting floods

Flood severity estimation

Convolutional neural networks

Contents

List of Figures

List of Tables

1 Introduction

Natural disasters like floods raise important challenges to civil protection authorities, at the same time also involving high costs for repair after the disaster. These costs can perhaps be reduced if an early detection is made. Currently, this detection is majorly sustained by remote sensing (e.g., satellite imagery), although the increase in the number of geo-referenced publications on social networks, namely images, may allow the use of ground-level information to support other sensors. In the context of my Master of Science (M.Sc.) thesis, I propose a set of intelligent systems that, leveraging state-of-the-art deep learning techniques based on convolutional neural networks for image classification (Altenberger and Lenz, 2018; Khan et al., 2018; Xie et al., 2018) and segmentation (Ghosh et al., 2019; Chen et al., 2018b), are able to (i) given a geo-referenced photograph, find out if a flood is being depicted, and estimate the severity of the flood (i.e., in images illustrating a flood, produce an estimate for the water level), and (ii) given a satellite image, create a binary mask that represents the precise extent of the flooded area. This work presents all the details of the proposed systems, as well as an extensive set of evaluation experiments that compare the obtained results with those from other similar studies, as well as against simpler baseline approaches.

1.1 Motivation

The widespread use of mobile consumer electronics has made the act of taking and sharing photos online, for instance using smartphones or digital cameras coupled with GPS receivers, became commonplace. Crowdsourcing through publicly shared photos has created a new opportunity to collect vast amounts of geo-referenced image data, which can be useful in disaster and crisis management. While crowdsourcing images posted on social media has been investigated in various contexts, apart from a few recent initiatives (Bischke et al., 2017b; See, 2019), we have that leveraging such data in connection to flooding events (e.g., using these data to detect the presence, or estimate the severity, of flooding events), remains relatively unexplored.

This new data collection methodology can have the advantage of providing a local perspective on inundations, that is very hard, or even impossible, to achieve with conventional sensors (e.g., it can perhaps be used estimate the boundaries of flood-water, including partial blockage of roadways due to flooding (Narayanan et al., 2014)).

Notwithstanding, the perspective that an overhead image provides is also highly valuable, and since the ease in accessing near-real-time high-resolution satellite imagery has been increasing every day, an automatic semantic segmentation model for delimiting flooded areas (i.e., for determining, for each pixel, the most likely class label from a finite set of possibilities, this way discriminating pixels referring to flooded areas), would also be very useful. An automatically and precise delimitation of the flood extent, can be very helpful to emergency responders, and it can also inform several downstream applications.

In the 2017 edition of the Multimedia Satellite Task from MediaEval (Bischke et al., 2017b), the organizers proposed two sub-tasks named Disaster Image Retrieval from Social Media (DIRSM), and Flood Detection in Satellite Images (FDSI). In these sub-tasks, some of the aforementioned ideas were addressed, specifically the identification of flooding events in ground-level imagery, and the semantic segmentation of satellite images. However, the majority of the participants used methods based on ensembles of random forests (Fu et al., 2017), logistic regression (N. Tkachenko and Procter, 2017), or support vector machines (Avgerinakis et al., 2017), not taking advantage of recent deep learning developments.

1.2 Thesis Proposal

In the context of my M.Sc. research project, I propose to implement and evaluate a system that, leveraging recent deep learning techniques, is able to classify ground-level images, collected from social networks, regarding the presence (and the severity), of the depicted flooding events. I argue that recent deep learning models, such as the DenseNet (Huang et al., 2017) or the EfficientNet (Tan and Le, 2019) neural architectures, together with an attention-guided methodology (Guan et al., 2018), are able to outperform simpler techniques (e.g., support vector machines or random forests). I also argue that similar techniques, based on fully-convolutional neural networks for semantic segmentation (e.g., extensions on the U-Net (Ronneberger et al., 2015) neural architecture) are capable of producing high-quality binary masks from high-resolution satellite imagery, corresponding to the delimitation of the flooding events. The techniques discussed in this dissertation achieved, in fact, very good results (i.e., corresponding to state-of-the-art results in some cases), attesting to the adequacy of the proposed neural models.

1.3 Contributions

In brief, the main contributions of this thesis are as follows:

- Creation of a dataset with 10,734 images, aggregating and extending pre-existing datasets, annotated regarding three flooding severity classes: (i) without evidence of a flooding event, (ii) evidence of a flooding event with less than one meter of depth, and (iii) evidence of a flooding event with more than one meter of depth. This work also presents a heuristic-based method for refining these classifications in the case of geo-referenced photos, in order to obtain an accurate estimate, in meters, for the water depth;

- The proposal of a set of intelligent methods, based on the work from Huang et al. (2017), Tan and Le (2019), and Guan et al. (2018), for classifying ground-level photographs in three different ways: (i) discriminate images with direct evidence for a flood, (ii) classify images into the three aforementioned flood-related classes, and (iii) output the water depth, in meters, from images depicting flooding events. Experimental results show that the proposed methods significantly outperform previous proposals, for instance, reported in the context of the DIRSM sub-task at MediaEval 2017;

- Development of a proof-of-concept web application that shows, in a global map, all the geo-referenced images used in the reported experiments. Furthermore, the users also have the possibility to upload new images, observe the classification made by the best performing model in terms of the predicted severity class and water depth, and register eventual corrections to the results of the model;

- Development of a fully-convolutional neural network architecture that is able to receive, as input, multiple information from overhead sensors (e.g., satellite images with RGB and near-infrared information, vegetation or water indexes, etc.), and output a mask that corresponds to the area affected by the depicted flooding event. Experimental results show that the proposed neural model achieves results that are on par with those from ensemble methods reported in the context of the FDSI sub-task at MediaEval 2017.

The work reported in this dissertation was also used as basis for three scientific articles that are currently under review:

- *Assessing Flood Severity from Georeferenced Photos* (Pereira et al., 2019b), submitted to the 13th workshop on Geographic Information Retrieval. This paper describes the methodology used for extending existing flood-related datasets into the three aforementioned flood severity classes. It also details the training strategies considered for the DenseNet (Huang et al., 2017) and EfficientNet (Tan and Le, 2019) neural architectures to address the tasks of (i) discriminating between images with flood evidence, and (ii) classifying those images into the three proposed flood-related classes, presenting also the obtained results;

- *Assessing Flood Severity from Crowdsourced Social Media Photos with Deep Neural Networks* (Pereira et al., 2019c), submitted to the Multimedia Tools and Applications journal. This article extends the previous workshop submission by presenting a new heuristic method to obtain an estimate, in meters, for the flood-water depth. It also describes an attention guided methodology (Guan et al., 2018) used to extend the DenseNet model, together with the changes related to predicting the value, in meters, for the water depth;

- *A Dense U-Net Model Leveraging Multiple Remote Sensing Data Sources for Flood Extent Mapping* (Pereira et al., 2019a), submitted to the International Journal of Geographic Information Science. This article details an adapted U-Net (Ronneberger et al., 2015) model for segmenting aerial imagery, including (i) an extensions corresponding to dense connections, (ii) the replacement of the typical convolutional blocks with dense blocks, and (iii) the addition of squeeze and excitation operations. Together with the complete model description, the article presents the obtained results in the task of segmenting flooded regions taking as input different types of features.

1.4 Structure of the Document

The rest of this document is organized as follows. Chapter 2 presents fundamental concepts related to deep learning and neural networks, detailing convolutional neural networks, and it also describes important related work in the areas of image classification, image segmentation, and in the usage of volunteered geographic information to identify natural phenomena. Chapters 3 and 4 describe, respectively, the proposed approaches for (i) classifying ground-level images regarding the evidence/severity of a flooding event, and (ii) delimiting flooding events in satellite imagery. Each of these two chapters also describes the considered evaluation methodologies, and the obtained results. Finally, Chapter 5 concludes this document, summarizing the main findings and highlighting possible directions for future research in the area.

2 Fundamental Concepts and Related Work

This chapter explains fundamental concepts that will help the reader understand the following sections and the proposed work. Section 2.1.1 first introduces concepts related to machine learning with neural networks in general, then Section 2.1.2 details convolutional neural network architectures and their application in image classification tasks, and Section 2.1.3 presents an introduction to fully-convolutional neural networks typically used for image segmentation tasks. Section 2.2 presents important related work related to image classification, image segmentation, and the use of volunteered geographic information for mapping natural phenomena. Finally, an overview of the related work is presented in Section 2.2.4.

2.1 Fundamental Concepts

The majority of the state-of-the-art computer vision systems existing nowadays rely on Convolutional Neural Networks (CNNs). CNNs are a specification over the simpler feed-forward neural networks since they use convolutions as the basis operation. A convolution can be seen as a filter that tries to extract relevant information from the input feature map (e.g., a filter that extracts circular shapes from an input image). Training a CNN consists of fine-tuning these filters to efficiently solve the task in hand (in this case, identify and classify flooding events). Section 2.1.1 starts by giving the reader a glimpse over neural networks and deep learning in general, Section 2.1.2 details convolutional neural networks, and Section 2.1.3 presents a particular type of CNNs referred to as fully-convolutional neural networks.

2.1.1 Introduction to Neural Networks and Deep Learning

A neural network, as the name implies, is a set of interconnected data processing units (i.e., neurons) that transmit information between them. They were initially based on a single neuron, that is often referred to as a perceptron (Rosenblatt, 1958). A perceptron can be seen as a function, receiving some values as input (the equivalent of the electric impulses in a biological neuron) and producing another value as output (the communication of an impulse to another

neuron), resulting from a linear combination of the inputs that is afterwards processed through a non-linear function, usually referred to as an activation function. More formally, representing the inputs as $\boldsymbol{i} = (i_1, i_2, \ldots, i_n)$ and considering a vector of weights $\boldsymbol{w} = (w_1, w_2, \ldots, w_n)$ together with a bias term b, the output $\hat{y}$ of a standard perceptron for binary classification corresponds to the following equation:

$$\hat{y} = \text{sign}(\boldsymbol{w} \cdot \boldsymbol{i} + b) \tag{2.1}$$

Although a single perceptron is already an interesting classification model, better results can often be obtained if several are used. With the combination of several perceptrons, we can have a simple model of a neural network that is commonly referred to as a Multi-Layer Perceptron (MLP). A MLP follows a layered structure, in which the first layer is referred to as the *input layer* and the last as the *output layer*. Intermediate layers are called *hidden layers*. A MLP with more than one hidden layer is referred to as a deep neural network. These mathematical objects are the core of what is called nowadays deep learning.

Training a neural network corresponds to finding all weights and biases that better map a certain set of points to a certain set of outputs (e.g., classes). This is a problem that can be solved using the back-propagation algorithm with a loss function that returns an error associated with the output made by the network. Back-propagation is a two-step algorithm: in the first step, the input layer is fed with input values for which we know the true output that should be produced. These values are then propagated to the next layers until reaching the last layer, where the network produces an output. When the last layer is reached the value of the chosen loss function is calculated (i.e., the difference between the real and the predicted values). In the next step, this error is propagated backwards through the network, so that the parameters can be updated in the direction of the negative gradient of the loss function (i.e., the gradient points in the direction of the steepest increase in the loss). This allows the network to update the weights and biases accordingly, using the chain rule of differentiation to compute the gradient and some variation of the general gradient descent algorithm. In general, a gradient descent algorithm is an optimization technique that finds the minimum of a function. There are some variations of this algorithm frequently used for training deep neural networks (Ruder, 2016).

One variation that is commonly used is the Adaptive Moment Estimation (Adam) method originally proposed by Kingma and Ba (2014). Its use comes from the fact that it can achieve good results fast. In practice, this optimizer performs larger updates for infrequent parameters, and smaller updates for frequent parameters.

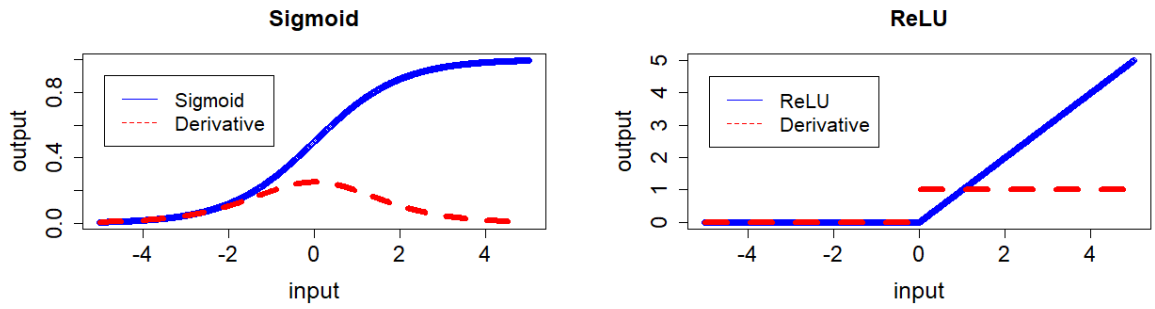


Figure 2.1: Comparison between the shape of the logistic sigmoid function and the ReLU function. The behavior of the derivatives of each function is also presented.

In the intermediate nodes of deep neural networks it is common to use non-linear activation functions such as the logistic sigmoid function or the Rectified Linear Unit (ReLU). Each one of these functions has advantages and disadvantages. For instance, ReLU is computationally more efficient, while the logistic sigmoid is limited to values between zero and one. While the logistic sigmoid tends to suffer from vanishing gradients, ReLU has a problem referred to in the literature as the dying ReLU problem, that happens when too many inputs get below zero, causing most of the units in the network to simply output zero (Agarap, 2018). The behavior of these two activation functions, and their derivatives, is shown in Figure 2.1, while the corresponding equations are presented bellow.

$$\text{sigmoid}(x) = \frac{1}{1 + \exp(-x)} \tag{2.2}$$

$$\text{ReLU}(x) = \max(0, x) \tag{2.3}$$

2.1.2 Convolutional Neural Networks

There are some problems for which a typical neural network would require an input layer with a large width, for example, image processing tasks. In this class of problems a neuron would be required for each value of a pixel color element. A Convolutional Neural Network (CNN) addresses this problem using convolutions (i.e., a mathematical operations that correspond to the integral of the product of two functions, with one of them mirrored). Imagine that you have two vectors, specifically $\boldsymbol{x}$ (called *input*) and $\boldsymbol{w}$ (*kernel*) with the values represented bellow:

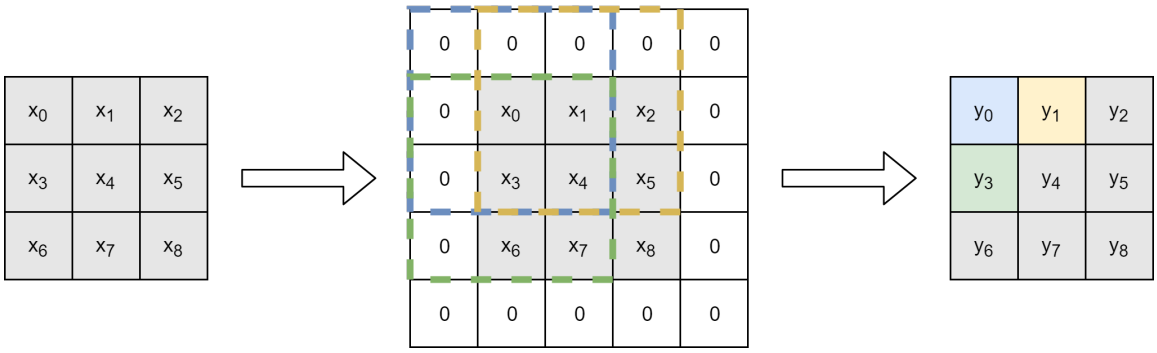


Figure 2.2: Representation of a 3×3 convolution operation applied over a same size matrix, with padding equal to one in both the vertical and horizontal dimensions. If padding had not been used, the result of the convolution would have been 1×1 matrix.

$$\boldsymbol{x} = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \end{bmatrix}$$

$$\boldsymbol{w} = \begin{bmatrix} 3 & 2 & 1 \end{bmatrix}$$

To convolve them, the first thing to do is to flip $\boldsymbol{w}$ horizontally, so it becomes [1 2 3]. Now, imagine a window containing the values of $\boldsymbol{w}$, sliding underneath $\boldsymbol{x}$. The output vector, referred to as $\boldsymbol{y}$, is simply given by the values of $\boldsymbol{w}$ that are underneath $\boldsymbol{x}$, multiplied by the values in $\boldsymbol{x}$ and then summed. A more formal way to describe a convolution is given in Equation 2.4, where k corresponds to a certain index of the output vector, and where the values for $\boldsymbol{x}$ and $\boldsymbol{w}$ in undefined positions is considered to be zero.

$$\boldsymbol{y}[k] = \sum_{j=-\infty}^{+\infty} \boldsymbol{x}[j] \cdot \boldsymbol{w}[k-j] \tag{2.4}$$

The result of the convolution between $\boldsymbol{x} = [1\ 2\ 3\ 4\ 5]$ and $\boldsymbol{w} = [3\ 2\ 1]$ is thus $\boldsymbol{y} = [3\ 8\ 14\ 20\ 26\ 14\ 5]$. Note that some values of $\boldsymbol{y}$ were not calculated using all values in the kernel. The ones that verify that condition are only $\boldsymbol{y} = [14\ 20\ 26]$.

The aforementioned procedure can be easily extended to n-dimensional spaces, and applied to matrices with n dimensions. Moreover, for several applications we want that the input and output to have the same size so, sometimes, *padding* is used (i.e., zeros are placed at the beginning and the end of the input). In the case of two-dimensional inputs, what happens when padding is used is exemplified in Figure 2.2.

A complete CNN architecture is composed of multiple layers, some of which filtering (i.e., convolving) the input to produce intermediate representations. These convolutional layers have parameters (i.e., the values in the kernel) that are learned, so that these filters are adjusted automatically to extract the most useful information for the task at hand. Moreover, typically a convolution is followed by a non-linear function (e.g., ReLU), that helps to increase the representative power of the network by expanding the size of the set of functions that the model is able to learn. Along with the number of filters, there are other hyper-parameters that can be configured on a convolutional layer, such as:

- Kernel size, corresponding to the size of each filter;
- Stride, i.e., the size of the kernel jump while filtering the input;
- Padding, representing the size of the padding applied to the input.

In addition to convolutional layers, a CNN normally also involves other types of layers, including pooling layers. Imagine the case where you have a 8×8 matrix, and assume that this matrix can be divided into four 4×4 matrices (two at the top and two at the bottom). A maximum pooling layer, for example, could choose the maximum value in these four sub-matrices and reduce the original matrix to a 2×2, matrix with the maxima of each sub-matrix in each entry. With this transformation some translation invariance can be obtained. For image analysis applications this means that it is possible to detect the same object, not necessarily in the same place as seen in previous training examples.

When training deep neural models such as CNNs, overfitting can frequently occur. Overfitting is a phenomena that represents an over-adaptation to the training data. An idea to address this problem involves using the dropout technique, originally presented by Srivastava et al. (2014). Dropout consists of randomly deactivating some connections between neurons in the training phase, forcing the remaining neurons to learn the same problem, thus creating a better model that represents the data.

To increase the training speed, the inputs can also be normalized, in the sense that the values can be transformed to be all in the same range, normally between $[0, 1]$. The idea behind batch normalization is to apply this principle to the result of a hidden layer (Ioffe and Szegedy, 2015). This is achieved by normalizing the output of the associated activation function by subtracting the batch mean and dividing by the batch standard deviation. However, doing this causes the weights in the next layer to be no longer optimal. This happens because the gradient descent

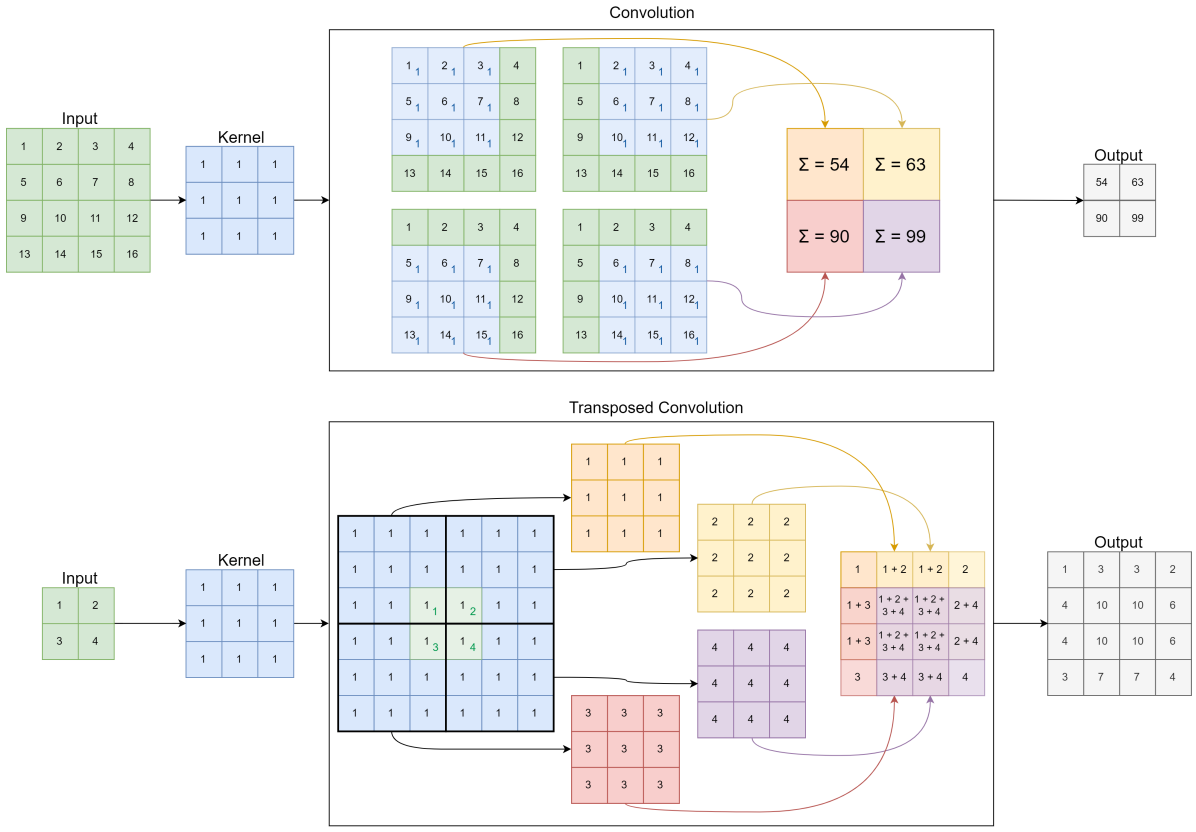


Figure 2.3: Illustration for the arithmetic operations involved in convolutions (at the top) and transposed convolutions (at the bottom), considering the same kernel with size 3×3, respectively for down-sampling or up-sampling the input data.

method undoes this normalization if it is possible to minimize the loss function without those values normalized. The problem is solved by making the mean and the standard deviation of each layer also trainable by the network.

2.1.3 Fully-Convolutional Neural Networks

Standard classification problems include taking an input object (e.g., a two-dimensional matrix of pixel values for an image) and returning a class as output. However, several other instances of problems (e.g., image segmentation) involve returning more complex outputs. In these problems, for instance, instead of assigning a class to the whole image, we want to delimit the area occupied by some object of interest there represented. This involves returning a two-dimensional matrix of values which represents a mask containing the delimitation of that object. Novel network architectures (i.e., models that in the literature are typically referred to as fully-convolutional neural networks or hourglass-shaped networks) for these types of problems frequently involve transposed convolutional layers.

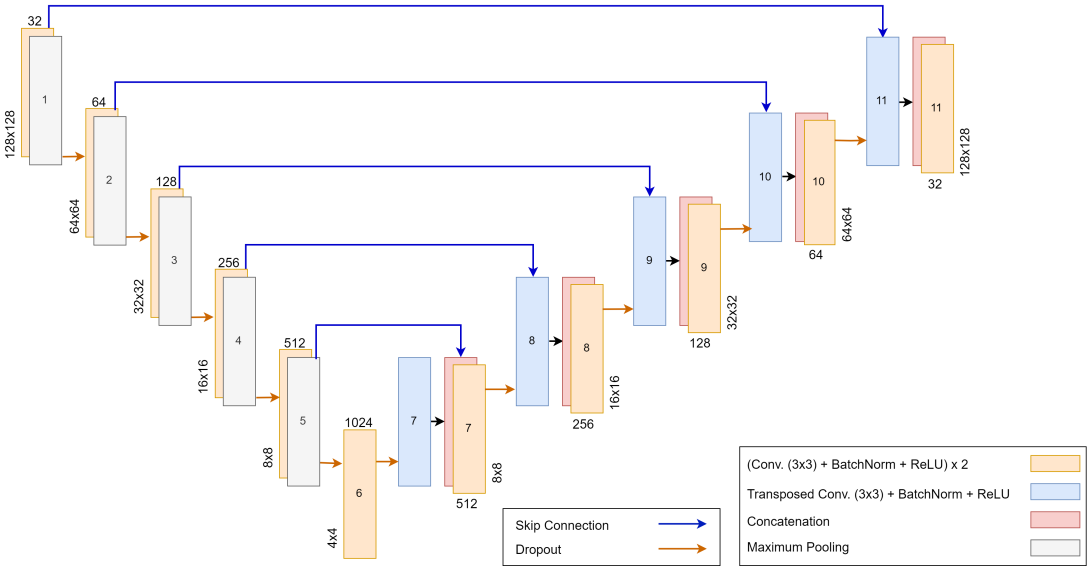


Figure 2.4: Graphical representation for the U-Net (Ronneberger et al., 2015) architecture.

Considering a two-dimensional space and an input matrix of size $n \times n$, a convolution operation outputs another matrix with size $m \times m$, where $m \leq n$ (i.e., usually the inputs have a many-to-one relationship with the outputs). Transposed convolution operations, also often called de-convolutions, will instead output a matrix with size $m > n$, up-sampling the input data and considering that inputs have a one-to-many relationship with the outputs. Figure 2.3 illustrates the differences in the arithmetic operations involved in convolutions and transposed convolutions, considering data with a single input channel and a stride of one, and in both cases also considering a convolution kernel with dimensionality 3×3. The top part of the figure illustrates a standard convolution over an input matrix with dimensionality 4×4, down-sampling the data into an output with dimensionality 2×2, and illustrating the positional connectivity between the input values and the output values (i.e., each output value depends on 9 of the input values). The bottom part of the figure illustrates a transposed convolution, up-scaling an input matrix with dimensionality 2×2 into an output with dimensionality 4×4, and again showing 1-to-9 relationships between the input and the output values.

Leveraging these ideas, Ronneberger et al. (2015) proposed a model referred to as U-Net. This model is one of the most commonly used neural architectures for image segmentation, and is composed by a down-convolutional and an up-convolutional path – see Figure 2.4. The first path is the contraction path (also called an encoder), which corresponds to a traditional stack of convolutional and max-pooling layers that consecutively reduces the size of the representation for an input feature map, to capture the context and extract relevant information. The second path is the symmetric expanding path (also called a decoder), which is used to enable precise localization and uses transposed convolutions combined with traditional convolutions.

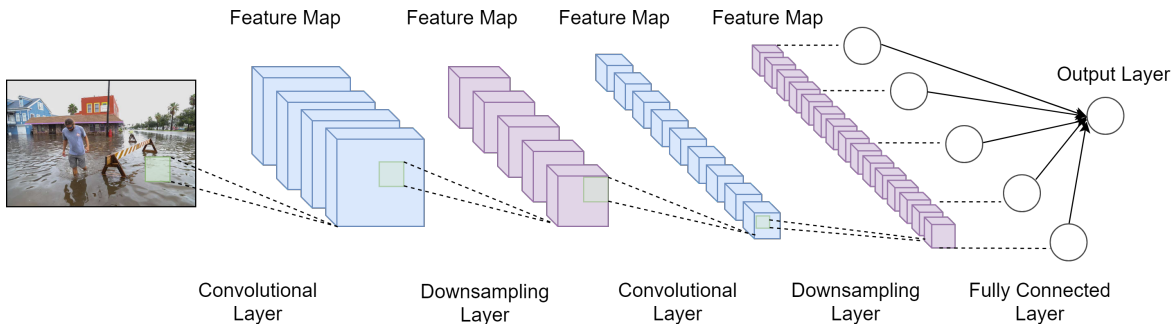


Figure 2.5: Graphical representation for the LeNet5 architecture, adapted from the description presented by Lecun et al. (1998).

2.2 Related Work

This section describes important related work. It starts by detailing recent work about deep learning to address image classification problems. Then, Section 2.2.2 covers models for image segmentation, focusing on remote sensing applications. Section 2.2.3 presents previous work that proposes the use of volunteered geographic information for different types of mapping applications. Finally, Section 2.2.4 presents a brief overview of the presented works.

2.2.1 Image Classification

The first task that I propose to address is the development of a system that, in the presence of an image, can automatically infer if that image depicts a flooding event, and/or the severity of that flood. For this, in this first section of the description of related work, recent studies that have obtained good results in the task of image classification are presented.

2.2.1.1 The DenseNet Neural Architecture

Convolutional Neural Networks (CNNs) have been extensively used in image classification tasks for quite some time (Xie et al., 2018). These methods directly process the pixels (e.g., the RGB values) from input images through a composite of convolution and pooling operations, in order to obtain high-level features that inform the final prediction layer. Early work includes the LeNet5 (Lecun et al., 1998) architecture with its five hidden layers – for which I present a graphical representation in Figure 2.5 – and more recently, due to an increase availability of data and computational power, the number of layers used in CNN architectures has increased significantly (Khan et al., 2018; Altenberger and Lenz, 2018).

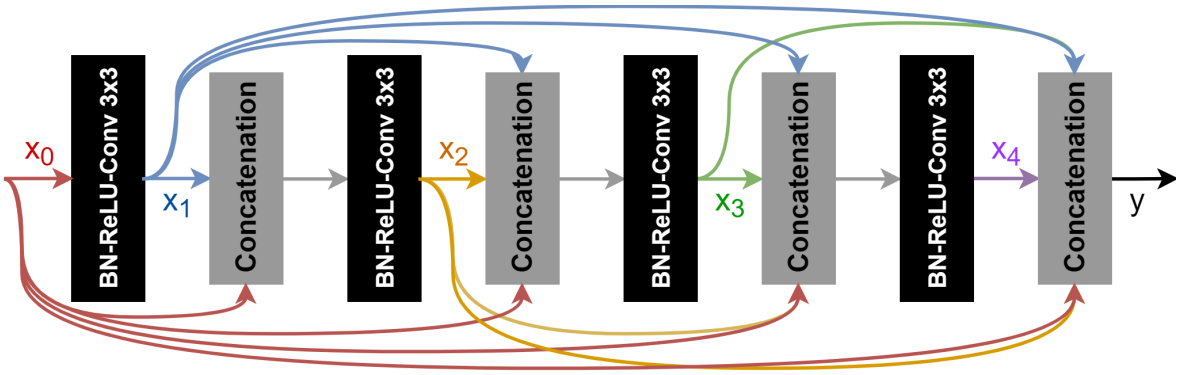


Figure 2.6: Graphical representation for a dense block.

However, this increase in the number of layers can also raise several problems, including the vanishing of information between distant layers during model training, through back-propagation and gradient descent. Recent work has shown that CNNs can be substantially deeper, more accurate, and efficient to train, if they contain shortcut connections between distant layers.

One of the earliest CNN architectures which used the idea of shortcut connections was ResNet (He et al., 2016). In this neural model, information preservation occurs explicitly through additive identity transformations between subsequent layers (i.e., ResNet models use shortcut connections in which we sum the output feature maps of one layer with the corresponding incoming feature maps). Considering H_l as the composition of multiple functions (e.g., batch normalization, convolution, and/or pooling) in a layer l, in the case of the ResNet architecture, $\boldsymbol{x}_l$ (i.e., the output of a layer l) can be formally defined as presented bellow:

$$\boldsymbol{x}_l = \mathrm{H}_l(\boldsymbol{x}_{l-1}) + \boldsymbol{x}_{l-1} \tag{2.5}$$

However, some studies demonstrated that many ResNet layers contribute very little to the final output, and the number of parameters can be quite large. To address these issues, a more recent work by Huang et al. (2017) proposed the idea of dense connectivity, presenting DenseNets. The DenseNet architecture is built using multiple dense blocks, each as shown in Figure 2.6. Each of these dense blocks can be seen as a small CNN where each layer is connected to every other layer in a feed-forward fashion. The elementary operations underneath dense blocks, besides concatenations, correspond to pre-activation batch normalization, followed by a ReLU activation (Agarap, 2018) and then a 3×3 convolution operation.

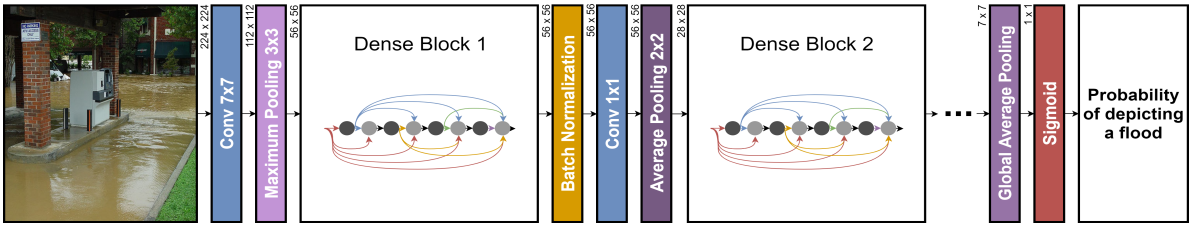


Figure 2.7: Graphical representation for the DenseNet architecture (Huang et al., 2017).

Whereas traditional CNNs with l layers have l levels of connections (i.e., between each layer and the subsequent layer), each dense block has $l \times (l+1)/2$ direct connections. Another difference that DenseNet has, when compared with ResNet, is that the result of the different connections are not summed, and instead concatenated. This means that the output of a layer l, in the DenseNet, can be defined as expressed bellow:

$$\boldsymbol{x}_l = \mathrm{H}_l([\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_{l-1}]) \tag{2.6}$$

This dense connectivity pattern presents several advantages, for instance addressing vanishing gradients, strengthening feature propagation, encouraging feature reuse, and even decreasing the number of parameters. The reduction on the number of parameters comes from the fact that there is no need to re-learn redundant feature maps, since the feature maps learned by any layer of a dense block can be accessed by all subsequent layers.

In a complete DenseNet model, as shown in Figure 2.7, the input image is first processed through a 7×7 convolutional layer with a stride of 2, followed by a 3×3 maximum pooling operation, also with a stride of 2. The result is then passed to a sequence of 4 dense blocks interleaved with transition layers, and finally processed through a 7×7 global average pooling operation before the final output layer. The transition layers are responsible for improving the model compactness, first applying a batch normalization, and then a 1×1 convolution followed by an average pooling operation over all the feature maps produced by the dense block. Representing as m the number of feature maps of a certain dense block, and as θ the compression factor associated to the aforementioned average pooling operation, a transition layer generates $\lfloor \theta \times m \rfloor$ feature maps as output.

Huang et al. (2017) compared the DenseNet and ResNet architectures on multiple datasets, including the ImageNet (Russakovsky et al., 2015) multi-label dataset – see Table 2.3 for some results. DenseNets obtained results on par with ResNets, whilst requiring significantly fewer parameters and computation to achieve comparable performance. The DenseNet-201 is going to be tested on this dissertation in the experiments related to image classification.

2.2.1.2 Efficient Convolutional Neural Networks

In a recent work by Tan and Le (2019), the authors argued that convolutional neural networks must be scaled up in multiple dimensions, since scaling it only one direction (i.e., depth only) will result in rapidly deteriorating gains relative to the computation increase need.

Most of the commonly used CNNs architectures are scaled up by adding more layers (e.g., on aforementioned DenseNet the authors reported experiments with the DenseNet-121, DenseNet-169, DenseNet-201 and DenseNet-264). Typically, the bigger the number (i.e., the number of blocks/layers on the network), the bigger the ability for the network to model a problem and achieve better results. However, as the authors demonstrated, simply going deeper rapidly saturates the gains (e.g., a DenseNet-1000 will not be much more accurate than a DenseNet-264). The other alternatives are to scale up the networks in width and resolution, but associated benefits quickly disappear as well. To address this problem, Tan and Le (2019) proposed a method to efficiently scale up CNNs. The method, referred to as compound scaling, uses a compound coefficient ϕ to uniformly scale the width, depth, and resolution of a certain network, being formulated as follows:

$$\begin{aligned} \text{depth:} \quad & d = \alpha^{\phi} \\ \text{width:} \quad & w = \beta^{\phi} \\ \text{resolution:} \quad & r = \gamma^{\phi} \\ \text{s.t.} \quad & \alpha \cdot \beta^2 \cdot \gamma^2 \approx 2 \\ & \alpha \geq 1, \beta \geq 1, \gamma \geq 1 \end{aligned} \tag{2.7}$$

In the previous expression, α, β, and γ are constants that can be determined by a grid search. Intuitively, ϕ is a user-specified coefficient that controls how many more resources are available for model scaling, while α, β, and γ specify how to assign these extra resources to the network width, depth and resolution, respectively. This equation comes from the fact that a regular convolution operation is proportional to d, w^2 and r^2, meaning that doubling the network depth will double the computational cost, but doubling the width or resolution will increase this cost by four times. Since convolutional operations usually dominate the computational cost, scaling a CNN with Equation 2.7 will approximately increase the cost by $(\alpha \cdot \beta^2 \cdot \gamma^2)^{\phi}$. In the paper, the authors constrained this condition to be approximately equal to 2 such that for any new ϕ, the computational cost will approximately increase by 2^{ϕ}.

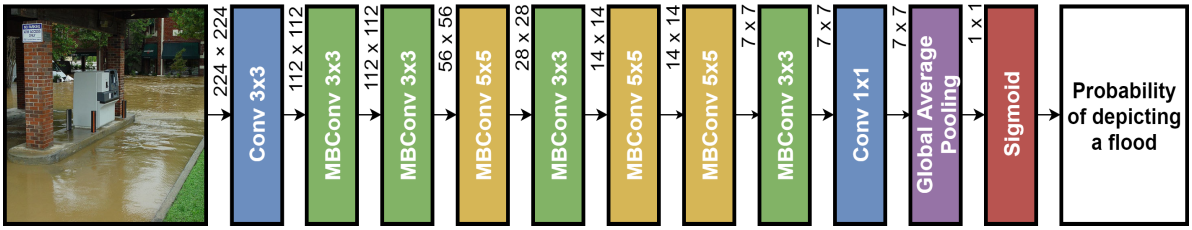


Figure 2.8: Graphical representation for the EfficientNet-B0 (Tan and Le, 2019) architecture.

In order to test this compound scaling formula the authors proposed a model referred to as EfficientNet. The main building blocks of the EfficientNet are mobile inverted bottleneck convolution operations (Sandler et al., 2018; Tan et al., 2019), to which the authors also added squeeze and excitation optimization (Hu et al., 2018). This elementary building block, named MBConv, essentially corresponds to a residual block that connects the beginning with the end through a skip connection. By adding these connections, the block gains the ability to assess earlier activations that were not modified by the convolutional operations. Each of these blocks starts with a 1×1 convolution in order to reduce the number of parameters, followed by a $n \times n$ depthwise convolution (i.e., a particular type of convolution operation where the kernel is divided into multiple kernels across the different channels in order to reduce the number of multiplications that are needed (Chollet, 2017)), reducing even further the number of parameters when compared to typical convolution operations. Afterwards, a channel squeeze and excite operation is applied, which improves channel inter-dependencies at almost no computational cost (Hu et al., 2018). Finally, each MBConv ends with another 1×1 convolution to squeeze the feature map in order to match the initial number of channels.

The authors started by testing a model named EfficientNet-B0, for which I present a graphical representation in Figure 2.8. Then, they scaled this model consecutively using the compound scaling method, in order to obtain a family of EfficientNets from B1 to B7. After applying the constrains specified in Equation 2.7 the authors discovered that the optimal balance for the proposed EfficientNet consisted on 1.20 to the depth, 1.10 to the width and 1.15 to the resolution. This means that to scale up their B0 model, in order to keep it as efficient as possible while expanding the implementation and improving the accuracy, the depth of layers should increase by 20%, the width by 10% and the resolution by 15%.

Tan and Le (2019) also compared EfficientNet models against other CNN architectures on the ImageNet dataset, with striking results when comparing the number of parameters/computations required versus almost every other CNN architecture (i.e., a 5 times reduction while keeping or beating most accuracy values). The EfficientNet-B3 neural model is also going to be used on the reported experiments related to image classification.

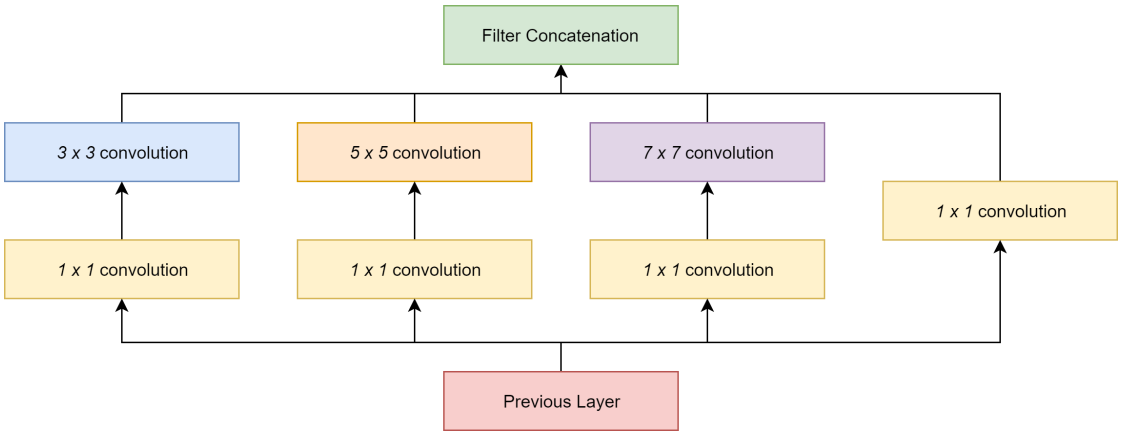


Figure 2.9: Representation of an inception module adapted from the description provided by Szegedy et al. (2015). The inception modules used by GoogLeNet include pooling layers.

2.2.1.3 Going Deeper with Convolutions

Another idea that tries to reduce the number of parameters used in deep convolutional networks are inception modules (Szegedy et al., 2015). An inception module – see Figure 2.9 – avoids the problem of choosing a fixed size to the filters on the same convolutional layer. The key idea is to join multiple convolutions with different sizes, concatenate the result of these operations, and feed the next layer with that result. Small convolutions are placed before the bigger ones, helping to reduce the number of computations and also adding an extra non-linearity, thus, making the model able to mapping even more complex functions.

Szegedy et al. (2015) used inception modules to build a model referred to as GoogLeNet. This network is twenty two layers deep, and starts with simple convolutional layers, after which there are multiple blocks of inception modules, some of them followed by maximum pooling layers. Another interesting addition to the architecture is the change of the second last fully-connected layer with an average pooling layer. With this change not only an increase of 0.6% in the accuracy can be achieved, but they also get a decrease in the number of computations that are needed, since average pooling is computational cheaper than a convolution.

During the training of this model, and to address the vanishing gradient problem, auxiliary classifiers are attached to intermediate layers. These auxiliary classifiers have two purposes. First, they make the layers in the middle of the network more discriminative and thus also able to extract better features. Second, during training, all the losses from each classifier are added up, taking contributions from the auxiliary classifiers lower than the main one. This helps to get a better approximation of the real error of the main classifier that otherwise, in the first layers, would be very small. This model achieved the first place in the ILSVRC 2014 detection challenge (Russakovsky et al., 2015), doubling the accuracy achieved by the winners of the previous year.

In the years following the publication of the original work on the inception modules (Szegedy et al., 2015), other improvements to this model were studied (Szegedy et al., 2016, 2017). Between the proposed improvements stand out the factorization of the largest convolutions (e.g., transform 5×5 convolutions into two 3×3 convolutions) and the usage of residual connections.

2.2.2 Image Segmentation

Besides image classification, the work described on this dissertation also involves image segmentation tasks (i.e., for segmenting flood regions, in which instead of assigning a class to an image, a class is assigned to each pixel of a given image). Some important studies related with image segmentation tasks are presented in the following sections.

2.2.2.1 High Resolution Urban Remote Sensing With Multimodal DNNs

Audebert et al. (2018) proposed an efficient multi-scale approach to leverage both a large spatial context and very high-resolution data in the context of urban remote sensing problems. Specifically, the authors proposed to use the SegNet (Badrinarayanan et al., 2017) architecture as a base model. The SegNet architecture, which is depicted in Figure 2.10, corresponds to a fully-convolutional network, that similarly to the U-Net model described in Section 2.1.3, also follows an encoder-decoder architecture. In the first step the image is encoded in a spatially compressed representation, increasing the receptive field of the network, whereas in the second step, every transposed convolutional layer decodes an increasingly less compressed and spatially larger representation. To increase the spatial awareness the transposed convolutional layers are interleaved with unpooling layers. These unpooling layers revert a previous pooling stage by using the activation at the index computed previously (i.e. the argmax from the max-pooling) and by filling the other values with zero. In order to make multi-scale predictions, the authors propose to branch the model to generate predictions in several resolutions, adding a convolutional layer after a convolutional block of the decoder.

This layer will transform feature maps into the label space. Let P_{full} denote the full resolution prediction, P_{down_d} the predictions at downscale factor d, and f_d a bilinear interpolation that up-samples a map by a factor d. It is possible to aggregate multi-resolution predictions using a simple summation and, during back-propagation, each branch will receive the contributions from the loss of the average prediction and the contribution coming from its own down-scaled loss. Equation 2.8 presents an example with four scales.

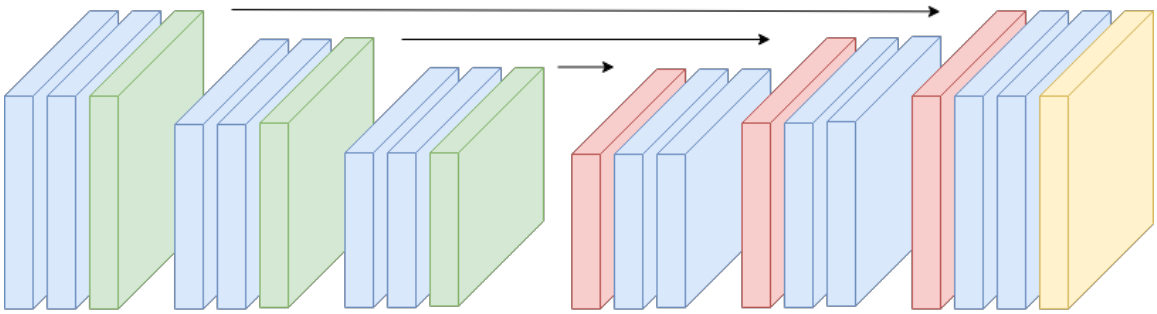


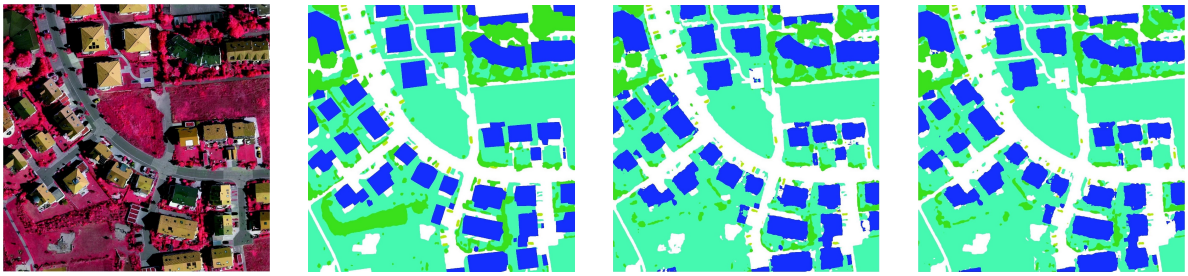
Figure 2.10: Graphical representation of the SegNet architecture, adapted from Badrinarayanan et al. (2017). In blue we have convolutional layers with batch normalization and a ReLU activation. In green we have pooling layers. In red we have transposed convolutional layers, and finally in yellow we have a softmax layer. The first three blocks correspond to the encoder, while the last three to the decoder.

$$P_{full} = \sum_{d \in \{0,2,4,8\}} \mathrm{f}_d(P_{down_d}) = P_0 + \mathrm{f}_2(P_2) + \mathrm{f}_4(P_4) + \mathrm{f}_8(P_8) \tag{2.8}$$

The authors also presented an early fusion architecture, based on FuseNet principle (Hazirbas et al., 2017), where RGB-D images (i.e., images with RGB channels plus depth information) are encoded using two separate parts. Those contributions are summed after each convolutional block. While the depth branch only deals with the depth information, the optical branch deals with both the RGB and depth information. After the early fusion, a single decoder up-samples the joint representation back into the label probability space. In this process only the indices from the main branch will be used. Therefore, it is necessary to choose which data source will be the primary and which one will be the auxiliary data. To deal with this, the authors propose to add a third encoder that does not correspond to any real modality, instead corresponding to a virtual fused data source. Due to this virtual source, the proposed model was called V-FuseNet.

At stage n, the virtual encoder takes as input the previous activations, concatenated with both activations from the other encoders. This strategy makes the architecture symmetrical and therefore relieves the choice of the main source.

Finally, an alternative fusion technique has also been proposed on the paper, relying only on the late feature maps and introducing a residual correction module. This module consists in a residual CNN that takes as input the last feature maps from two fully-convolutional neural networks, trained either with optical data or using an auxiliary data source. The two predictions are averaged to obtain a smooth classification map. Then, the correction module, takes as input the last feature maps from the two deep networks and is re-trained in a residual fashion, learning a small offset to apply to each pixel-probability.



(a) Original image (b) Ground-truth (c) SegNet standard (d) SegNet multi-scale

Figure 2.11: Results obtained with the multi-scale prediction strategy from Audebert et al. (2018) on a excerpt of the ISPRS Vaihingen dataset. It should be noted that the small objects are regularized thanks to the multi-scale prediction. In this figure, white corresponds to roads, blue to buildings, cyan to low vegetation, green to trees, and the small yellow dots to cars.

With this module, the network can learn, for example, that if the auxiliary data is better to recognize vegetation, a bigger weight should be assigned to it. Experiments with the ISPRS semantic labelling datasets of Potsdam[1] and Vaihingen[2] proved that fully-convolutional networks are well suited for semantic labeling of multi-modal very high-resolution urban remote sensing data. Examples for the obtained results can be seen in Figure 2.11. The early-fusion method proved to be adequate to learn stronger features, while the late-fusion approach can recover errors on hard pixels that are missed by all other models.

2.2.2.2 W-Net: Bridged U-Net for 2D Medical Image Segmentation

The most commonly used neural architecture for the segmentation of medical images is perhaps the aforementioned U-Net (Ronneberger et al., 2015) model. As previously stated, the down-convolutional part aims at extracting features for classifying each pixel/voxel, while the up-convolutional part aims at locating regions of interest more precisely. This model is typically chosen to process medical imagery because it has a high performance in small and unbalanced datasets (i.e., in medical-related datasets it is common to have a significantly larger number of samples corresponding to instances where the examined illness is not present).

More recently, Chen et al. (2018b) proposed an extension of the U-Net architecture which essentially bridges two U-Nets together and explores another type of activation function, namely Exponential Linear Units (ELUs), combined with the more traditional ReLU activation. A graphical representation of this model is presented in Figure 2.12. The authors named the proposed approach as W-Net.

[1] http://www2.isprs.org/commissions/comm3/wg4/2d-sem-label-potsdam.html
[2] http://www2.isprs.org/commissions/comm3/wg4/2d-sem-label-vaihingen.html

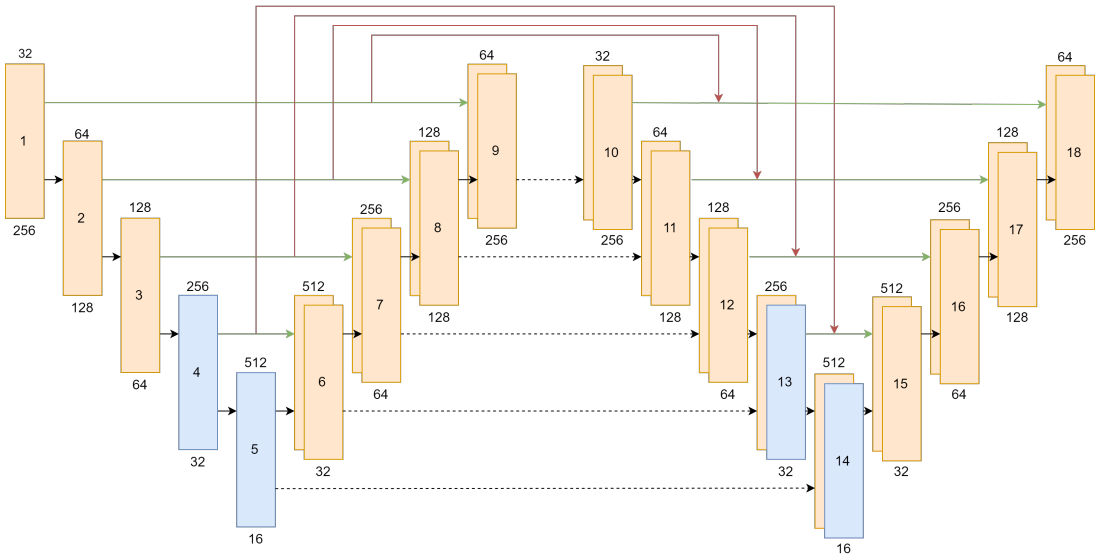


Figure 2.12: Graphical representation of the W-Net model, adapted from Chen et al. (2018b). The orange boxes represent the convolutional layers with two filters, with batch normalization and ELU as activation function. Blue boxes follow the same specification, but they instead consider a ReLU activation function. Dashed lines correspond to bridging connections, while red lines corresponds to skip connections. Finally, green connections correspond to concatenation.

Bridging two U-Net models brings forth the problem of how to transfer the information between the models. The authors ran some experiments and tested both the concatenation and addition of feature maps. The results showed that concatenation is better in the case of connections that bridge the two models, but addition is better in the case of the skip connections between layers in the same network.

Regarding the activation function, the authors presented a solution to optimize the commonly used ReLU – as seen in Section 2.1.1, the ReLU function can get saturated over the negative axis (because the output value is always zero). They propose to use Exponential Linear Units (ELU), as shown in Equation 2.9, that on the positive axis are equal to ReLU but in the negative axis are exponential, thus taking more time to become saturated.

$$\text{ELU}(x) = \begin{cases} x & \text{if } x \geq 0 \\ a(e^x - 1) & \text{otherwise} \end{cases} \tag{2.9}$$

In the paper, the authors argue that replacing all the activation functions with ELU would not solve the saturation problem, since in deeper networks ELU will also become saturated and will have the same behavior as ReLU. To address this issue the authors propose to use both the ELU and ReLU activations simultaneously. This can be done since in the positive axis the two functions have the same behavior, so the result will still be valid.

However, in the saturated negative values, if a layer has ReLU as activation function, all the following ELU layers will be reset to zero, meaning that those layers will not be saturated anymore. The experiments conducted by the authors showed that this simple improvement can, in some cases, increase the accuracy of the model by 2%.

Finally, the third contribution is a new loss function referred to as Cos-Dice. This function is an improvement over the dice loss function usually used in medical image segmentation. The dice loss function uses the Dice Similarity Coefficient (DSC) to generate the training loss, which is calculated as follows:

$$\mathrm{DSC}(GS, SEG) = \frac{2\|GS \cap SEG\|}{\|GS\| + \|SEG\|} \tag{2.10}$$

In the previous equation, GS represents the goal segmentation, while SEG represents the corresponding automatic segmentation. Finally, $\|GS \cap SEG\|$ refers to the overlap region between the two areas. The dice loss function is simply defined as:

$$\mathrm{L}_{Dice} = 1 - \mathrm{DSC} \tag{2.11}$$

This loss function is particularly interesting for unbalanced datasets, but there is still room for improvements. Experiments showed that there is not much difference in the error propagated when the intersection percentage is between 20% and 70%. This will cause an oscillation when the learning rate decreases, and we would like to have a loss function that has a bigger penalty when the intersection area is small, and a small penalty when the area is big. The Cos-Dice function has this property, using q as an adjustable parameter:

$$\mathrm{L}_{CosDice} = \cos^q\left(\frac{\pi}{2} \cdot \mathrm{DSC}\right), q > 1 \tag{2.12}$$

The Cos-Dice function is smoother than the Dice loss when the intersection area is large, and rougher than Dice loss when the intersection area is small. To test the presented architecture, the authors submitted their model to the MICCAI PROMISE12 challenge[3] in medical image segmentation and compared it with the traditional U-Net, along with other models. W-Net outperformed all the other models in almost every accuracy measurement. Besides the original application in medical image segmentation, I believe that the U-Net and W-Net models are also promising approaches for the segmentation of satellite imagery.

[3] `http://promise12.grand-challenge.org/`

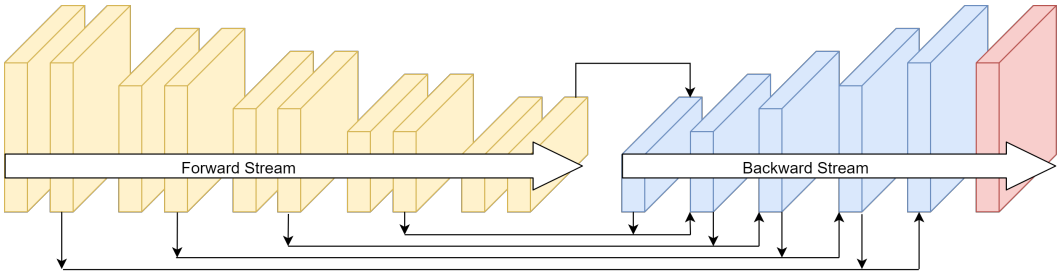


Figure 2.13: Graphical representation of the RiFCN architecture, adapted from Mou and Zhu (2018). RiFCN has a forward stream and a backwards stream. The forward stream is similar to the first part of a typical fully-convolutional network, while the backward stream incorporates recurrent connections. The layer represented in red is a simple softmax layer.

2.2.2.3 Recurrent Network in Fully Convolutional Network for Semantic Segmentation of High Resolution Remote Sensing Images

Fusing multi-level convolutional feature maps and preserving object boundaries are two problems that are under-explored in studies related to semantic segmentation of remote sensing images. Mou and Zhu (2018) proposed to address these problems with methods inspired on Recurrent Neural Networks (RNNs). In an RNN, the output of a layer is passed to the next-layer and fed back to the same layer. The authors applied this principle to a convolutional network and built a bidirectional network model referred to as Recurrent Network in Fully Convolutional Network, or simply RiFCN. A graphical representation is presented in Figure 2.13. This architecture is composed by a forward and a backward stream. At the end, a pixel-wise classification layer is added. While the forward stream is a convolutional neural network for feature extraction (i.e., containing several convolutional layers, grouped in blocks and interleaved with pooling layers), the backward stream embeds high-level features into low-levels ones, and incorporates boundary-aware feature maps from the shallowest layer.

Given an image with size $w \times h$ as input, let $k = \left[\frac{w}{2^l}, \frac{h}{2^l}\right]$ be the resolution of the feature map in the level $l = (0, 1, \ldots, L)$. Let also $\boldsymbol{F}^l_{fwd}$ denote the feature maps generated by the l-th convolutional block of the forward stream. During the backward stream, in each level l, the network takes $\boldsymbol{F}^l_{fwd}$ and the fused feature maps from the previous level, $\boldsymbol{F}^l_{bwd+1}$, as inputs to produce new features of the current level. Finally, consider φ as a function for fusing different feature maps at different resolutions, consider $*$ as the convolution operation, and $\star_s$ as a transposed convolution operation with stride s. $\boldsymbol{W}_{fwd}$ are the weights of the convolution, and analogously $\boldsymbol{W}_{bwd}$ are the weights of the transposed convolution. The function φ can be defined as follows, noting that σ represents a ReLU activation.

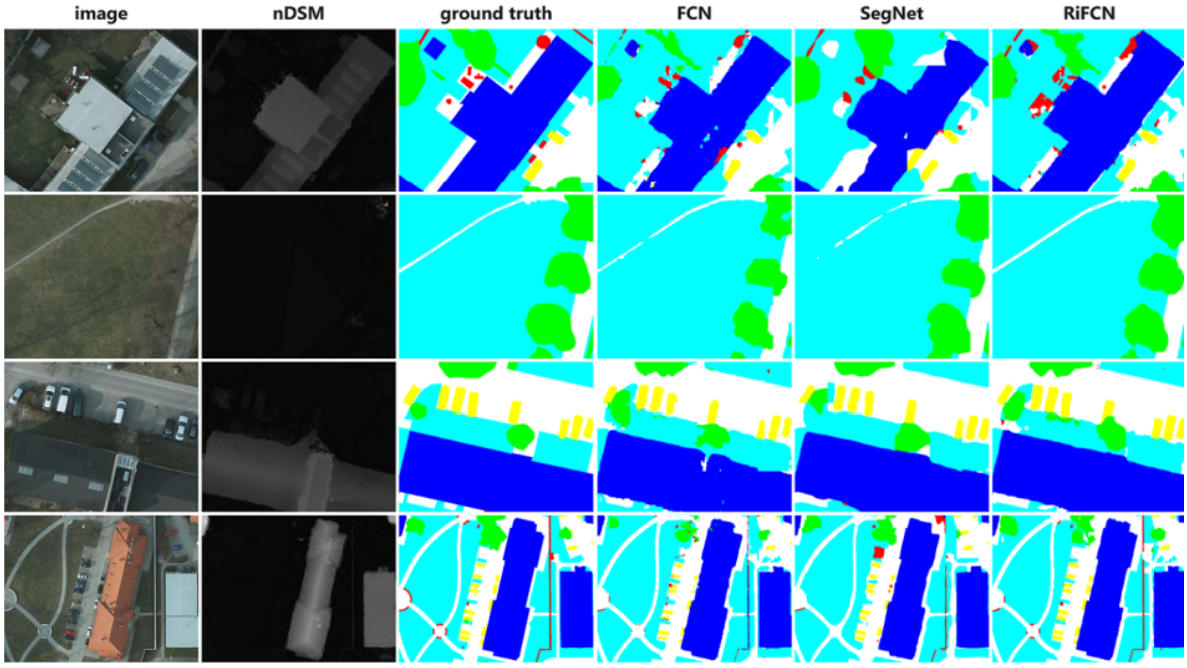


Figure 2.14: Example segmentations obtained over the ISPRS Potsdam dataset. The first column corresponds to the original image, the second to the normalised digital surface model, and the others to segmentations obtained with different models. White corresponds to impervious surfaces, blue to buildings, cyan to low vegetation, green to trees, yellow represents cars and, finally, red is the background.

$$\varphi(\boldsymbol{F}^{l}_{fwd}, \boldsymbol{F}^{l+1}_{bwd}) = \sigma(\boldsymbol{W}_{fwd} * \boldsymbol{F}^{l}_{fwd}) + \sigma(\boldsymbol{W}_{bwd} \star_{s} \boldsymbol{F}^{l+1}_{bwd}) \tag{2.13}$$

The authors tested this architecture using the ISPRS Potsdam and Inria Aerial Image Labeling (Maggiori et al., 2017) datasets, comparing it against SegNet and other simpler models. The authors concluded that the proposed approach improved the quality/consistency of the results, and also the accuracy measurements. Example for the segmentation masks obtained by the authors are presented in Figure 2.14.

2.2.2.4 Symmetrical Dense-Shortcut Fully Convolutional Networks for Semantic Segmentation of Remote Sensing Images

Symmetrical Normal-Shortcut FCN (SNFCN) and Symmetrical Dense-Shortcut FCN (SDFCN) are two models proposed by Chen et al. (2018a) to address the task of semantic segmentation of very-high-resolution remote sensing images. The difference between the two proposed models is that the latter (i.e., SDFCN) contains three additional identity mapping shortcut connections between symmetrical encoder-decoder path pairs, while in the first model (i.e., SNFCN) each layer only has one connection to next layer.

Both models follow a symmetrical encoder-decoder architecture that uses shortcut blocks as the basic element. Considering $\mathrm{BN}(\boldsymbol{x})$ as the batch-normalization operation and $\mathrm{Conv}(\boldsymbol{x})$ as the convolution operation, $\mathrm{F}^{main}_{shortcut}(\boldsymbol{x})$ and $\mathrm{F}^{fine}_{shortcut}(\boldsymbol{x})$ can be defined as follows:

$$\mathrm{F}^{main}_{shortcut}(\boldsymbol{x}) = \mathrm{BN}(\mathrm{Conv}(\mathrm{Conv}(\mathrm{Conv}(\boldsymbol{x}, \boldsymbol{W}_1), \boldsymbol{W}_2), \boldsymbol{W}_3)) \tag{2.14}$$

$$\mathrm{F}^{fine}_{shortcut}(\boldsymbol{x}) = \mathrm{BN}(\mathrm{Conv}(\boldsymbol{x}, \boldsymbol{W}_0)) \tag{2.15}$$

With this, shortcut blocks can be defined as shown in the Equation 2.16. In these blocks, while the main branch learns the residual information of the input data, the shortcut branch helps to propagate gradients directly from output to input.

$$\mathrm{F}_{shortcut}(\boldsymbol{x}) = \mathrm{ReLU}(\mathrm{F}^{main}_{shortcut}(\boldsymbol{x}) + \mathrm{F}^{fine}_{shortcut}(\boldsymbol{x})) \tag{2.16}$$

Due to memory limitations the authors sliced the input images into patches. During the training stage, the patches are sequentially trained by the models. During prediction, the trained models classify each patch from the input image and results are merged with different overlay strategies in a majority voting method. The authors tested both models in the ISPRS Vaihingen and Potsdam datasets, with the SDFCN obtaining a higher score on the accuracy metrics than SNFCN and other models like SegNet. It should also be noted that in both models no pre-training was made (i.e., weights were randomly initialized and afterwards tuned using the training dataset containing satellite imagery).

2.2.2.5 Hourglass Networks for Segmentation and Density Estimation

As evidenced by the previous studies described in the preceding sections, hourglass-shaped networks, such as the U-Nets models, are typically used in image segmentation tasks. In these neural architectures, as shown in Figure 2.4, it is common to have shortcut connections between mirrored layers. These shortcut connections are able to help mitigate the vanishing gradient problem. Oñoro Rubio and Niepert (2018) proposed to add more shortcut connections to these types of networks, namely between different spatial dimensions.

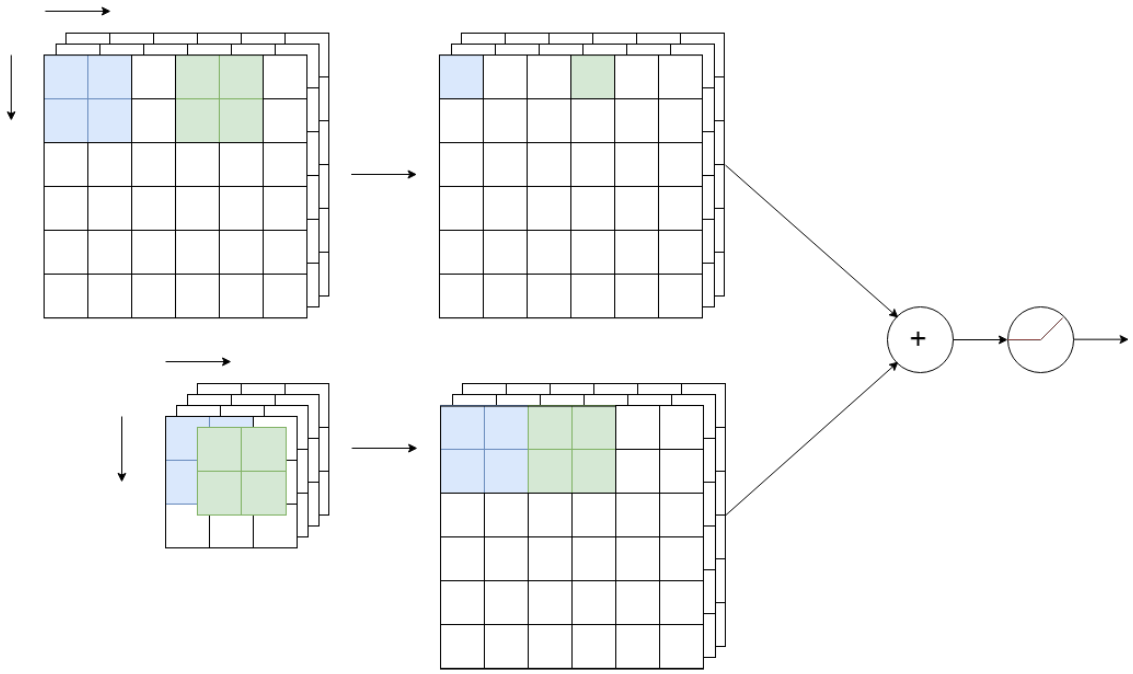


Figure 2.15: Contextual convolution as described by Oñoro Rubio and Niepert (2018). It is possible to verify that a movement on the spatially more extensive tensor corresponds to a fraction of a movement on the smaller tensor.

To address the problem of connections between different dimension convolutions the authors propose to add contextual convolutions that connect the bottleneck representation (i.e., the representation created in the middle of the hourglass network) with layers ahead. Let $\boldsymbol{T}_1$ be a feature map of dimensions $w_1 \times h_1 \times d_1$ and $\boldsymbol{T}_2$ another feature map with dimensions $w_2 \times h_2 \times d_2$, such that $w_2 > w_1$ and $h_2 > h_1$. The authors purposed to tie the movement of the filter operating over $\boldsymbol{T}_1$ to the movement of the filter operation in $\boldsymbol{T}_2$.

Assuming that both convolutional layers have the same stride of one and the same padding strategy, one movement of the filter over $\boldsymbol{T}_2$ to the right (to the bottom) corresponds to $\lfloor w_1/w_2 \rfloor$ movements to the right ($\lfloor h_1/h_2 \rfloor$ to the bottom) of the filter over $\boldsymbol{T}_1$. A graphical representation of this process is presented in Figure 2.15.

The authors used the aforementioned technique and changed the U-Net architecture to use their convolutional filters, building a model referred to as Contextual U-Net. The authors tested the proposed model in the ISBI 2012 dataset[4], where the Contextual U-Net significantly outperformed the traditional U-Net in all accuracy measurements.

[4] `http://www.brainiac2.mit.edu/isbi_challenge/home`

2.2.2.6 Hourglass-Shape Network Based Semantic Segmentation for High Resolution Aerial Imagery

A problem that arises when using Fully Convolutional Networks (FCN) in an encoder-decoder fashion is that the transposed convolutional layers can introduce classification ambiguities. This happens because of the up-sampling process, which can cause some detail to be lost. A model that tries to mitigate this problem was proposed by Volpi and Tuia (2017), named Full Patch Labeling by Learned Up-Sampling (FPL). This model uses multiple transposed convolutional layers to progressive up-sample the classification scores, being different from a standard FCN since each convolutional module uses different convolutions, with different values for stride and kernel size. However, both the FPL and FCN architectures have two main problems: the insufficient spatial information in the decode path and the lack of contextual information. These problems cause both architectures to misclassify small objects (e.g., cars in aerial images) and, due to the lack of contextual information, it becomes difficult to correctly infer classes in areas that, for example, contain shadows. As seen previously, SegNet uses unpooling layers in the decoder path to mitigate the problem of insufficient spatial information. However this comes with a large cost since the number of parameters to train increases for almost the triple when comparing with FPL neural architecture.

Liu et al. (2017) proposed an architecture referred to as Hourglass-Shaped Convolutional Network (HSN). This model has about the same number of parameters as the FLP but without the two former problems. To achieve this, inception modules are used and, like SegNet, skip connections from the encoder to the decoder are also added. Another improvement proposed by the authors for semantic segmentation on high-resolution remote sensing imagery is to use weighted belief propagation in a post-processing phase. The problem is that the output of the network may have some zigzag patterns in the segmentation border.

The authors tested the proposed HSN architecture over the Vaihingen dataset and observed an accuracy around 97% between the two top predictions, showing that most of the time, the right labels lie in the top two scores determined by the network. This is where the post-processing phase is useful. Let $d_i = f_i(c_1) - f_i(c_2)$, where $f_i(c_n)$ refers to the n-th top value of the score function to a certain pixel i. Therefore, d_i can be seen as the confidence of the output for the pixel i. The classification problem was then formulated using a Markov Random Field (MRF). A node i on the MRF corresponds to a pixel in the original image, which is directly connected to its four spatial neighbors. Let y_i denote the class label assigned to node i. The goal is then to minimize the following energy function:

$$\mathrm{E} = \sum_{i \in \mathrm{img}} \mathrm{E}_d(y_i) + \sum_{i,j \in \mathrm{img}} \mathrm{E}_s(y_i, y_j) \tag{2.17}$$

In the previous equation, E_d refers to the data energy term describing how confident the estimated label y_i is, and E_s is the smoothness energy which penalizes the inconsistency between node i and its neighbors n_i. Both terms are computed as follows:

$$\mathrm{E}_d(y_i) = \frac{\exp\left(f_i(y_i)\right)}{\sum_j \exp\left(f_i(j)\right)} \tag{2.18}$$

$$\mathrm{E}_s(y_i, y_j) = v_2 \cdot \exp\left(-\frac{1 - \delta(y_i - y_j)}{t}\right) \tag{2.19}$$

In Equation 2.19, both v_2 and t are hyper-parameters, and $\delta(x)$ is the Dirac delta function, which returns the infinite value in the point zero and zero for all the other values. A Weighted Belief Propagation (WBP) algorithm can be applied to iteratively minimize the energy function E. Considering $m_{ij}(y_i)$ as the message passed from node i to node j, and w_i as the weight for node i, we have that $b_i(y_i)$ is the belief, which represents how confident the node i is to take label y_i. The WBP algorithm can be defined with the two equations bellow. After that, the messages are updated until convergence. The final label $\hat{y}_i$ at node i is determined by $\hat{y}_i = \arg\max_{y_i} b_i(y_i)$.

$$m_{ij}(y_i) = w_i \sum_{y_i}^{C} \mathrm{E}_s(y_i, y_j)\mathrm{E}_d(y_i) \prod_{y_k \in N_i \setminus y_i} m_{ki}(y_i) \tag{2.20}$$

$$b_i(y_i) = \mathrm{E}_d(y_i) \prod_{y_k \in N(y_i)} m_{ki}(y_i) \tag{2.21}$$

Due to the memory limitations, in the inference stage, the authors concluded that the input high-resolution images can be sliced into small non-overlapping patches. However, this cause inconsistent segmentation across the patch borders. To address such boundary effects, Overlap Inference (OI) is employed whereby input images are split into overlapped patches. At the output of the network, the class scores in overlapped areas are averaged.

2.2.2.7 From Satellite Imagery to Disaster Insights

The segmentation of satellite imagery can be used for flood mapping, for instance by looking for and delimiting damages in man-made structures (e.g., roads and buildings). A recent work that combines this idea with the usage of deep learning comes from Doshi et al. (2018).

The authors began by using a semantic segmentation model, referred to as Residual Inception Skip Network (Doshi, 2018). Instead of the traditional convolutional blocks, this model uses inception modules, like the ones presented by Szegedy et al. (2015) with the GoogLeNet neural architecture previous described, in a encoder-decoder fashion. Pre-trained over ImageNet, this model has the goal of extracting man-made features on imagery of before, and after, the disaster affected area. Next, the authors computed the difference of the two segmentation masks to identify changes. To eliminate some noise, dilation with a radius of five pixels, followed by the removal of small connected components, was also applied.

The authors also introduced a metric, referred to as Disaster Impact Index (DII), that can be used to find priority regions to coordinate relief efforts. DII can be mathematically defined as follows, where the numerator denotes the number of pixels in the grid which have the feature detected in the pre-disaster CNN mask but not in the post-disaster mask, and the denominator denotes the number of feature pixels predicted in each grid pre-disaster, averaged over the whole region (i.e., n_{grid} is the total number of grids in the region):

$$\text{DII} = \Delta Pred = \frac{\|\eta Pred_{before} = 1 \& Pred_{after} = 0\|_{grid}}{\left(\frac{1}{n_{grid}}\right) \sum_{i=1}^{n_{grid}} \|\eta Pred_{before} = 1\|_{grid_i}} \tag{2.22}$$

To validate the proposed approach the authors tested their model in satellite images of the Hurricane Harvey (2017) flood, and the Santa Rosa (2017) fire, in three settings (i) prediction of pixel-wise change before and after disaster, and evaluation against pixel-wise ground-truth labels, (ii) aggregation of pixel-wise changes over grids of size 256×256 to compute the DII, which was then evaluated against gridded ground-truth computed from pixel-wise labels, and finally (iii) computing the DII, but evaluating against labels for impact area. Using the second configuration, the authors obtained the best scores, i.e., a F1-Score of 81.2% on the gridded flood dataset, and 83.5% on the gridded fire dataset.

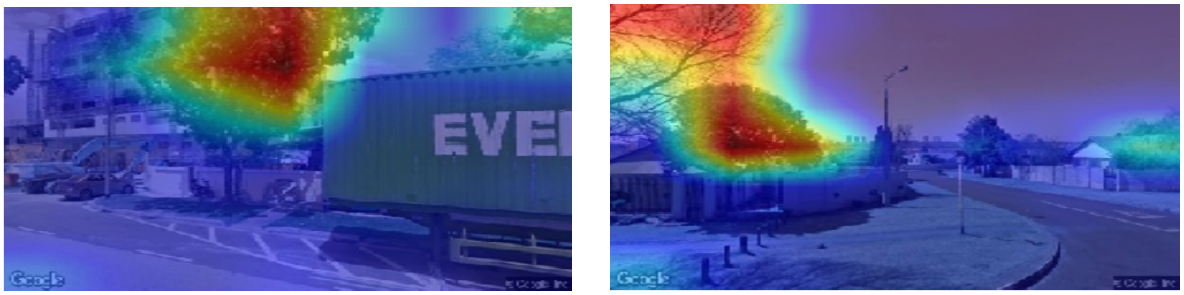


Figure 2.16: Examples of the results obtained by Cai et al. (2018) from applying Grad-CAM. Areas closer to red have a more positive contribution to a higher prediction of GVI than the contribution of areas closer to blue.

2.2.3 Mapping with Volunteered Geographic Information

The last part of the related work section presents studies that discuss the usage of ground-level images (e.g., obtained from social networks), in some cases combined with satellite imagery, to support the creation of maps, for instance in the context of flood mapping.

2.2.3.1 Deep Learning for Large-scale Quantification of Urban Tree Cover

Using aerial images for inferring tree cover is particularly challenging, since there is a lack of information that can only be obtained from images taken from the ground. Cai et al. (2018) proposed to detect the presence of vegetation using street-level imagery obtained from Google Street View[5] (GSV). One approach tested by the authors was to perform semantic segmentation with the goal of identifying the pixels that may contain vegetation, to then calculate the Green View Index (GVI) from those pixels. This was done using the PSPNet (Zhao et al., 2017).

Another approach tested by the authors, alternative to pixel-wise classification, involved the use of the already addressed ResNet (He et al., 2016) neural model to directly calculate the GVI. To benchmark the proposed models the authors used the original *Treepedia*[6] method, that uses unsupervised segmentation with the same goal of calculate the GVI.

Due to the lack of an intermediate image segmentation mask, the authors concluded that it is difficult to interpret the learning process of the ResNet model. To address this limitation, the usage of a method that allows the visualization of the features learned by intermediate layers, referred to as Gradient-weighted Class Activation Map (Selvaraju et al., 2016) or simply Grad-CAM, is also proposed.

[5] `http://www.google.com/streetview`
[6] `http://senseable.mit.edu/treepedia`

Through partial linearization and taking the rectified linear function of a linear combination of the feature maps and the weights, the authors were able to visualize areas of the original input image that contributed positively to the prediction of a particular class (note that, in Section 3.2.1 I will explain in more detail how these activation/attention maps are generated). Figure 2.16 presents some examples of the results obtained by the authors. With the usage of the two aforementioned models the authors significantly outperformed the Treepedia method. The model that directly calculates the GVI achieved better results in the accuracy measurements and was faster in the evaluation process.

2.2.3.2 Landuse Characterization Using Ground-Based Pictures: A Deep Learning Solution Based on Globally Available Data

Srivastava et al. (2018) also tested the usage of images obtained from GSV to perform land use classification. To achieve this, the authors built a model referred to as VIS-CNN – see Figure 2.17 – that works as follows. Firstly, several images of a certain point of interest, which is intended to be classified, are collected from GSV and passed as input to a CNN.

In particular, the authors selected the VGG16 (Simonyan and Zisserman, 2014) model, pre-trained over the ImageNet dataset. Then, using the features extracted from that convolutional neural network, some fully connected layers are applied to extract a feature vector for each image. These feature vectors are then aggregated to obtain a fixed-size vector that can be used as input to another model. To aggregate these vectors, the authors tested two different strategies: *maximum* and *average* aggregation. The first strategy (i.e., the maximum aggregation) selects the maximum value of each vector, while the second uses the average between the elements of those vectors, to create a single vector. This vector is then passed as the input to a final fully connected classifier which maps the aggregated feature vector to the probability of that representation belonging to a certain class.

The authors tested the proposed model over a dataset of images from Île-de-France in France, against two other models. In the first model (i.e., CNN-MV), a majority voting approach was used (i.e., each image of a certain place is classified and then the class that appeared the most is the predicted class for that place). In the second model (i.e., CNN-AVG), the average between the different feature representation of each image was averaged and classified using a multi-layer perceptron. The VIS-CNN with the average aggregator was the model that obtained the best score in all the accuracy measurements.

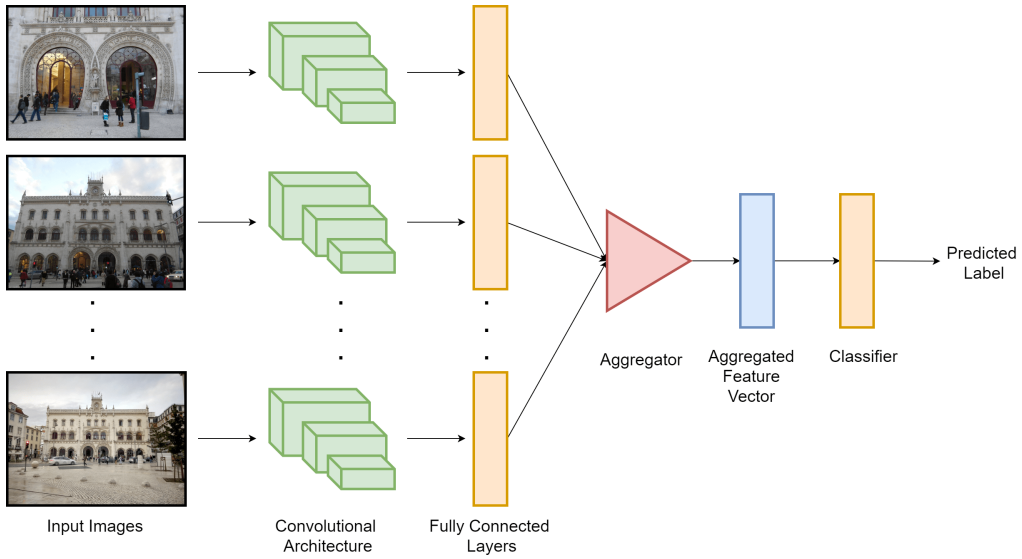


Figure 2.17: Representation of the VIS-CNN model proposed by Srivastava et al. (2018).

2.2.3.3 Georeferenced Social Multimedia as Volunteered Geographic Information

A large number of images are shared every day on social networks, most of them containing precise information about the location from which they were taken. This raises an opportunity to use these images to make geographical discoveries, since in certain cases, they can be more useful than aerial images (e.g., it is often possible to observe physical materials not observable from above, at a significantly finer resolution).

Newsam and Leung (2019) presented several case studies in which they used geo-referenced photos as Volunteered Geographic Information (VGI) to perform geographical discoveries. For instance, in an early study, the authors used VGI from Flickr and Geograph[7] to create a binary land classification for an area of 100×100 km^2 in Great Britain (Leung and Newsam, 2010). They achieved this using a supervised classification framework based on support vector machines, that automatically labels individual images as being of developed or undeveloped scenes and then aggregates these labels to produce a two-class land cover map. Each image was assigned to a 1×1 km tile based on their geographical information, and then the authors tested two ways to produce the maps. In the first approach, the class for each 1×1 km tile was assigned taking into account the ratio of images classified with the label developed to the total number of images in the tile, creating a fraction map. In the second approach, the authors applied a threshold to that fraction map – see Figure 2.18. The second approach resulted in an improvement of 3.8% in terms of accuracy, achieving 75% of accuracy in the images obtained from the Geograph. A smaller accuracy of 66.9% was obtained with images from Flickr.

[7] `http://www.geograph.org.uk/`

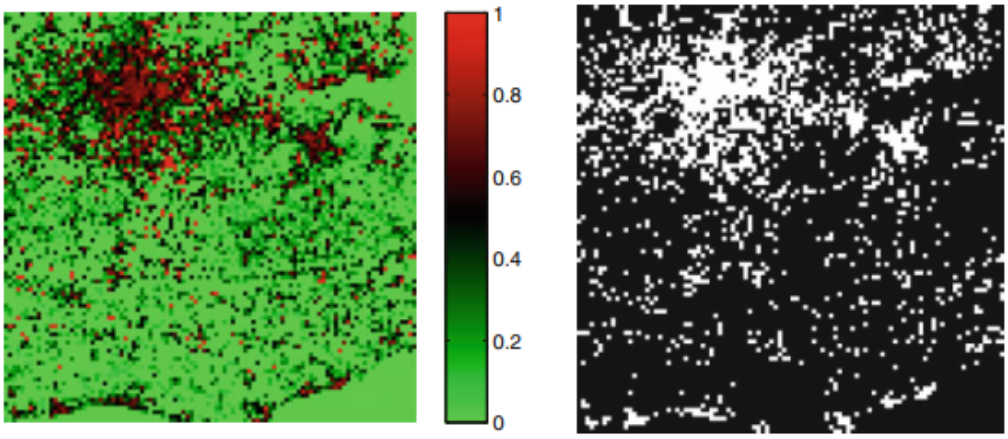


Figure 2.18: Results obtained by Newsam and Leung (2019) in their first case study, where VGI is used to identify developed and underdeveloped areas. In the left is the fraction map obtained, and in the right is the classification obtained after applied some threshold.

This was an expected result, since the images from Geograph were often taken with the purpose of supporting geographical discoveries. Furthermore, a similar approach was used by the authors for land use classification in two university campuses, obtaining good results overall.

2.2.3.4 Investigating the Feasibility of Geo-Tagged Photographs as Sources of Land Cover Input Data

Another important work that investigates the feasibility of using geographically-tagged photographs for land cover applications was presented by Antoniou et al. (2016). In this paper the authors try to find out if it makes sense to use images shared in social networks like Flickr, Panoramio and Geograph, to make geographical discoveries. In addition, the authors also discuss which are the most relevant metadata for each type of usage. The authors address three possible types of use, namely (i) calibration of Land Cover and Land Use (LCLU) maps, (ii) validation of LCLU use maps, and (iii) verification of remotely sensed products.

First, the authors discussed the importance of various types of metadata commonly available in association with images shared on social networks (e.g., date, tilt, direction/orientation, weather, information about the photographer, etc.), and each metadata element was marked *essential*, *desirable* or *unnecessary*.

The majority of the analyzed metadata was marked as desirable, but not essential for all the use cases (e.g., the accuracy of the GPS-enabled device would be useful to have a more precise classification, but not critical). The different requirements for the three analyzed use cases do not really differ in terms of the metadata requirements, but rather in the amount of photographs available and their spatial distribution.

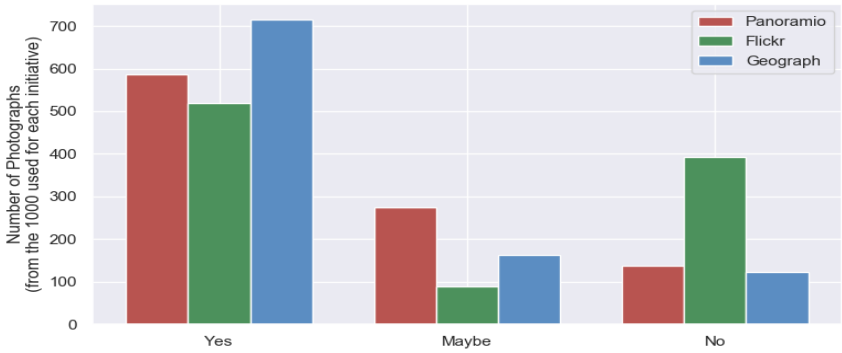


Figure 2.19: Mean number of photographs assigned to the different classes by five volunteers in the work of Antoniou et al. (2016). Most of the classified photographs are useful for making geographical discoveries.

In a second step, approximately 3000 photographs from the three aforementioned sources were manually classified by seven volunteers into *yes*, *maybe* and *no* with regard to their usefulness to make geographical discoveries. A photograph classified with *yes* indicates usefulness, *maybe* indicates that more than one type of land cover type was identified, and *no* was used for when no usefulness was found. After removing some noise in the classification, the authors kept the classification made by five of the seven volunteers. A plot that shows the obtained results is presented in Figure 2.19. Like in the work of Newsam and Leung (2019), more usefulness is found on the images from Greograph when compared with Flickr. Nonetheless, 52% of the images are still useful, and since Flickr is a social network with more photographs and users, good results can still be expected, if more photographs are used.

2.2.3.5 A Near Real-Time Flood Mapping Approach by Integrating Social Media and Post-event Satellite Imagery

There are some problems when using VGI to do flood mapping. For example, simply overlaying different data sources might be problematic, since VGI might have biased distributions. Moreover, a high density of VGI resources related to floods does not necessary mean a high probability of a flood occurrence. To address the aforementioned problems Huang et al. (2018) proposed to combine post-event satellite imagery with VGI for a more comprehensive and rapid flood mapping. The authors also developed a kernel-based weighting algorithm to assign different confidence levels at the points where VGI is available, based on the Normalized Difference Water Index (NDWI) distributions in the post-event image. This way, a more spatially continuous, near real-time flood probability distribution is extracted during a flood event.

Assume that a location i is flooded, and that some other location, considered as j, refers to a random location in the same area. As the elevation of j from i increases (or simply the distance of j increases), j is less likely to be flooded. This allows to define a Differential Height (DH) to the location j in relation to i. DH is defined as follows, where h_i represents the elevation of the location i, and h_j the elevation of j:

$$\mathrm{DH}_{ij} = \begin{cases} 0 & \text{if } h_i - h_j < 0 \\ h_i - h_j & \text{otherwise} \end{cases} \tag{2.23}$$

Also, given a location i where VGI is available, the Inverse Distance Weight (IDW) implemented Probability Index Distributions (PID) at a random location j is calculated as follows, where d_{ij} represents the geospatial distance between location i and j, and α and β are power parameters to adjust the influence of DH_{ij} and d_{ij}, respectively:

$$\mathrm{PID}_{ij} = (\mathrm{DH}_{ij})^{\alpha} \times \frac{1}{(d_{ij})^{\beta}} \tag{2.24}$$

An ill-conceived theory is that the existence of many flood-related tweets, in a certain area, indicates a strong possibility of flooding. However, this may only mean that there is a greater user density. To address this problem, the authors proposed an assignment of weights to different locations taking into account the wetness variations from satellite images. One way to calculate the NDWI, where ρ_{green} and ρ_{SWIR} are surface reflectance of green and Short-Wave Infrared (SWIR) bands, respectively, is the following:

$$\mathrm{NDWI} = \frac{\rho_{green} - \rho_{SWIR}}{\rho_{green} + \rho_{SWIR}} \tag{2.25}$$

The authors also developed a Gaussian kernel weighting algorithm to aggregate the NDWI of all pixels in the kernel centered at the VGI locations. The NDWI influence is gradually weaker when a pixel is further away from the location of the VGI. The Gaussian kernel is given by the equation bellow, where h is an adjustable parameter:

$$\mathrm{G}(x, y) = \left(\frac{1}{2\pi h}\right) \cdot \exp\left(-\frac{x^2 + y^2}{2h}\right) \tag{2.26}$$

With the Gaussian kernel and with the NDWI it is possible to calculate the Wetness (W) for a random position j, in relation to a location i where VGI is available:

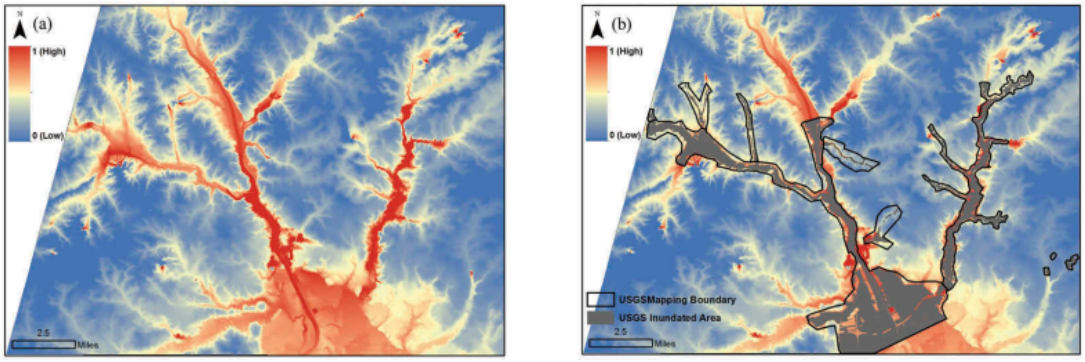


Figure 2.20: In the left is the final Flood Probability Map (FPM) obtained by Huang et al. (2018). In the right is the FPM overlaid with the USGS inundation map and its map boundary.

$$\mathrm{W}(i_x, i_y) = \iint_{(j_x, j_y) \in R} \mathrm{G}(j_x, j_y) \cdot \mathrm{NDWI}(j_x, j_y)\, dj_x\, dj_y \tag{2.27}$$

The final Flood Probability Model (FPM) for j is given by:

$$\mathrm{FPM}_j = \sum_i \mathrm{W}(i, j) \times \mathrm{PID}_{ij} \tag{2.28}$$

The authors tested this methodology in images from a flood that occurred in South Carolina in the year of 2015, and compared the flood map obtained with the proposed method against the one obtained by the United States Geological Survey. The obtained results from their experiments are shown in Figure 2.20.

2.2.3.6 Extraction of Pluvial Flood Relevant VGI by Deep Learning from User Generated Texts and Photos

Another important work that tries to use VGI to do flood detection was presented by Feng and Sester (2018). The authors proposed a framework to extract and analyze VGI collected from Twitter, corresponding to the process illustrated on Figure 2.21. After some pre-processing of the content obtained from Twitter (e.g., extracting the image content from tweets that contained hyperlinks to images in Instagram) the authors tested different classifiers for each modality (i.e., image and text classification).

Regarding text classification, the approaches tested consisted of random forests, logistic regression, SVM (with linear and radial basis function kernels), and convolutional neural network models. All these methods were trained based on tf-idf (i.e., term frequency-inverse docu-

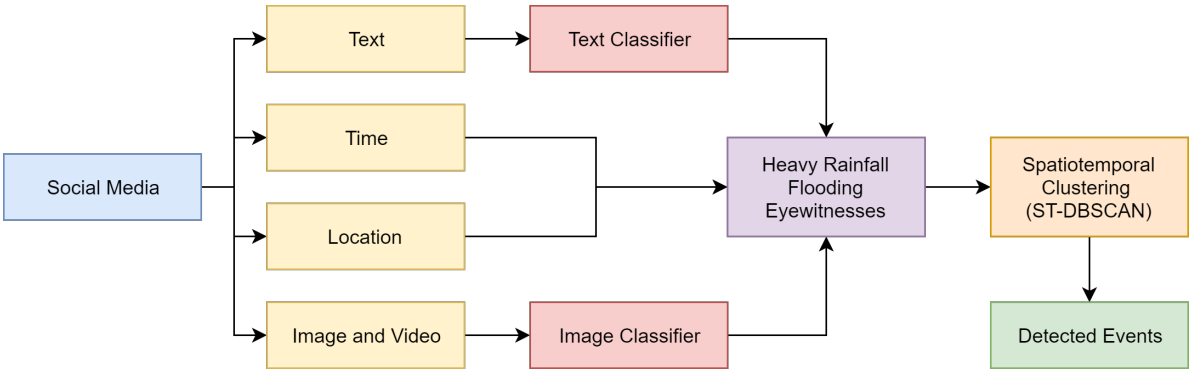


Figure 2.21: Outline of the framework proposed by Feng and Sester (2018) for flood relevant VGI extraction from social media.

ment frequency) features. The approach using convolutional neural networks reached a higher F1-Score, although its training time was considerably higher than that of the other methods. However, their prediction time after training (which is relevant to perform flood mapping) is practically the same as that of the other methods.

Regarding image classification, since the proportion of flooding and rainfall relevant photos was very small with respect to the whole dataset, the authors decided to build three separate datasets using images from Twitter and from a search engine.

The first dataset contained only rainfall and flooding irrelevant images, the second contained relevant images and the last one contained only images of water surfaces. Then, the authors trained a classifier with the images from the first and from the second datasets, and another with the images from the second and the third. Only images classified positively by the two classifiers were considered to contain a flood. In this task, the tested classifiers were based on logistic regression, multi-layer perceptrons, random forests, gradient boosted trees and xgboost (i.e., an optimized gradient boosting implementation). The xgboost model outperformed the other methods with a higher accuracy and F1-Score.

Finally, since the location from where the analyzed tweets were posted may not coincide with the location of the event, a spatiotemporal clustering algorithm was used to address this problem. Both text and images positively classified as being flood related (with their geographical information) were passed as input to the ST-DBSCAN clustering algorithm (Birant and Kut, 2007), and only then a certain location is marked as containing a flood. Since unsupervised learning techniques are outside of the scope of this work, the details of this approach will not be explained here.

The authors tested the framework in floods that occurred in London, Paris, and Berlin, obtaining good results overall. In addition, the authors also verified that depending on the type of floods (pluvial and fluvial) the spatial distribution of the tweets related with floods changes. That is, when there is a fluvial flood, the information comes from points closer to the river that overflowed, while if it is a pluvial flood the images are much more distributed across space.

2.2.3.7 Land Cover Classification from Geo-Tagged Field Photos

Xu et al. (2016) also tested the usability of creating land cover classification supported by geo-referenced photographs. The authors started by using an inception based CNN architecture (Szegedy et al., 2015), with the weights initialized from the resulting of training over the ImageNet dataset. The last layer of the chosen model was removed, in order to use the network as a feature extractor, and another type of classifier over the features obtained was used. This is a method referred to in the literature as transfer learning, which shows its advantages when the research dataset is not similar to the original dataset. Because of its efficiency, the authors selected a multinomial logistic regression model as the classifier. Considering $\boldsymbol{x}$ as the input (i.e., the result of the feature extractor), k the number of the desired classes, and θ the multinomial logistic regression parameter that is intended to be learned, $\mathrm{h}_\theta(\boldsymbol{x})$ represents the probability of $\boldsymbol{x}$ belonging to each k class. A formal way of defining a multinomial logistic regression is presented bellow:

$$\mathrm{h}_\theta(\boldsymbol{x}) = \begin{bmatrix} P(y=1|\boldsymbol{x},\theta) \\ \dots \\ P(y=k|\boldsymbol{x},\theta) \end{bmatrix} = \frac{1}{\sum_{j=1}^{k} \exp\left(\theta^T \boldsymbol{x}\right)} \begin{bmatrix} \exp\left(\theta^{(1)T} \boldsymbol{x}\right) \\ \dots \\ \exp\left(\theta^{(k)T} \boldsymbol{x}\right) \end{bmatrix} \tag{2.29}$$

Training the previous model consists in finding the values for θ that minimize the difference between the predicted and the actual probability of a certain sample belonging to each class. Since the dataset is unbalanced, a weighted variation of gradient descent was used to find the optimal θ. Considering $\boldsymbol{w}$ as the vector with all the weights (i.e., $\boldsymbol{w}[i]$ represents the weight of the element i), n represents the total number of examples and n_i the number of examples of a certain category. Each entry of $\boldsymbol{w}$ is then calculated as follows:

$$\boldsymbol{w}[i] = \frac{n}{n_i \times k} \tag{2.30}$$

Team	Visual	Metadata	Visual + Metadata	Open Run	Open Run
MultiBrasil	87.88	62.53	85.63	91.59	41.13
WISC	62.75	74.37	80.87	81.61	81.99
CERTH-ITI	92.27	39.90	83.37	–	–
BMC	19.69	12.46	11.93	11.89	11.79
UTAOS	95.11	31.45	68.12	89.77	**82.68**
RU-DS	64.70	**75.74**	85.43	–	–
B-CVC	70.16	66.38	83.96	75.96	–
ELEDIA@UTB	87.87	57.12	90.39	**97.36**	–
MRLDCSE	**95.73**	18.23	92.55	–	–
FAST-NU-DS	80.98	71.79	80.84	–	–
DFKI	95.71	77.64	**97.40**	64.50	–

Table 2.1: Results (expressed using the mean over AP@{50, 100, 250, 480}) for the classification task of the Multimedia Satellite Task at MediaEval 2017.

Considering m as the size of the features generated from the network, $\boldsymbol{X}$ is a $n \times m$ matrix that represents the CNN features of all the training samples, $\boldsymbol{Y}$ is a $n \times k$ matrix that represents the actual probabilities for each training sample belonging to each class, γ is the iteration step size, and finally λ is a weight decay term. Each update is given by:

$$\theta = \theta - \gamma \left[\boldsymbol{X}^T \cdot \mathrm{diag}(\boldsymbol{w}) \cdot (\mathrm{h}_\theta(\boldsymbol{x}) - \boldsymbol{Y}) + \lambda \cdot \theta \right] \tag{2.31}$$

The geo-referenced field photo library[8] is a dataset that contains nineteen classes of land cover and it was used by the authors to test the proposed model. For their evaluation, the authors defined the *top-1* accuracy as the percentage of testing samples whose most possible land cover types match their actual types, and *top-3* accuracy as the percentage of testing samples whose actual types were among the most three possible land cover types predicted. For the top-1 accuracy the obtained value was 48.50%, and for the top-3 it was 76.24%.

2.2.3.8 MediaEval Multimedia Satellite Task

MediaEval is a benchmarking initiative dedicated to evaluating new algorithms for multimedia access and retrieval[9]. One of the tasks available in the competition is referred to as Multimedia Satellite Task on Emergency Response for Flooding Events (Bischke et al., 2017b, 2018) and it is very meaningful in the context of this work. More precisely, this benchmark measures the capacity of a system to analyze social media images, complemented with satellite imagery, to map flooding events.

[8] `http://www.eomf.ou.edu/photos/`
[9] `http://www.multimediaeval.org/about/`

Team	Run 1	Run 2	Run 3	Run 4	Run 5
MultiBrasil	**87**/**82**	**86**/80	**88**/**84**	78/49	87/**84**
WISC	80/**83**	81/77	–	–	–
CERTH-ITI	75/56	–	–	–	–
BMC	37/40	37/40	37/40	–	–
UTAOS	82/73	80/70	83/74	83/74	81/73
DFKI	73/69	84/70	84/74	–	–

Table 2.2: Results (expressed using the IoU) for the segmentation task of the Multimedia Satellite Task at MediaEval 2017. In the left are the results corresponding to segmentations in the same location, in the right the results over new locations.

Specifically, the task presented in 2017 shows several similarities with the tasks that I propose to address in my work. This 2017 task was composed of two sub-tasks. The first, referred to as Disaster Image Retrieval from Social Media (DIRSM), had the goal of retrieving all images, in a given dataset, showing evidence of a flood. The main challenge of this task lies in the proper discrimination of water levels in different areas (e.g., images showing a lake versus showing a flooded street). Whilst the second, named Flood Detection in Satellite Images (FDSI), had the goal of developing a method that is able of identify flooded regions from satellite imagery containing information from RGB and near-infrared spectral bands.

Regarding the DIRSM task the participants, in addition to the provided images had available metadata (e.g., nickname of the user, the date, etc.) about those same images. Five runs were allowed in the competition, being that, in the first only the imagery provided could be used; in the second only the metadata about those images; in the third a fusion between the metadata and the visual information; and in the last two everything was allowed, including the usage of external sources to enhance the training of the models. Table 2.1 provides the results obtained by the different teams that took part in the competition. In what concerns the FDSI task, there were again five runs, evaluated on two different datasets: one containing patches from images in the same location of the training data; and another one with new locations. In the first three runs only the provided images could be used for training; in the last two runs, external sources were also allowed. Table 2.2 presents the results obtained in the segmentation task.

2.2.4 Overview

This section presented previous work related to my M.Sc. thesis, presenting: (i) neural models for image classification, (ii) neural models for image (e.g., satellite imagery) segmentation, and (iii) studies related to using ground-level photos for mapping applications.

Model	Authors	Datasets		
		CIFAR-10	**CIFAR-100**	**ImageNet**
EfficientNet-B0	Tan and Le (2019)	1.90	11.9	23.7/6.8
EfficientNet-B1		–	–	21.2/5.6
EfficientNet-B2		–	–	20.2/5.1
EfficientNet-B3		–	–	18.9/4.5
EfficientNet-B4		–	–	17.4/3.7
EfficientNet-B5		–	–	16.7/3.3
EfficientNet-B6		–	–	16.0/3.1
EfficientNet-B7		1.10	8.30	15.6/2.9
DenseNet-121	Huang et al. (2017)	–	–	25.0/7.7
DenseNet-169		–	–	23.8/6.9
DenseNet-201		–	–	22.6/6.3
DenseNet-264		–	–	22.2/6.1
GoogLeNet	Szegedy et al. (2015)	–	–	–/6.67

Table 2.3: Error obtained by the models surveyed in Section 2.2.1. Over ImageNet, the results are expressed at top-1/top-5 error.

Model	Authors	Vaihingen			Potsdam		
		F1	**Acc**	**IoU**	**F1**	**Acc**	**IoU**
FuseNet	Audebert et al. (2018)	90.1	90.8	–	–	–	–
V-FuseNet		90.3	91.1	–	90.6	–	–
RiFCN	Mou and Zhu (2018)	–	–	–	86.1	86.6	–
SNFCN	Chen et al. (2018a)	–	87.7	61.3	–	–	–
SDFCN		–	87.8	62.4	–	86.9	70.4
HSN	Liu et al. (2017)	82.9	84.9	–	72.9	86.6	–
HSN + OI		83.5	85.4	–	73.3	86.9	–
HSN + OI + WBP		83.4	85.4	–	73.4	87.1	–

Table 2.4: Results obtained by the comparable models presented in Section 2.2.2. When crop was applied, the presented result corresponds to the best. Accuracy was computed as the ratio between the number of pixels with the correct predicted label over the total number of pixels.

Table 2.3 summarizes the methods presented in Section 2.2.1. As a starting point in my work, I used the DenseNet-201 (i.e., the largest version whose weights, resulting from a pre-training over the ImageNet dataset, are publicly available) and the EfficientNet-B3 (i.e., the biggest version that fits on the available hardware). All the proposed adaptations considered to those models are presented in Section 3.2.

In turn, Table 2.4 reviews some of the models from Section 2.2.2 (i.e., those that were evaluated on common satellite imagery data). As a starting point, I used the U-Net architecture described in Section 2.1.3. In this case, a complete W-Net architecture like the one surveyed in Section 2.2.2.2, was not considered since the used dataset contains a small number of training instances, making it unfeasible to use such a deep network. The set of modifications introduced to the vanilla U-Net proposed by Ronneberger et al. (2015) are detailed in Section 4.1.

3 Flood Severity Estimation from Social Media Photos

This chapter presents the details of the considered neural approaches leveraged to classify ground-level images regarding the presence of a flooding event, as well as regarding the severity of that same flood. First, Section 3.1 describes the collections of photos used to support the evaluation experiments, detailing the process of extending the ground-truth annotations, and presenting a statistical characterization of the resulting data. Next, Section 3.2 presents the deep learning methods that were considered for image classification, specifically detailing the model adaptations, and the considered training strategy. Section 3.3 presents the considered evaluation metrics and all the obtained results, as well as a comparison with other approaches. Finally, Section 3.4 overviews the contents of this chapter.

3.1 A Dataset for Flood Severity Estimation

The photos that were used to support the experiments reported in this chapter were originally made available as part of different datasets for evaluating computer vision and information retrieval experiments in tasks related to processing flood-related imagery.

The first of three different datasets that were considered in my work consists of 6,600 Flickr images extracted from the Yahoo Flickr Creative Commons 100 Million (YFCC100m) dataset, originally shared under Creative Commons licenses, and made available in the context of the Disaster Image Retrieval from Social Media (DIRSM) sub-task of the MediaEval 2017 Multimedia Satellite Task (Bischke et al., 2017b). Only one image per user was considered in the creation of this dataset, to avoid a bias towards content from the same locations, and from the most active content-sharing users. Relevance scores were collected from two annotators and final ground-truth labels, for whether the photos depict flood related information or not, were determined through the agreement of both annotators in classifying the images with a high confidence. The dataset was originally separated with a ratio of 80/20 into training and testing splits. A total of 1,077 images in this dataset, which contain evidence of a flooding event, are also geo-referenced with latitude and longitude coordinates.

The second set of images came from the European Flood 2013 dataset[1], originally developed in the context of interactive content-based image retrieval experiments at the Computer Vision Group of the University of Jena. The majority of the images in this second dataset relate to the central European floods that occurred in May and June 2013, and have been fetched in July 2017 from the Wikimedia Commons category named *Central Europe floods, May/June 2013*, or from its sub-categories. All the images in the dataset were annotated by hydrologists regarding their relevance in terms of (i) depicting a flooding event, or (ii) containing visual cues (e.g., traffic signs) that can be used to derive an estimation of the inundation depth from the image. After manual inspection, I noticed that some images annotated with the indication of depicting a flooding event were too similar. To avoid any sort of bias, in my experiments, I decided to use only different images from the subset that contained visual clues to infer the water depth. From the complete set of 3,710 images in the European Floods 2013 dataset, only 1,876 images were used in my tests. A total of 883 of these images are geo-referenced into latitude and longitude coordinates, and a total of 224 images contain bounding boxes for the regions depicting objects that can be used to infer the flood-water depth.

To further increase the set of images used in my tests, and given the similarity between the tasks of different editions of the MediaEval Satellite Task, I also used images from the Flood Classification for Social Multimedia sub-task of MediaEval 2018 (Bischke et al., 2018).

The goal of the competition was to develop an algorithm that, given a set of images from social media, (i) retrieves all the images that provide evidence for road passability, and (ii) discriminates between images showing passable versus non passable roads. The original dataset for this task consisted of 11,070 identifiers that point to Twitter messages with images containing in the description the tags *flooding*, *flood* and/or *floods*. The majority of the images have been collected during three big hurricane events in 2017 and were annotated according to two dimensions: (i) one binary label for the evidence of road passability, and (ii) for those images that are labeled as showing evidence, a second binary label for the actual road passability classification. The images were labelled by human annotators in a crowdsourcing setup on the platform named Figure Eight[2]. In order to use the images in the experiments reported in this dissertation, I manually re-labeled them regarding the containment of direct evidence of a flooding event. During this process, some near-duplicates or images with too poor resolution were discarded, resulting in a total of 4,212 images.

[1] `http://www.inf-cv.uni-jena.de/Research/Datasets/European+Flood+2013.html`
[2] `http://www.figure-eight.com`

(a) None (b) Below 1 Meter (c) Above 1 Meter

Figure 3.1: Samples from the dataset containing images labeled according the flood severity.

To support the realization of experiments concerned with a more thin-grained characterization of the image, the ground-truth associated to all the photos were extended with annotations in order to discriminate between 3 distinct flood severity classes (i.e., not flooded, water level below 1 meter, and water level above 1 meter). All those images were manually classified using a Graphical User Interface (GUI), developed in Python programming language, specifically for this task[3]. A total of 1,954 images could not be annotated according to one of the aforementioned classes due to (i) a lack of architectural features that could be used to estimate the water depth in an urban area, or (ii) some ambiguities in the height of the objects contained in the images (e.g., structures like bridges with pillars almost entirely submerged, but without in-depth knowledge about the architectural infrastructures being depicted, it was impossible to estimate the water depth). In the case of multiple submerged objects that led to assume that the flood had different depths, the object closest to the point where the photograph was taken was used as the tiebreaker. In total, the re-annotation process resulted in a collection of 10,734 photos.

Figure 3.1 shows example images, together with the corresponding class assignments. In turn, Figure 3.2, presents a statistical characterization of the resulting dataset of 10,734 images, separately counting the number of images in each class, and also counting images according to the different provenance sources (i.e., the training and testing splits from MediaEval 2017 and MediaEval 2018, and the European Flood 2013 dataset).

[3] `http://www.github.com/jorgemspereira/Flood-Image-Tagger`

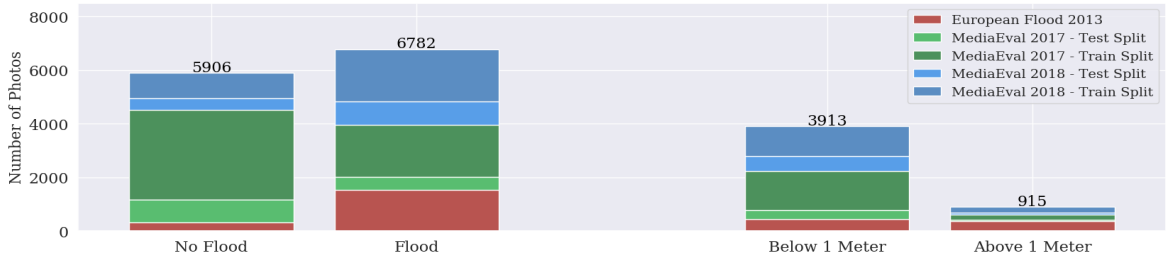


Figure 3.2: Statistical characterization for the resulting dataset.

It is interesting to notice that the per-class distribution of the number of images is somewhat skewed, with relatively few photos within the class for severe flooding events. The per-class distribution is also significantly different between the datasets.

In an attempt to go beyond the considered three flood severity classes, and envisioning even more thin-grained estimates for the water depth, I also used an automated procedure based on heuristics, relying on a Digital terrain Elevation Model (DEM), to assign the geo-referenced photos into a numeric value corresponding to an estimate for the water depth, in meters.

Since I had a substantial number of images manually classified regarding the severity of the depicted flood (i.e., less than 1 meter of water, and above 1 meter), I tried to leverage this information to help in getting a more refined estimate for the real water depth. I argue that this information, combined with the precise location for where the photo was taken plus the height of the terrain in that area, can be used to reasonably estimate the water level. Specifically, considering diff_heights as the set of absolute differences between the terrain elevation in a certain neighbour and the considered point (i.e., diff_heights = $[|\text{height\_nbr\_1} - \text{height\_loc}|, \ldots, |\text{height\_nbr\_x} - \text{height\_loc}|]$), an approximation for the flood-water depth can be obtained using the following equation:

$$\text{depth} = \max(\text{diff\_heights}) - \text{avg}(\text{diff\_heights}) \tag{3.1}$$

The previous idea can be further combined with some heuristics: (i) an image annotated with the class corresponding to less than 1 meter of water illustrates a small flood, so the flooded area can be given by a small surrounding area (e.g., represented by the 8 immediate neighbours of the pixel corresponding to the coordinates of the photo, in a raster representation for the terrain elevation), and (ii) an image annotated with the class corresponding to more than 1 meter illustrates a larger flood, so the flooded area should be larger as well, and thus a bigger neighborhood is needed. In my case, for images depicting severe floods, I started with a neighborhood of 24 pixels, and incrementally increased that neighborhood until a depth above

(a) Estimated Depth: 0.1689m (b) Estimated Depth: 3.4915m (c) Estimated Depth: 0.1315m

(d) Estimated Depth: 1.2157m (e) Estimated Depth: 1.3001m (f) Estimated Depth: 0.9328m

Figure 3.3: Examples of the water depth estimated using the flood severity estimation algorithm.

1 meter is obtained, or until a maximum of 168 neighbors are considered. If the depth obtained when 168 neighbors are considered is not above 1 meter, the depth for that image was set to 1 meter. Some examples for the obtained flood-water depth are presented in Figure 3.3, while Figure 3.4 presents box plots illustrating the distribution for the estimated water depth values, separately in the case of geo-referenced photos associated to the class corresponding to a water depth of less than 1 meter (i.e., a total of 1,016 photos), and in the case of geo-referenced photos depicting a flood with a water depth above 1 meter (i.e., a total of 221 photos).

To support the automatic assignment of the water depth, I mostly used the ALOS World 3D-30m digital elevation model provided by the Japan Aero-space Exploration Agency (JAXA) (Tadono et al., 2014). This high resolution map contains the elevation, in meters, over a raster grid with a resolution of 30 meters per pixel. The JAXA DEM has an estimated root mean squared error of 1.78 meters on the vertical axis, and it has been extensively used in a wide variety of applications (e.g., damage prediction for natural disasters, or water resource investigation).

However, the JAXA DEM contained some relevant points without data values (i.e., points corresponding to the location of some photographs). To fill these points I used two other DEMs, namely the EU-DEM (Mouratidis and Ampatzidis, 2019), and the ASTER Global Digital Elevation Map from NASA (Reuter H.I., 2007). Before merging the three DEMs, a cubic spline interpolation was applied to match the corresponding resolutions to those from the model with the highest resolution (i.e., the 25m EU-DEM model with an European coverage).

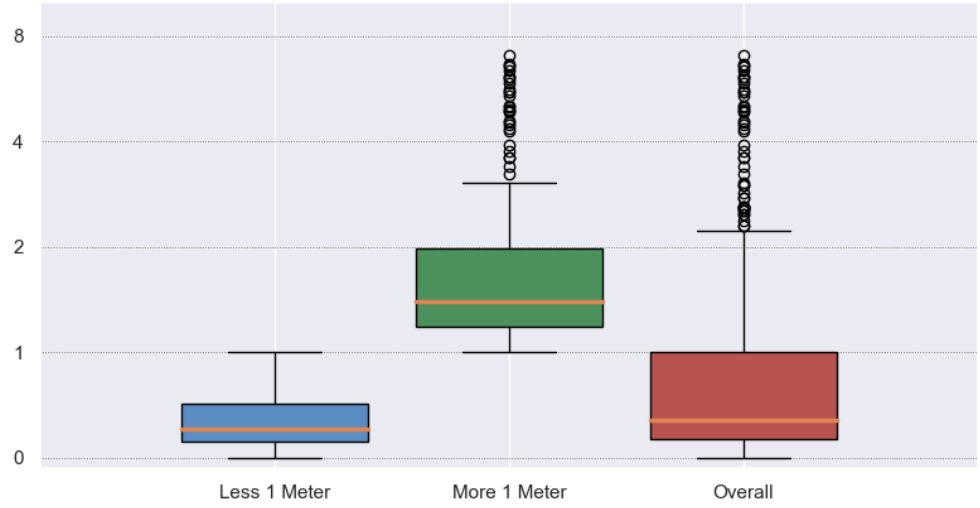


Figure 3.4: Distribution for the estimated water depth values, separately for each of the two flood severity classes, and also in the complete set of images.

After the three DEMs are converted to the same resolution, the pixels without data from the JAXA DEM were filled with the values from the EU-DEM for locations within Europe, and filled with values from NASA's DEM otherwise. Finally, the complete DEM was interpolated to a 10 meter resolution using cubic spline interpolation. Notice that, when leveraging the resulting raster, a neighbourhood of 8 pixels in the raster representation of the DEM corresponds to a region of 30×30 meters, while a neighbourhood of 24 pixels corresponds to 50×50 meters, and a neighbourhood of 168 pixels corresponds to a region of 130×130 meters.

A manual examination of the results, obtained with the heuristic procedure outlined in this section, and which are exemplified in Figure 3.3, indicates that the resulting values often correspond to seemingly correct assessments. However, future developments can perhaps consider alternative and more accurate procedures for estimating the water depth, e.g. combining information in high resolution DEMs with the limits of inundated areas estimated from remote sensing products, referring to the same calendar dates and locations, as the ground-level photos (i.e., perhaps more detailed boundaries for the inundated areas depicted in the photos can be used, instead of assuming an heuristic region surrounding the geo-spatial coordinates associated to the photos). In Section 4.1, I address the semantic segmentation of flooded areas in satellite imagery, proposing a novel fully-convolutional neural network (i.e., based on the U-Net), that could be used to support this task.

3.2 The Considered Image Classification Methods

As previously reported, I used the DenseNet (Huang et al., 2017) and the EfficientNet (Tan and Le, 2019) neural architectures – explained in Section 2.2.1 – to leverage the datasets created to support the tasks of (i) discriminating between images with flood evidence, (ii) classifying images into the three aforementioned flood-related classes, and (iii) outputting the precise water depth in images with flooding events. Besides these two models, I also adapted an Attention Guided Convolutional Neural Network (AG-CNN), originally proposed by Guan et al. (2018). This section starts by detailing the AG-CNN model, whereas Section 3.2.2 presents the model adaptations that were considered in all the three analyzed models in order to predict a thin-grained value for the water depth (i.e., instead of a binary or a 3-way classification). Finally, Section 3.2.3 presents all the considered model training strategies and hyper-parameter tuning.

3.2.1 An Attention Guided Two-Step Approach

The evidence for a particular class (e.g., a particular flood severity assessment) is usually only present in a small and localized area of an input image. Taking inspiration on recent work focusing on medical images, I also experimented with an adaptation of the Attention Guided Convolutional Neural Network (AG-CNN) architecture proposed by Guan et al. (2018). The AG-CNN approach consists of a model with two branches: (i) a global branch (i.e., a model that processes the entire image), and (ii) a local branch (i.e., a model that only processes a portion of the image), that are lately fused for informing the classification decision. As the backbone for each one of the branches, I selected the aforementioned DenseNet architecture (i.e., the EfficientNet model was not selected due to hardware constraints).

The AG-CNN involves a global branch that processes the entire image, from where an attention map is generated. The attention map is generated through an approach proposed by Selvaraju et al. (2016), and the values from the attention map are used as a mask to crop a discriminative region from the source image. More formally, in order to obtain the attention map of width w and height h for a class c, I calculated the gradient of the neuron in the last layer corresponding to the activation value y^c (i.e., the neuron that corresponds to the prediction of class c by the considered model for a certain input image, or simply the last neuron in the case of a binary model) with respect to the feature maps f^l obtained by the last convolutional layer l with a resolution equal to that of the input image. Considering z as the number of pixels in the feature map, the gradients flowing back are global average-pooled to obtain the neuron importance weights α_l^c for each pixel, as demonstrated in the following equation:

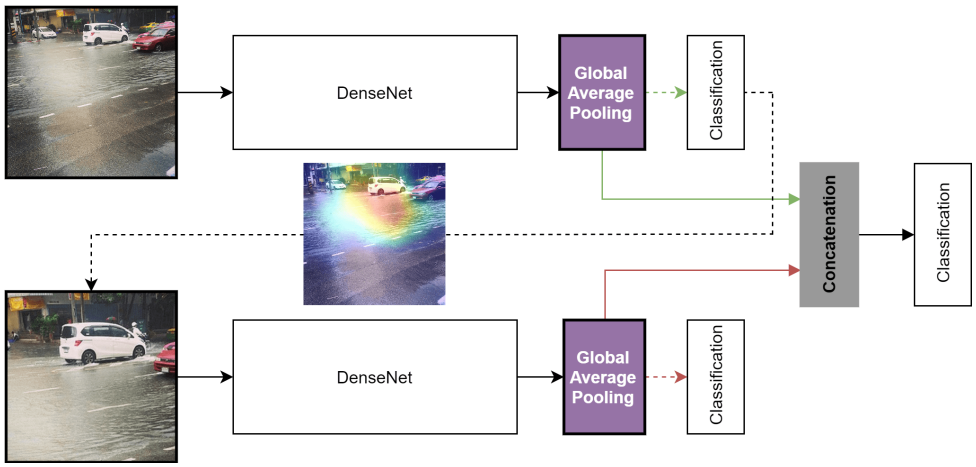


Figure 3.5: Graphical representation for the Attention Guided DenseNet architecture that was used on my experiments.

$$\alpha_l^c = \frac{1}{z} \sum_i \sum_j \frac{\partial y^c}{\partial f_{ij}^l} \tag{3.2}$$

These per-pixel weights represent a partial linearization of the deep network downstream from f, and capture the importance of a feature map l for a target class c. Next, a weighted combination of forward activation maps with a ReLU operation is applied, obtaining the attention map for the corresponding input image:

$$\text{attention\_map} = \text{ReLU}\left(\sum_l \alpha_l^c f^l\right) \tag{3.3}$$

Since some of the images from the European Floods 2013 dataset have bounding boxes that correspond to objects from which it is possible to infer the water depth, I used those bounding boxes to calculate an optimal threshold by which to crop the attention maps. I specifically calculated the Jaccard coefficient between the ground-truth bounding boxes and the result of cropping the attention map by a certain threshold. The final value selected for the threshold was the one that maximizes the average Jaccard coefficient computed from all the considered images. Then, this threshold was applied to crop all the images. Figure 3.6 presents the obtained bounding boxes, together with the manually annotated bounding boxes, attesting the quality of the attention maps generated by the DenseNet. After cropping all the images, the second (local) branch of the AG-CNN is trained.

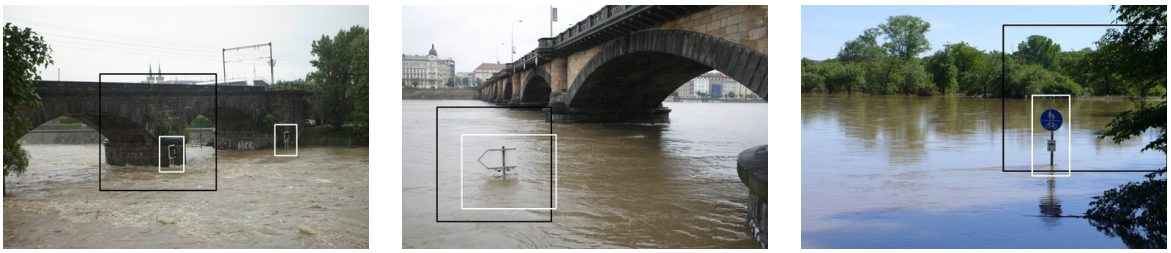


Figure 3.6: Bounding boxes manually annotated, together with the bounding boxes obtained after using a thresholded attention map, in white and black, respectively.

After the weights for the global and the local branches were pre-trained (i.e., both branches were trained separately on the original and the cropped images, afterwards setting those weights as fixed) the results of the last global average pooling layers from both models are concatenated to form a new classification layer, as demonstrated in Figure 3.5. Between the concatenation and the final classification layer, dropout regularization with a drop rate of 50% was also used. Finally, the last layers of the AG-CNN model were fine-tuned, receiving as input both the original and the corresponding cropped images.

3.2.2 Computing Fine-Grained Predictions for the Water Level

In order to estimate the precise water depth, I adapted the DenseNet, the EfficientNet, and the AG-DenseNet models, using the scheme depicted in Figure 3.7. In complement to an output layer with 3 neurons, equal to the last layer in the model used in the experiments for discriminating between 3 flood severity classes, and by which I initialized the weights of this model, I added 2 more layers with 1 neuron each. The activation function of these two layers was set to a thresholded linear function, with the first only taking values between zero and one, and the second only taking values larger than one.

The two aforementioned layers with thresholded linear activations were then connected to a decision layer that starts by calculating a soft-argmax (i.e., a differentiable variation of the argmax function) for the values of the first classification layer, obtaining the predicted flood severity class (i.e., no flood, flood less than 1 meter, and flood with more than 1 meter). The soft-argmax function, considering p as the probability vector output by a standard softmax layer and β as a hyper-parameter that controls the temperature of the resulting probability map, can be defined as show in Equation 3.4, outputing a value between 0 and 2 that, when thresholded, can encode the predicted flood severity class.

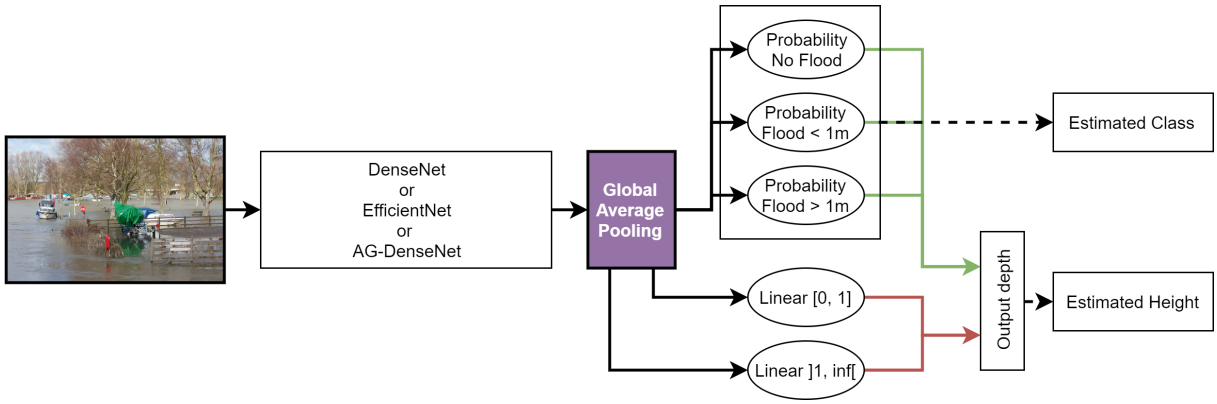


Figure 3.7: Model adaptations used for estimating the exact water depth.

$$\text{soft-argmax}(p) = \sum_{i=0}^{2} \text{softmax}(p \times \beta) \times i = \sum_{i=0}^{2} \frac{e^{p[i]\times\beta}}{\sum_{k=0}^{2} e^{p[k]\times\beta}} \times i \tag{3.4}$$

Notice that by setting β to a large value, the previous expression will produce results that approximate the integer values of 0, 1, or 2. Taking into account the result of Equation 3.4, the classifier then decide as follows: (i) if the result is less than 0.5 (i.e., the model classifies the photo as not containing a flooding event) outputs zero for the water depth, (ii) if the result is less than 1.5 (i.e., the model classifies the photo as containing a flood with less than 1 meter), the output is the result of the neuron with activation function between zero and one, and otherwise (iii) the model classifies the photo as containing a flood with more than 1 meter, and thus outputs the result of the neuron with a activation function that outputs a value larger than one. The final model thus leverages two outputs with separate loss functions, simultaneously trying to classify images into the three considered flood severity classes, and to estimate the precise water depth for the depicted flood.

3.2.3 Training Strategy and Hyper-Parameter Tuning

My experiments with DenseNet and AG-DenseNet models used an implementation for a 201-layer DenseNet model with $\theta = 0.5$, pre-trained on the ImageNet dataset and provided as part of the Keras[4] deep learning library. I also used a pre-existing Keras implementation for the EfficientNet model[5], pre-trained also on ImageNet. Due to GPU memory constraints, and in order to obtain directly comparable results (i.e., without altering the considered batch size on my experiments), I used the EfficientNet-B3 since this was the biggest version that could fit on the available hardware.

[4] `http://www.keras.io/applications/#densenet`
[5] `http://www.github.com/titu1994/keras-efficientnets`

For fine-tuning the pre-trained models with the described flood-related datasets, I considered a selection of hyper-parameters and model training strategies that relied on the guidelines from previous publications, as for instance the guidelines discussed by Xie et al. (2018). An initial set of tests was used to validate the ideas presented in this section, and all the developed code, as well as the datasets that were created, are available online[6].

The last layer of the pre-trained DenseNet or EfficientNet models was replaced by a new fully-connected layer, with a number of nodes compatible with the classification task. In the case of the AG-DenseNet approach, the last layer of each branch was similarly replaced. In all the cases, the entire set of network weights was afterwards fine-tuned with the flood-related images, doing this fine-tuning in three stages for the case of the AG-DenseNet model, as discussed on Section 3.2.1. When considering a binary classification, the last layer consists of a single node with a sigmoid activation function, and the training involves minimizing a binary cross-entropy loss. When estimating flood severity classes, the last layer consists of three output nodes, and training involves a softmax activation function, together with the categorical cross-entropy loss function. When predicting a thin-grained value for the water depth, the models were first pre-trained with the images labeled according to the three flood severity classes, and then the model output was replaced before re-training the network according to the strategy outlined in Section 3.2.2. The final model training consisted on minimizing the sum between the categorical cross-entropy and the mean squared error.

Training relied on the Adam (Kingma and Ba, 2014) optimization algorithm together with a Cyclical Learning Rate (CLR), as described by Smith (2017). In more detail, the learning rate varied between 10^{-5} and 10^{-4}, according to a triangular policy that decreases the cycle amplitude by half after each period (i.e., annealing the learning rate), while keeping the base learning rate constant. I used mini-batches of 16 images, created through a generator that considered simple real-time data augmentation procedures (i.e., randomly flipping the input images horizontally, and/or randomly shifting the brightness by a factor between 0.8 and 1.2).

Training proceeded for up to a maximum of 50 epochs. However, a small validation set (i.e., 10% of each training split in cross-validation experiments, and 20% of all the available training data when using fixed splits) was used to define an early stopping criterion. Training stopped if the validation loss did not decrease for 5 consecutive epochs. The final model weights were taken from the training epoch with the smallest value for the validation loss.

[6] `http://www.github.com/jorgemspereira/Classifying-Geo-Referenced-Photos`

3.3 Evaluation Metrics and Experimental Results

For evaluate the obtained models, I used the standard classification metrics (i.e., accuracy, precision, recall, and F1-Score) together with the Average Precision at k (AP@k), considering a cutoff value of $k = 480$, and also an average value across multiple cutoffs (i.e., 50, 100, 250, and 480). This metric was the official metric of the MediaEval 2017 competition – see Table 2.1 for all the results obtained in the competition – and measures the number of relevant images among the top k retrieved results, thus taking the rank into consideration when sorting photos according to the confidence of the classifier in assigning the positive (i.e., flood-related) class. More formally, the AP@k can be defined using the following equation:

$$\text{AP@k} = \frac{\sum_{i=1}^{k} = \frac{\#\text{relevant images retrieved in the first } i\text{-th places}}{i}}{k} \tag{3.5}$$

The mathematical description of both precision and recall is presented bellow, where t_p represents the number of true positives (i.e., flood photos that were identified as such), f_p the number of false positives (i.e., incorrectly labeled flood photos) and finally f_n the number of false negatives (i.e., flood photos that were not identified):

$$\text{Precision} = \frac{t_p}{t_p + f_p} \tag{3.6}$$

$$\text{Recall} = \frac{t_p}{t_p + f_n} \tag{3.7}$$

To have a single numerical value that well represents the accuracy of the classifier, the harmonic mean between precision and recall was also used. This value is usually referred to as the F1-Score, and can be defined as follows:

$$\text{F1-Score} = 2 \cdot \left(\frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \right) \tag{3.8}$$

In a first set of experiments, I assessed the ability of the DenseNet, EfficientNet, and AG-DenseNet models to detect whether a given photo presents direct evidence of a flooding event, following the general task definition and evaluation methodology of the Disaster Image Retrieval from Social Media (DIRSM) sub-task of the MediaEval 2017 Multimedia Satellite Task.

		Classification				Ranked Retrieval	
	Model	**Pre**	**Rec**	**F1**	**Acc**	**AP@{50,100,250,480}**	**AP@480**
MediaEval 2017 Train Split	DenseNet	92.05	91.66	91.85	94.09	99.49	98.26
	EfficientNet	88.74	**93.54**	91.08	93.33	**99.59**	**98.59**
	AG-DenseNet	**92.65**	91.88	**92.26**	**94.40**	99.01	97.94
MediaEval 2018 Re-annotated	DenseNet	67.52	**98.75**	80.20	82.27	**98.14**	**95.85**
	EfficientNet	**71.63**	97.29	**82.51**	**85.00**	96.65	91.73
	AG-DenseNet	69.28	97.71	81.07	83.41	96.33	95.19
European Floods 2013	DenseNet	59.60	93.13	72.68	74.55	89.66	83.28
	EfficientNet	56.19	**96.46**	71.01	71.36	85.60	79.76
	AG-DenseNet	**61.86**	91.25	**73.74**	**76.36**	**92.21**	**85.72**
All Photos	DenseNet	86.58	**95.42**	90.79	92.95	99.34	97.97
	EfficientNet	**90.02**	92.09	91.04	93.41	**99.52**	**98.30**
	AG-DenseNet	89.78	93.33	**91.52**	**93.71**	99.46	98.11
CV MediaEval 2017 Train Split	DenseNet	88.60	**89.26**	88.89	91.90	**99.78**	**99.32**
	EfficientNet	**90.56**	89.10	**89.80**	**92.64**	97.98	98.50
	AG-DenseNet	89.75	88.90	89.27	92.25	99.59	98.98
CV MediaEval 2018 Re-annotated	DenseNet	96.40	97.10	96.75	95.61	99.98	99.90
	EfficientNet	**98.06**	97.56	**97.81**	**97.06**	99.98	99.88
	AG-DenseNet	95.96	**97.67**	96.80	95.65	**100.0**	**100.0**
CV European Floods 2013	DenseNet	96.73	96.58	96.63	94.46	**100.0**	**99.99**
	EfficientNet	**98.26**	97.22	**97.73**	**96.27**	**100.0**	**99.99**
	AG-DenseNet	97.44	**97.29**	97.35	95.63	99.98	99.92
CV All Photos	DenseNet	94.69	**94.41**	94.54	93.96	99.98	99.99
	EfficientNet	**95.87**	94.25	**95.05**	**94.56**	**100.0**	**100.0**
	AG-DenseNet	94.97	94.35	94.65	94.10	**100.0**	**100.0**
Best MediaEval 2017	—	—	—	—	—	95.73	87.82

Table 3.1: Results on the task of discriminating photos showing an evidence of a flooding event, according to metrics for classification and for ranked retrieval.

Table 3.1 presents the obtained results for the different models, considering the aforementioned evaluation metrics, and trained through the complete procedure described in the previous section (i.e., with hyper-parameters tuned to the best values). The table features three main sets of rows, corresponding to (i) experiments in which results were evaluated on the official test split from the MediaEval 2017 competition, (ii) experiments based on 10-fold cross-validation, and (iii) the best results that were reported at MediaEval 2017 when using visual features alone. The first four rows specifically correspond to using different sets of photos for model training, namely (i) all the photos in the official training split from MediaEval 2017, (ii) all the photos re-annotated from MediaEval 2018, (iii) all the photos in the European Floods 2013 dataset which contain objects that allow assessing the height of the water, and (iv) a combination of the photos from the three previous items. The next four rows correspond to cross-validation experiments using (i) all photos from the training split of MediaEval 2017, (ii) all the photos re-annotated from MediaEval 2018, (iii) the same set of photos from European Floods 2013 dataset as in the previous set of rows, and (iv) a combination of the photos from MediaEval 2017 (i.e., train and test splits), MediaEval 2018, and from the European Floods 2013 dataset. Finally, the last row corresponds to the best results reported by the participants in MediaEval 2017, specifically in terms of the AP@480 metric, and the mean of the average precision at the different cutoffs (i.e., corresponding to the AP at 50, 100, 250, and 480).

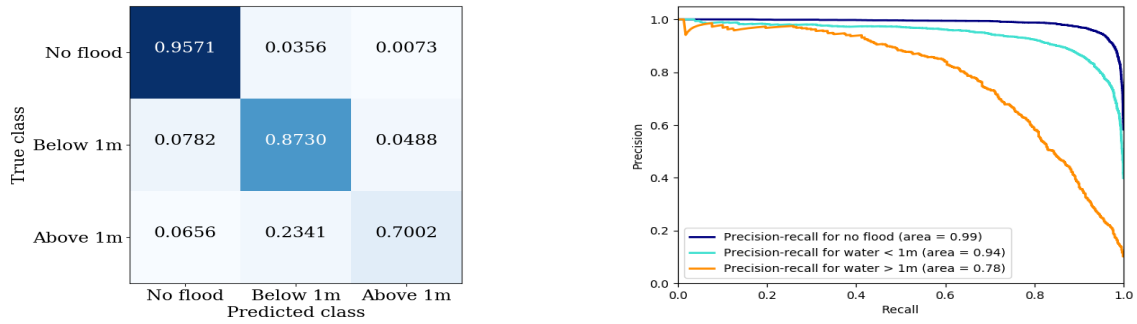
	Model	Acc	MAE	Macro-Averaged		
				Pre	Rec	F1
Complete approach	DenseNet	90.46	0.103	85.16	83.96	84.87
	EfficientNet	90.24	0.107	84.44	84.10	84.19
	AG-DenseNet	**90.67**	**0.102**	**86.01**	**84.71**	**85.25**
- data augmentation	DenseNet	88.97	0.120	83.88	81.75	82.53
	EfficientNet	**90.03**	**0.111**	83.88	**83.88**	**83.80**
	AG-DenseNet	89.02	0.119	**84.42**	81.62	82.74
- cyclical learning rate	DenseNet	88.93	0.122	**83.29**	80.95	82.21
	EfficientNet	**89.39**	**0.118**	83.08	**82.80**	**82.86**
	AG-DenseNet	89.02	0.121	83.18	81.96	82.49
- model pre-training	DenseNet	90.18	0.108	**85.07**	83.87	84.28
	EfficientNet	70.83	0.312	61.98	55.42	55.81
	AG-DenseNet	**90.24**	**0.106**	84.74	**84.40**	**84.45**
Only EU-Floods'13 data	DenseNet	84.17	0.169	84.87	85.65	84.50
	EfficientNet	**84.94**	0.163	85.24	**85.84**	**85.28**
	AG-DenseNet	**84.94**	**0.162**	**85.47**	85.53	84.94

Table 3.2: Results for cross-validation experiments on the task of classifying photographs according to the flood severity classes using different configurations.

The results confirm that all the considered models indeed achieve a very good performance in this particular task, significantly outperforming the official results from the competition. The best results over the test split from the dataset of the MediaEval 2017 competition, in terms of the main classification metrics (i.e., accuracy and F1-Score), were obtained using the AG-DenseNet. However, when considering the ranked retrival metrics used in the MediaEval competition, the best results were instead achieved by the EfficientNet model. In the cross-validation experiments, the EfficientNet model has achieved better results in all metrics, in the majority of the experiments. Nonetheless, all three models achieved very similar results, and future endeavors can perhaps consider experiments with an AG-EfficientNet model, or with a larger EfficientNet (e.g., the B4 variation), in an attempt to further improve the results.

In a second set of experiments, I assessed the ability of the same three neural models to discriminate between the three different flood severity classes. These tests leveraged a 10-fold cross-validation methodology, using the entire set of photos that resulted from the annotation process described in Section 3.1.

The quality of the results was measured in terms of macro-averaged values for the precision, recall, and F1 metrics. Besides these standard metrics for multi-class classification problems, I also measured the overall classification accuracy, and the Mean Absolute Error (MAE). In this last case, note that the different classes can be seen to correspond to ordinal values encoding the flood severity level (i.e., 0, 1 and 2), and thus it is possible to measure the differences between the ground-truth and the estimated values through the MAE.



(a) Confusion Matrix

(b) Precision-Recall Curves

Figure 3.8: Normalized confusion matrix and precision-recall curves for each class, in the complete approach for predicting the flood severity labels.

Table 3.2 presents the obtained results, again attesting to the high quality of the predictions returned by the different models. The 2nd, 3rd and 4th sets of rows, shown on Table 3.2, correspond to ablation tests in which I removed some of the strategies listed in Section 3.2.3 (i.e., not considering data augmentation procedures, random initialization instead of model pre-training with ImageNet, and a fixed learning rate of 10^{-5} instead of the cyclical scheme). The last row corresponds to a cross-validation test in which I only used photos from the European Flood 2013 dataset, and in which the flood-related images are known to contain objects from which the water depth can, perhaps, be derived. In complement to Table 3.2, Figure 3.8 presents (i) a confusion matrix for the classification results of the complete approach over the entire dataset, side-by-side with (ii) a precision-recall curve for each of the classes. The plots in Figure 3.8 show that the majority of the errors relate to confusion between the flood-related classes (i.e., depth above 1 meter classified as below 1 meter), and that the area bellow the precision-recall curve for the class that represents floods with more than 1 meter is the smallest. Moreover, it is interesting to notice that the experiments with the smaller European Floods 2013 dataset produced slightly inferior results, despite the fact that these images should, in principle, be more informative. This result, and also the fact that worse results are available for the class related to floods with water above 1 meter, suggests that the number of images available for model training can indeed impact the result quality.

In a final set of experiments, I evaluated the ability of the aforementioned neural models to predict the precise water depth in flood related images. This test used a smaller set of 1,025 geo-referenced photos for which I calculated the precise water depth (from the training split of the MediaEval 2017 competition as well as from the European Floods 2013 dataset), combined with all the photos that do not contain evidence of a flooding event (i.e., a water depth of zero). The remaining 212 photographs that I classified from the test split of the MediaEval

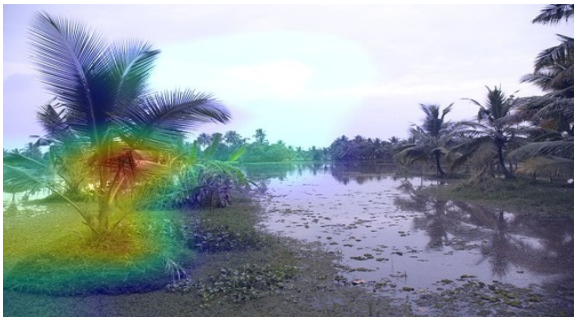
			Accuracy with Threshold			
Model	**MSE**	ρ	0.10	0.25	0.50	1.00
DenseNet	**0.575**	**0.536**	**25.00**	**52.83**	**76.41**	**93.40**
EfficientNet	0.889	0.116	24.58	50.94	75.94	90.56
AG-DenseNet	0.776	0.279	20.28	45.29	67.45	93.39

Table 3.3: Results for the task of classifying photographs according to the precise water depth.

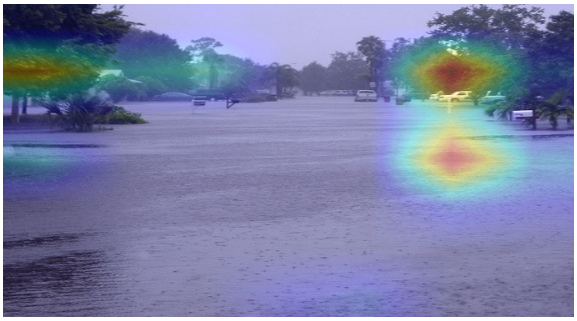
2017 competition were used to evaluate the models. Although this particular experiment used a relatively small set of photos (i.e., images from the entire dataset which were geo-referenced into latitude and longitude coordinates, from which I derived the water depth), it should be noticed that the proposed procedure makes its predictions by using only the visual contents of the images, and is thus not limited to processing geo-referenced photos.

Table 3.3 presents the results obtained in this last test, specifically reporting (i) the Mean Squared Error (MSE) between the water depth estimated by the model and the ground-truth estimate provided by the DEM-based heuristics, (ii) the Pearson correlation ρ between both estimates, and (iii) the accuracy with various thresholds over the predicted depth (i.e., a certain predicted depth is considered correct if it is between the correct value minus the threshold, and the correct value plus the threshold). The best overall results in this task were obtained using the DenseNet model, corresponding to a MSE of approximately 58 cm in the predicted water depth. The DenseNet model also obtained a MSE of 12.75 cm on the subset of photos where the ground-truth water depth is bellow 1 meter, and a MSE of 3.29 meters on the subset of photos depicting water depth above 1 meter. From all the three models that were used in my tests, the DenseNet is the one involving fewer parameters, and thus also the easiest to train on the relatively small set of images with precise information on the water depth.

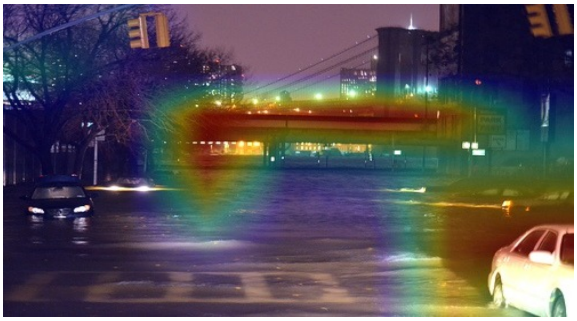
Figure 3.9 presents attention maps obtained for some examples from the MediaEval 2017 test split, resulting from the predictions done by the best performing model (i.e., notice that attention maps derived from the DenseNet model are the same as those used by the AG-DenseNet as explained in Section 3.2.1, given that AG-DenseNet uses a DenseNet on its first branch), as well as the values predicted for the water depth. The examples demonstrate that, in the majority of the cases, not only does the model pay more attention to relevant areas from which it is possible to infer the water depth, as the obtained values are very reasonable and similar to the ones obtained by the proposed heuristic. Images with cars where the wheels are partially submerged are often assigned to a water depth of few centimeters, whereas images with partially submerged trees are instead assigned to a water depth above 1 meter.



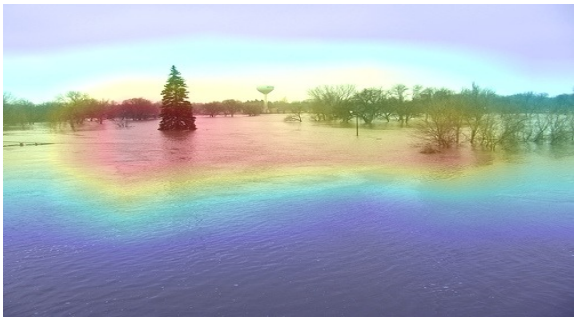
(a) Real/Predicted: 0.25/0.20m



(b) Real/Predicted: 0.26/0.23m



(c) Real/Predicted: 0.66/0.59m



(d) Real/Predicted: 1.42/1.38m

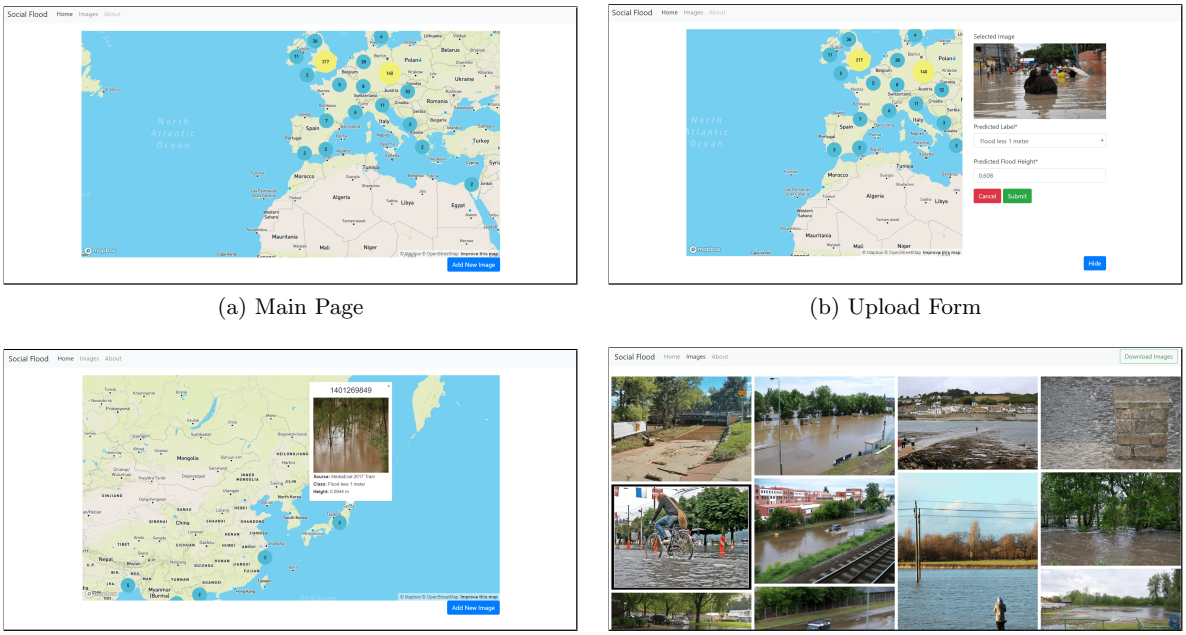
(e) Real/Predicted: 1.54/1.39m

(f) Real/Predicted: 1.85/1.59m

Figure 3.9: Examples illustrating the water depth values predicted with the attention maps.

3.4 Overview

This chapter discussed the use of convolutional neural networks for analyzing ground-level images taken during flooding events, particularly in the tasks of (i) discriminating images showing direct evidence of floods, or estimating the severity of the flooding event either (ii) in terms of three distinct classes, or (iii) by directly estimating water depth. Considering distinct datasets from previous studies (i.e., the European Flood 2013 dataset, and data from the 2017 and 2018 editions of MediaEval competitions focusing on flooding events), I specifically evaluated models based on the DenseNet and EfficientNet neural architectures, concluding that these approaches produce high-quality results, thus motivating the use of crowdsourced photos to complement other sources of information collected through remote sensing (e.g., flood-related information derived from satellite imagery (Malinowski et al., 2015; Cian et al., 2018; Shen et al., 2019)).



(a) Main Page

(b) Upload Form

(c) Image Description

(d) Complete Image Collection

Figure 3.10: Screenshot of the web application created as proof-of-concept for this work.

As demonstrated in the performed tests, the number of available training instances significantly influences the obtained results. This problem occurs mainly in the experiments concerned with predicting the flood severity class, and/or when estimating an approximation for the value of the flood-water depth. An idea that could be used to address this limitation is the creation of some kind of social platform that allows the users to upload flood-related images, envisioning the increase in the number of flood-related images available for future work in the area.

One of the contributions of my work is the development of a proof-of-concept web application that illustrates, on a global map, all the geo-referenced images used in my experiments. Besides the ability to show all the used images, the web application also allows users to upload new images, observe the classification made by the best performing model (i.e., the DenseNet) in terms of the predicted severity class and water depth, and register eventual corrections to the results of the automated method. This web application was developed using a Python web framework named Django[7], and all the source code, together with the weights of the best performing model, are freely available on a GitHub repository[8], thus allowing it to be used as a starting point for larger projects, that could lead not only to have more generalizable models but also in helping emergency responders into coordinate rescue efforts. Some screenshots of the web application are presented in Figure 3.10.

[7] `http://www.djangoproject.com/`
[8] `http://www.github.com/jorgemspereira/Social-Flood`

4 Mapping Flood Extents using Satellite Imagery

This chapter presents the details of the proposed U-Net neural architecture that is able of receiving as input a set of overhead features (e.g., RGB bands, near-infrared, water and vegetation indexes, etc.) and output a precise delimitation of the depicted flood. First, Section 4.1 presents the considered technical innovations applied to the fully-convolutional neural network previously proposed by Ronneberger et al. (2015). Then, Section 4.2 presents the input data sources used in my work. Section 4.3 details the model training strategies, and Section 4.4 presents the evaluation metrics that were considered, and discusses the obtained results, comparing them with other approaches. Finally, Section 4.5 summarises the content of this chapter.

4.1 The Proposed Segmentation Method

One of the most commonly used neural models for image segmentation is the U-Net (Ronneberger et al., 2015), from which I added the following modifications: (i) addition of channel-and-spatial squeeze and excitation blocks, (ii) replacement of the traditional convolutional blocks by densely connected convolutional blocks, and (iii) addition of extra connections between the encoder and the decoder blocks. Notice that the base U-Net architecture leverages five convolutional blocks in both the encoder and the decoder, together with skip-connections connecting symmetrical blocks– consult Section 2.1.3 for the details on the U-Net model, as well as graphical representation for the exact model that served as basis for the proposed modifications.

The frequently used convolutional filters that extract hierarchical information from images apply the same processing to all the channels in the input feature map, although they cannot emphasise informative features and suppress less useful ones, since they should be able to extract whatever is necessary to solve a task efficiently. To address this limitation, Hu et al. (2018) proposed Squeeze and Excitation (SE) networks, introducing a building block for convolutional networks that improves channel inter-dependencies at almost no computational cost. In addition, these building blocks can be easily added to existing models. The idea presented by the authors is to add parameters in each channel of a convolutional block so that the network can adaptively better adjust the weighting of each feature map.

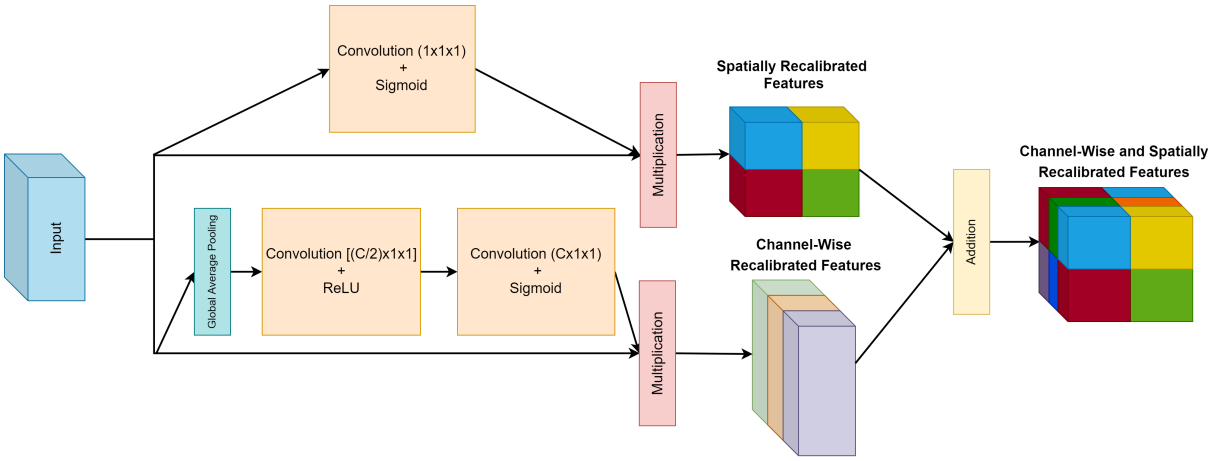


Figure 4.1: Graphical representation for the channel-and-spatial squeeze and excite blocks used on the proposed segmentation model, and which were originally proposed by Roy et al. (2018).

A channel SE block, starts by squeezing the global spatial information into a channel descriptor. This is achieved by using global average pooling to generate channel-wise statistics. Formally, considering an input tensor $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_C]$ as a combination of channels $\mathbf{u}_i \in \mathbb{R}^{H \times W}$, the spatial squeeze produces a vector $\mathbf{z}_k \in \mathbb{R}^{1 \times 1 \times C}$ with its k-th element equal to:

$$\mathbf{z}_k = \mathrm{F}_{sq}(\mathbf{u}_k) = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} \mathbf{u}_k(i, j) \tag{4.1}$$

To make use of the information aggregated in the squeeze operation, the excitation phase aims to fully capture channel-wise dependencies. This operation is capable of learning a non-linear (and non-mutually-exclusive) relationship between the different channels. To achieve this, a single gating mechanism is used. This gating mechanism, considering $\sigma(\cdot)$ as the sigmoid operation, $\gamma(\cdot)$ as the ReLU function, and weights $\mathbf{W}_1 \in \mathbb{R}^{(C/r) \times C}$ and $\mathbf{W}_2 \in \mathbb{R}^{C \times (C/r)}$ associated to two fully-connected layers, can be fully defined as show in Equation 4.2. To limit the model complexity, and to aid in generalization, the gating mechanism is parameterised by forming a bottleneck with two fully-connected layers around the non-linearity (i.e., a dimensionality-reduction layer with reduction ratio r, a ReLU activation, and a dimensionality-increasing layer returning to the channel dimension of the transformation output). The output of the squeeze and excite block is obtained by a channel-wise multiplication between the input of this block and the resulting feature map.

$$\mathbf{s} = \mathrm{F}_{ex}(\mathbf{z}, \mathbf{W}) = \sigma(g(\mathbf{z}, \mathbf{W})) = \sigma(\mathbf{W}_2 \cdot \delta(\mathbf{W}_1 \mathbf{z})) \tag{4.2}$$

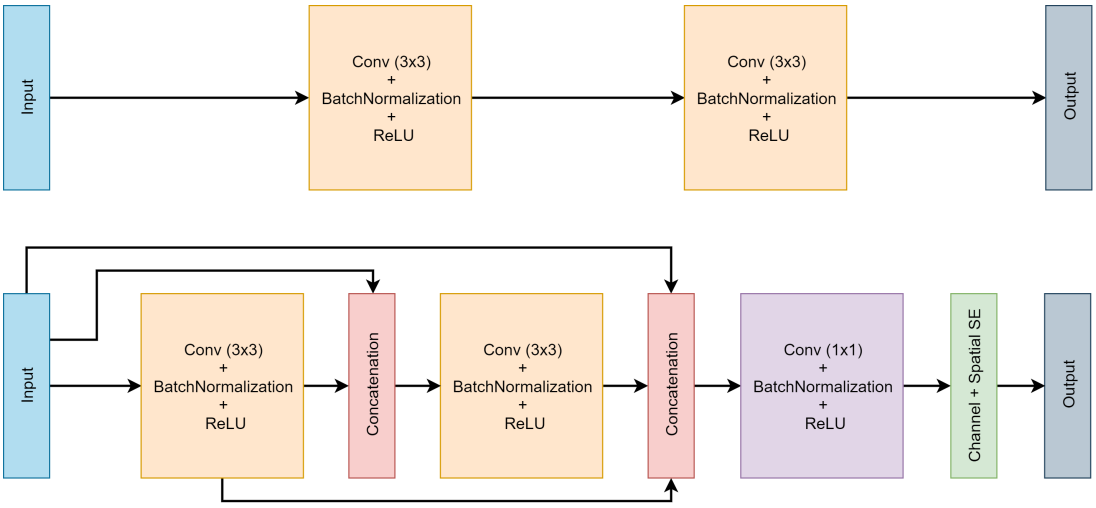


Figure 4.2: Comparison between the convolutional blocks found in standard U-Net architectures (on top) and the densely connected blocks introduced by Dong et al. (2019) that were used in my model (at the bottom). Notice that I also extended the blocks from Dong et al. (2019) with a final channel-and-spatial squeeze and excitation block, as detailed by Roy et al. (2018).

A recent work by Roy et al. (2018) proposed update the channel SE blocks with a spatial SE operation, forming a concurrent channel-and-spatial squeeze and excite block. Instead of squeezing the input feature map along the channels as the channel SE blocks, the spatial SE block slices the input tensor $\mathbf{U} = [\mathbf{u}^{1,1}, \mathbf{u}^{1,2}, \mathbf{u}^{i,j}, ..., \mathbf{u}^{H,W}]$ with $\mathbf{u}^{i,j} \in \mathbb{R}^{1\times1\times C}$ corresponding to the spatial location (i, j), specifically with $i = \{1, 2, ..., H\}$ and $j = \{1, 2, ..., W\}$.

The spatial squeeze operation is achieved through a convolution $\mathbf{Q} = \mathbf{W}_{sq} \star \mathbf{U}$ with weight $\mathbf{W}_{sq} \in \mathbb{R}^{1\times1\times C\times1}$, generating a projection tensor $\mathbf{Q} \in \mathbb{R}^{H\times W}$. Each $\mathbf{Q}_{i,j}$ of the projection represents the linearly combined representation for all the channels C for a spatial location (i, j). This projection is then passed through a sigmoid layer to re-scale activations to $[0, 1]$, which is used to recalibrate or excite $\mathbf{U}$ spatially, as formally described by the equation bellow:

$$\mathrm{F}_{sSE}(\mathbf{U}) = [\sigma(\mathbf{Q}_{1,1})\mathbf{u}^{1,1}, ..., \sigma(\mathbf{Q}_{i,j})\mathbf{u}^{i,j}, ..., \sigma(\mathbf{Q}_{H,W})\mathbf{u}^{H,W}] \tag{4.3}$$

Each value $\sigma(\mathbf{Q}_{i,j})$ corresponds to the relative importance of the spatial information (i, j) of a given feature map. A graphical representation for the resulting combination of the two SE blocks described above is presented in Figure 4.1.

As noted in the beginning, I also replaced the typical convolutional blocks used in U-Net architectures for densely connected blocks as presented by Dong et al. (2019). The intuition behind these blocks is that each convolution in one block has directly access not just to the output of the previous convolution, but also to the input, giving more information to learn

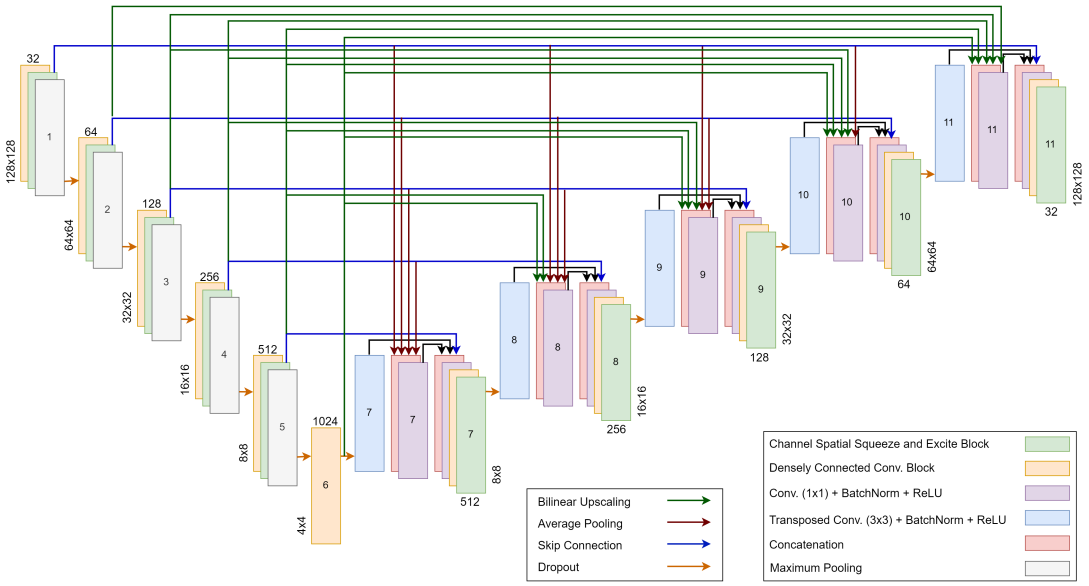


Figure 4.3: Graphical representation for the considered U-Net neural network architecture.

important features. A densely connected block starts by applying a 3×3 convolution, followed by a batch normalization operation, which is then followed by the ReLU non-linear activation function. The output of the activation function is then concatenated with the input of the block, and the same set of operations is then applied. The results from the first activation function, and the results from the second, are concatenated with the input, and then a 1×1 convolution (again followed by batch normalization and a ReLU activation function) is applied. As explained before, in the end of each one of these blocks I also added a channel-and-spatial squeeze and excite block. A graphical representation for these blocks is shown in Figure 4.2.

Another idea that I adapted onto the proposed network was the use of a dense connectivity pattern, originally presented in the context of the aforementioned DenseNet (Huang et al., 2017) neural architecture – see Section 2.2.1.1 for the details on the DenseNet.

This idea is typically used for image classification tasks, and only more recently was it also tested in image segmentation (Zhen et al., 2019). The dense connectivity pattern eliminates the need to re-learn redundant feature maps between distant layers, since in the proposed model all the encoder blocks have directly access to the features learned by all the decoder blocks. However, the spatial dimensions between non-symmetric blocks are different, and thus a bi-linear scaling was used to concatenate the features from a smaller block into a larger one, and in the opposite case (i.e., features from a larger block into a smaller), I used average pooling.

The complete network architecture used in my experiments is illustrated in Figure 4.3. As in a standard U-Net, this model is composed by an encoder (on the left-side) and a decoder (on the right) that, together, form a hourglass-shaped architecture, that is trained to predict the

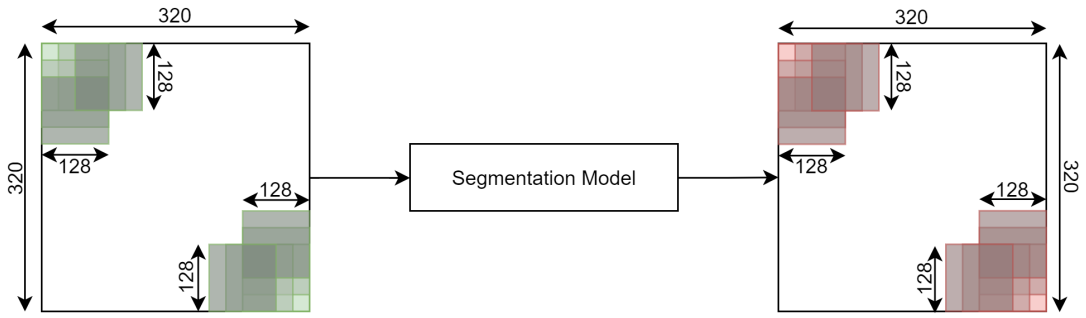


Figure 4.4: Strategy that produces the segmentation mask over the entire test images.

segmentation maps for the input aerial images. The contracting path of my model starts with a densely connected block, followed by a channel-and-spatial squeeze and excitation module. A maximum pooling operation is then applied to the result of this block, down-scaling the dimensionality of the feature maps successively until reaching the bottleneck of the U-Net (i.e., the representation in the middle). The bottleneck uses just a densely connected block, since Roy et al. (2018) argued that adding channel-and-spatial squeeze and excitation modules, at this point, does not improve the results. The expansive path starts with a 3×3 transposed convolution layer (followed by batch normalization and a ReLU non-linearity), optimally achieving up-sampling. Then, all the features learned by the blocks in the encoder (except the symmetrical block, that does not suffer from bi-linear up-sampling or average pooling) are concatenated and compressed using a 1×1 convolution. The result of this concatenation is further concatenated with the features learned by the symmetrical encoder block, and again compressed using a 1×1 convolution. Finally, a densely connected block and a channel-and-spatial squeeze and excitation module is used. Between each block in both the encoder and the decoder, dropout (Srivastava et al., 2014) with a drop rate of 50% is used. The model ends with a 1×1 convolutional layer with a sigmoid activation function, thus predicting a value for each pixel corresponding to the probability of that pixel belonging to a flooded area.

Notice that, in Figure 4.3, the output sizes are reported for an example input image with a resolution of 128×128 pixels (i.e., I generated patches of size 128×128 with a stride equal to 16, transforming each of the original input images with 320×320 pixels into 169 partially overlapping patches). Since I want to evaluate my model over the images with size 320×320, the final classification for a certain pixel is the arithmetic average of the values obtained in all patches containing that pixel, as shown in Figure 4.4. Using image patches with a smaller size helps in the sense of having models with fewer parameters, easier to fit in GPU cards with smaller amounts of memory. Moreover, in an initial set of tests, I verified that averaging the results from multiple patches actually improves the quality of the obtained segmentation masks.

Locations Present in the Dataset	Collection Date	Train Instances	Test Instances
Location 1 - Australia, Forbes	24/09/2016	143	62
Location 2 - Australia, Condobolin	25/09/2016	110	48
Location 3 - Madgascar, Maroantretra	06/03/2017	137	59
Location 4 - Colombia, Mocoa	31/03/2017	18	8
Location 5 - Peru, Sullana	23/02/2017	31	14
Location 6 - Panama, Chitre	30/04/2017	23	11
Location 7 - Dominican Republic, Duarte	23/04/2017	–	58

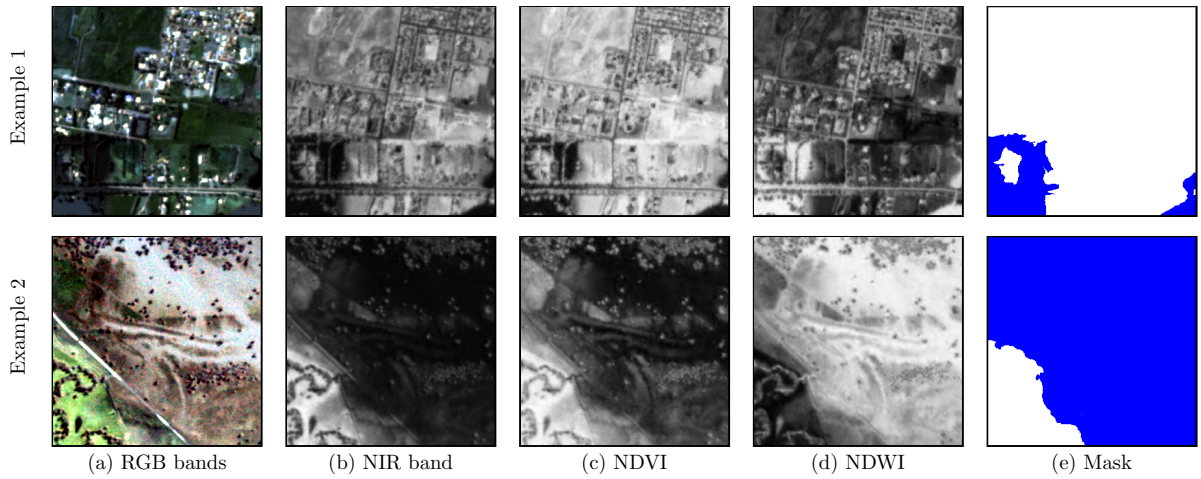
Table 4.1: Characterization of the dataset used in the MediaEval 2017 FDSI sub-task.

4.2 The Input Data Sources

In the context of the Flood Detection in Satellite Imagery (FDSI) sub-task of the MediaEval 2017 Multimedia Satellite Task (Bischke et al., 2017b), a large dataset of satellite imagery related to the delimitation of flood events was released. More precisely, the participants had access to a set of 722 satellite image patches obtained from PlanetScope satellites (i.e., a constellation of 120 satellites build and operated by Planet). The considered PlanetScope imagery has 4 spectral bands, namely blue (Band 1, 455 – 515 nm), green (Band 2, 500 – 590 nm), red (Band 3, 590 – 670 nm), and near-infrared (Band 4, 780 – 860 nm) bands, and a ground spatial resolution of 3.7 meters, corresponding to an orthorectified pixel size of 3 meters.

The data was collected between 01/06/2016 and 01/05/2017, and a detailed technical description is given in the specification of the Planet four band analytic ortho scene product. The complete set number of images was divided into a train set containing 462 images, and a test set with 260 images. Table 4.1 details the number of instances contained in the dataset used in this competition, discriminating between images from different regions. Notice that images from the 7-th location were only considered for evaluation, and the task organizers proposed to use that subset to assess the ability of the methods to operate on images from different regions to those used for model training.

As input to the proposed segmentation model, I used not only the four original spectral bands provided in the imagery from the FDSI competition, but also derived indexes such as the NDVI and the NDWI, and information about terrain elevation and imperviousness. The Normalized Difference Vegetation Index (NDVI) is a simple indicator that can be used to assess whether the target being observed contains live green vegetation or not. Considering the near-infrared band (NIR) and the RED band from an RGB image, it is possible to calculate the NDVI through the next equation:



(a) RGB bands (b) NIR band (c) NDVI (d) NDWI (e) Mask

Figure 4.5: Two examples for the various input features, as well as the segmentation masks (where blue corresponds to flooded areas), that were considered in my experiments.

$$\text{NDVI} = \frac{(\text{NIR} - \text{RED})}{(\text{NIR} + \text{RED})} \tag{4.4}$$

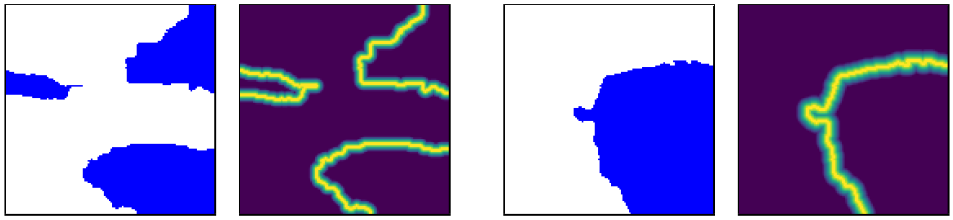
Similarly to the NDVI, the Normalized Difference Water Index (NDWI) is an indicator that can be used to assess whether the target being observed contains water or not. Considering GREEN as the green band from the RGB image, and again NIR as the near-infrared band, the NDWI can be calculated as follows:

$$\text{NDWI} = \frac{(\text{GREEN} - \text{NIR})}{(\text{GREEN} + \text{NIR})} \tag{4.5}$$

Figure 4.5 presents two examples from the images contained in the training dataset, presenting the original RGB image and the information on the near-infrared channel, together with the NDVI and NDWI values, as well as the provided segmentation mask.

The Digital Elevation Model (DEM) used in these experiments was the same JAXA DEM that I used to obtain an estimate for the water depth in ground-level imagery, detailed in Section 3.1. However, this time, the DEM was up-sampling (again using a cubic spline interpolation) to a resolution of 3 meter per pixel in order to match the resolution of the FDSI dataset.

On what concerns the terrain imperviousness information used in my experiments, it was obtained from the geodatabases presented by Gleeson et al. (2014), corresponding to permeability and porosity maps that are based on a recently completed high-resolution global lithology map, which differentiates fine and coarse-grained sediments and sedimentary rocks. The information presented in these databases was converted to a gridded format, in the same resolution of the



(a) Example 1 (b) Example 2

Figure 4.6: Two examples for the distance weight mask generated from the segmentation mask.

FDSI dataset, through a rasterization procedure. Each pixel in the resulting dataset corresponds to a value encoding the logarithmic permeability (i.e., an estimate, in squared meters, for the quantity of water allowed to flow through the surface of the terrain).

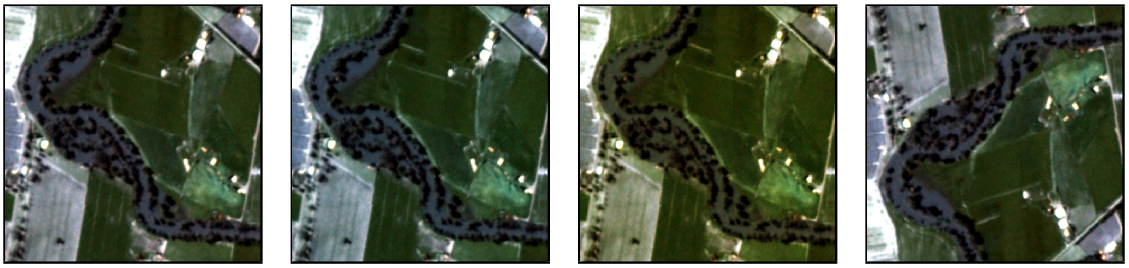
4.3 Training Strategy and Hyper-Parameter Tuning

The selection of hyper-parameters and model training strategies, followed the same guidelines described in Section 3.3, and all the source code is available online[1]. The last layer of the U-Net consists of a single channel with a sigmoid activation function, and the model training involves minimizing a customized version of the binary cross entropy-loss which emphasizes the importance of pixels closer to the segmentation boundary (Caliva et al., 2019). This emphasis is achieved using a pixel-wise weight that is calculated taking into account the distance to the segmentation border, for each pixel. These distance values are then transformed into pixel-wise loss weights using the procedure from Equation 4.6. Notice that, when leveraging the following equation, only pixels near the border are over-emphasized (i.e., pixels far from the border are assigned with a weight equal to 1.00), taking weight values between 1.00 and 1.20 (i.e., if the pixel is coincident with the segmentation boundary). Figure 4.6 presents two examples for the obtained weight maps, showing the segmentation masks for two patches of 128×128 pixels (i.e., the weight masks are calculated per input patch), together with the generated weight mask.

$$\text{weight}[i] = \left(0.10 + \max\left(0.90, 1.00 - \frac{\text{euclidean\_distance}[i]}{\max(\text{euclidean\_distance})}\right)\right)^2 \tag{4.6}$$

Training relied again on the Adam (Kingma and Ba, 2014) optimization algorithm, with the initial learning rate set to 10^{-3}. A random subset of 15% of the training image patches was used as a small validation set, decaying the learning rate 10 times when the validation loss did not decrease during 5 consecutive epochs. I built the validation set through a stratified approach

[1] `http://www.github.com/jorgemspereira/A-U-Net-for-Flood-Extent-Mapping`



(a) Original Image (b) Lower Temperature (c) Higher Temperature (d) Horizontal Flip

Figure 4.7: Examples for the input images with the considered data augmentation operations.

(i.e., trying to follow the same class distribution of the training split). To achieve this, I mapped each training instance to one of ten possible classes, representing the percentage of pixels that contain a flood. These classes were then used to create the stratified validation split.

As presented in Table 4.1, the number of instances per location is somewhat unbalanced. To deal with this problem, I weighted the value of the loss function not only pixel-wise, but also instance-wise, giving more weight to the image patches from locations with fewer examples. Considering p as the percentage of examples for a certain location, I weighted the instances of that location according to $1 - p$. To limit the minimum value of the weights by one, I then also added one minus the minimum weight value (i.e., $1 - \min(\text{all\_weights})$), between the weights of all the locations, to all the individual weights.

Training considered mini-batches of 12 image patches, together with the corresponding segmentation masks, with dimensionality $128 \times 128 \times n$, where n is the number of channels used in a particular experiment (i.e., when using only the RGB channels we have $n = 3$, with RGB plus the near-infrared channel $n = 4$, and when using all the features $n = 8$). The patches of training images were created through a generator that considered simple real-time data augmentation procedures, in an attempt to have a model that better generalizes across different locations and conditions. The generator first chooses randomly between a flip over the horizontal or vertical axis, and then chooses randomly to maintain the temperature of the colors, or change the color tone (i.e., by cooling or heating the image). For instance, if the generator selects cooling the color tone, the values of the blue channel are slightly increased, and the values of the red channel are slightly decreased (and vice-versa if the generator selects heating). Figure 4.7 presents examples for the resulting images after some of the data augmentation procedures.

Training proceeded for up to a maximum of 200 epochs, using the validation set to define an early stopping criterion, effectively stopping the model training if the validation loss does not decrease for 10 consecutive epochs. The final model weights were taken from the training epoch with the smallest value for the validation loss.

	Same Region		Different Region	
Model	**Acc**	**IoU**	**Acc**	**IoU**
Complete Approach	94.91	86.03	**92.69**	**79.16**
- Boundary Distance Weights in Loss	**95.29**	**86.93**	92.53	78.72
- Squeeze & Excitation	94.45	84.71	92.21	78.16
- Dense Connections	95.02	86.14	91.49	75.79
- Densely Connected Blocks	94.77	85.64	92.33	78.51
- Data Augmentation	94.81	85.63	92.40	77.94
FDSI 1st place (Nogueira et al., 2017)	–	88.23	–	84.10
FDSI 2nd place (Bischke et al., 2017a)	–	84.36	–	74.13

Table 4.2: Accuracy and IoU for all variants of the considered model, using all the four channels available in the MediaEval 2017 FDSI task. Some results obtained by the participants in the competition are also presented – see Table 2.2 for the complete list of results.

4.4 Evaluation Metrics and Experimental Results

The Flood Detection in Satellite Images (FDSI) dataset, as previously described, consists of high resolution satellite images derived from Planet's 4-band satellites. The images have 320×320 pixels, with four different spectral channels, and they are provided in the GeoTiff format. All the images are also projected in the UTM projection system using the WGS84 datum (EPSG:3857) and, leveraging the collection dates and the geo-spatial footprints of the images, I collected additional information to be used in complement to the original four spectral channels (i.e., information on terrain elevation and permeability, as explained in Section 4.2).

Following the evaluation methodology of the FDSI task from the Multimedia Satellite Task of the MediaEval 2017, I used the Jaccard index, commonly known as the PASCAL Visual Object Classes (VOC) Intersection-over-Union metric (IoU), to assess result quality. Intuitively, the Jaccard index, between the ground-truth and the predicted segmentation maps (i.e., sets of cells in a raster map corresponding to flooded areas), represents the ratio between the area that overlaps and the area of the union. More formally, considering t_p as the number of true positives (i.e., the pixels that contain evidence of a flood and were identified as such), f_p the number of false positives (i.e., incorrectly labeled flood pixels), and finally f_n the number of false negatives (i.e., flood pixels that were not identified), the intersection-over-union metric can be defined as presented in Equation 4.7. Note that, as considered in the official results of the competition, the IoU was calculated taking into account the set of all the pixels in the images from the test set.

$$\text{IoU} = \frac{t_p}{t_p + f_p + f_n} \tag{4.7}$$

Locations Present in the Dataset	Accuracy	IoU
Location 1 - Australia, Forbes	95.15	85.60
Location 2 - Australia, Condobolin	94.67	87.52
Location 3 - Madgascar, Maroantretra	95.41	88.79
Location 4 - Colombia, Mocoa	94.85	60.51
Location 5 - Peru, Sullana	92.14	78.37
Location 6 - Panama, Chitre	95.58	74.24
Location 7 - Dominican Republic, Duarte	92.69	79.16

Table 4.3: Accuracy and IoU results obtained when using all the four channels over each location considered on the test split from the MediaEval 2017 competition, using the complete approach.

Table 4.2 presents the results of ablation tests considering the RGB + near-infrared channels, in which I removed some of the strategies listed in the previous section (i.e., using the more common binary cross-entropy instead of the boundary distance weighted variation, removing the squeeze and excitation blocks, removing the dense connections between the blocks in the encoder and the decoder, replacing the densely connected blocks by the more typical convolutional blocks of the U-Net model, and discarding the data augmentation procedures).

The complete approach achieved the best results in the locations that do not appear in the training split. However, when considering the images from the same locations of the training set, the variation of the considered model that achieved the best results is the one that uses the unweighted binary cross-entropy. It is also interesting to notice that the removal of the dense connections between the encoder and the decoder slightly improves the results in the locations that appear in the training split, although it notably worsens the results in the images from different locations. Additionally, Table 4.3 presents the results obtained per each one of the considered locations in the test split, for the complete approach. Significantly worse results were achieved over the subset of photos from Colombia, which contains fewer training instances resulting in flood related pixels being misclassified as not containing evidence of a flooding event.

The bottom lines in Table 4.2 contrast the obtained results against those of other teams participating in the FDSI challenge. Team MultiBrasil (i.e., the team that achieved the first place in the competition), in particular, achieved the best results using an ensemble of neural networks models that were trained in each one of the six locations from the training split. The resulting six models were then used as feature extractors to a Support Vector Machine (SVM), thus producing the final segmentation masks. My complete approach, despite producing slightly inferior results, consists into only one neural network model that is trainable end-to-end, consequently facilitating the complete training procedure.

	Same Region		Different Region	
Input	**Acc**	**IoU**	**Acc**	**IoU**
RGB	93.13	81.67	89.92	72.25
Previous + NIR	94.91	86.03	**92.69**	**79.16**
Previous + NDVI + NDWI	95.23	86.79	90.81	73.22
Previous + Elevation	**95.37**	**87.06**	90.99	73.90
Previous + Imperviousness	95.21	86.65	89.14	68.22

Table 4.4: Accuracy and IoU using the considered model, for the test images in the same and in different locations, and when leveraging different types of input features.

Table 4.4 presents the results obtained with the complete model when using (i) only the RGB colors, (ii) RGB colors plus the near-infrared information, (iii) all the previous features plus the NDVI and the NDWI, (iv) the previous features plus terrain elevation, and (v) all the original channels combined with all other remote sensing products, in a total of 8 input features. The obtained results suggest that when considering images from the training locations, the additional features indeed lead to improvements in the overall results. However, in the test split with the images corresponding to the new locations, the obtained results are in fact worse, suggesting that these additional inputs are in fact less generalizable across different locations.

Figure 4.8 illustrates the segmentation results for six examples taken from the testing split of the dataset, when using all the considered input features, where the first column corresponds to the original image, the second column to the ground-truth segmentation mask, the third column presents the results of the complete model leveraging all the features, and the fourth column compares the obtained results with the ground-truth, highlighting in red the pixels corresponding to incorrect predictions. As highlighted in the examples, most errors produced by the proposed model correspond to misclassified pixels at the border of the flooded regions.

4.5 Overview

This chapter reported on a set of experiments evaluating the performance of an adapted version of the U-Net neural network architecture, originally proposed for the segmentation of medical images, on the semantic segmentation of flooded areas in high-resolution aerial images. Through experiments with the dataset from the FDSI sub-task of the Multimedia Satellite Task in the MediaEval 2017 competition, I showed that the proposed U-Net architecture is quite effective in this task, outperforming several baselines and some of the other approaches previously tested on this same dataset (e.g., support vector machines leveraging carefully engineered features or other neural approaches proposed by the participants on the task), and gaining also in the generalization across different locations, against other methods.

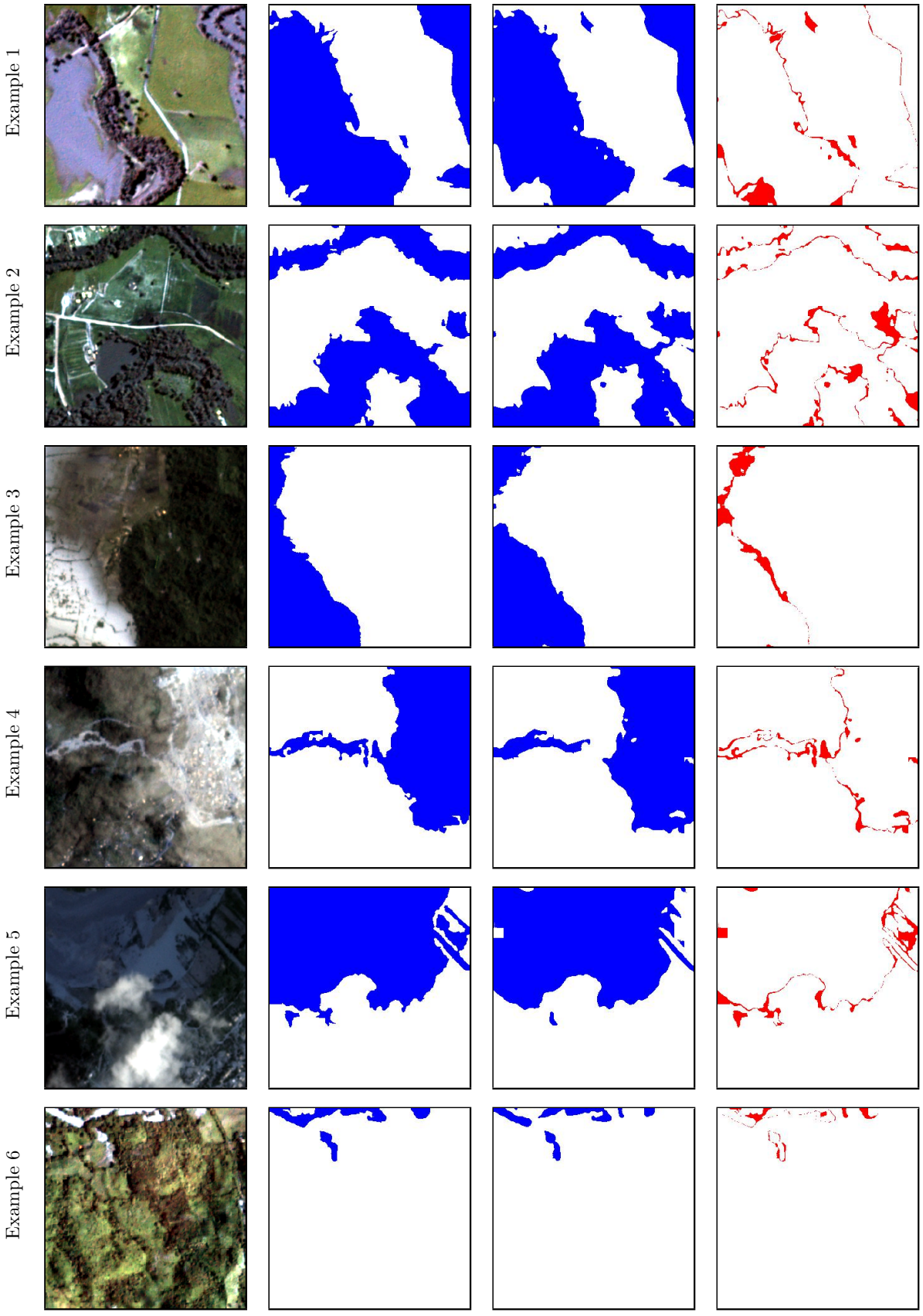


Figure 4.8: Six examples for the obtained results, comparing the ground-truth with the generated segmentation masks, illustrating the differences between them.

5 Conclusions and Future Work

Flood detection and monitoring is nowadays mostly made through remote sensing (e.g., satellite imagery). However, with an increase in the number of geo-referenced publications on social networks, perhaps this ground-level information can also be used to support those other sensors. Focusing on ground-level images, that can provide details that are difficult or even impossible to recognize in satellite imagery, this dissertation reported on experiments using convolutional neural networks to address several tasks related to flood detection (e.g., discriminate between images with flood evidence, or output the precise water depth in images with flooding events). The obtained results significantly outperform the official results obtained by the participants on the DIRSM sub-task of the Multimedia Satellite Task at MediaEval 2017, attesting the adequacy of the proposed neural methods.

Furthermore, since satellite imagery can also provide knowledge that is difficult to obtain from photographs taken at the ground-level, this dissertation also advances a novel semantic segmentation method, based on the U-Net neural architecture, that is able to receive as input multiple information from overhead sensors, and output a mask that delimits the area affected by the flooding event. The obtained results are comparable to those obtained by an ensemble method reported in the context of the FDSI sub-task at MediaEval 2017, also surpassing several other approaches and producing high-quality masks relying only on a single model.

5.1 Overview on the Contributions

The most important contributions of my M.Sc. thesis can be summarized as follows:

- **Creation of a dataset for flood severity estimation**: I built a dataset containing 10,734 images, aggregating and extending pre-existing datasets, annotated regarding three flooding severity classes: (i) no evidence of a flooding event, (ii) evidence of a flooding event with less than one meter of depth, and (iii) evidence of a flooding event with more than one meter of depth. I also presented a heuristic-based method for refining these classifications in order to obtain an accurate estimate, in meters, for the water depth;

- **A set of intelligent systems for flood detection**: I tested several convolutional neural networks (i.e., the DenseNet (Huang et al., 2017), the EfficientNet (Tan and Le, 2019), and the AG-DenseNet (Guan et al., 2018)) for classifying ground-level photographs into three different ways: (i) discriminate between images with flood evidence, (ii) classify images into the three aforementioned flood-related classes, and (iii) output the precise water depth in images with flooding events. Experimental results shows that the proposed methods significantly outperform previous proposals, for instance, reported in the context of the DIRSM sub-task at MediaEval 2017;

- **A proof-of-concept web application**: I developed a web application that shows, in a global map, all the geo-referenced images used in the reported experiments. In this web application, the users also have the possibility to upload new images, observe the classification made by the best performing model in terms of the predicted severity class and water depth, and register eventual corrections to the result of the automated method;

- **A novel satellite imagery segmentation method**: I developed a fully-convolutional neural network architecture that is able of receiving, as input, multiple information from overhead sensors (e.g., satellite images with RGB and near-infrared information, vegetation or water indexes, etc.), and output a mask that corresponds to the area affected by the depicted flooding event. Experimental results show that the proposed neural model achieves results that are on par with those from ensemble methods reported in the context of the FDSI sub-task at MediaEval 2017.

5.2 Future Work

Perhaps the most interesting idea for future work relates to combining the two methods described in Chapters 3 and 4. For instance, instead of the heuristic procedure described in Section 3.1, for estimating the ground-truth water depth, we can perhaps use the locations and dates of the photos to retrieve matching satellite imagery, and then use the segmentation model to estimate the associated flooded area. The limits of the flooded area, together with a high resolution Digital Elevation Model (DEM), can then be used to better estimate the water depth at the location of the photo. Similarly, we can perhaps use the regions associated to satellite images to retrieve geo-referenced ground-level photos contained within. By classifying these photos according to the flood-severity level, we can perhaps latter leverage this information (i.e., density of photographs depicting a flood) to help in delimiting the flooded areas.

It would also be interesting to continue improving the model for the classification of ground-level imagery, mainly in the task of predicting the exact water depth. For instance, as explained in the work by Chaudhary et al. (2019), in complement to end-to-end classifiers based on CNNs such as the ones reported on this dissertation (i.e., the EfficientNet, the DenseNet, and the AG-DenseNet models) it would be interesting to explore semantic segmentation models (e.g., the proposed U-Net neural architecture) to infer the position and relative level of occlusion of particular types of objects (i.e., objects with a known approximation of the real height, from which measurements can be derived) within images, in order to automatically produce a thin-grained estimate for the water level.

Different neural architectures can also be tested in the future, in an attempt to further improve results. For instance, Katharopoulos and Fleuret (2019) proposed an attention sampling approach similar to the attention guided convolutional neural network used in this dissertation. Their approach corresponds to a fully differentiable end-to-end trainable model that samples and processes only a fraction of the full resolution input images (i.e., the locations to process are sampled from an attention distribution computed from a low resolution view of the input).

The heuristic method that was used to infer the ground-truth estimates for the water depth, in meters, can also be improved, as mentioned previously. For instance, the current approach used to merge the different DEMs, in order to deal with missing information, is quite simple. In a recent study, Petrasova et al. (2017) proposed a generalized approach to DEM fusion, which produces a smooth transition between junction points, while preserving important topographic features. The transition between the different DEMs is controlled by a distance-based weighted average, considering also a spatially variable blending zone width that is based on elevation differences. For future work, it would be interesting to use the method described in the work from Petrasova et al. (2017) to better fuse the different DEMs used on my experiments.

Regarding the task of semantic segmentation of satellite imagery to obtain the delimitation of flooding events, a recent work by Wang et al. (2019) proposes an interesting novel approach that could be tested in the proposed variation of the U-Net. Their method is also based on a hour-glass architecture but uses an encoder in the reverse direction of the decoder that generates the segmentation map. The encoder performs encoding in the forward pass and the same parameters are used to perform decoding in the backward pass. In the future, I would like to test this idea, since it would significantly reduce the number of parameters of the proposed U-Net and, at the same time, hopefully increasing the quality of the segmentation results.

Besides taking ideas from previous developments in fully-convolutional networks for image segmentation, other recent developments in CNNs for image classification can also be integrated into the proposed segmentation approach. Examples include octave convolutions (Chen et al., 2019) or attention augmented convolutions (Bello et al., 2019). Other ideas that can be easily tested relate to (i) using model pre-training through auxiliary tasks such as discriminating patches basis on the containment of small versus large flooded areas, (ii) using loss functions specifically designed for segmentation tasks and that can better approximate the IoU metric (Chen et al., 2018b), or (iii) using cosine learning rate annealing (Xie et al., 2018) in order to obtain a better parameterization of the model, leading to better results.

Another recent idea for semantic segmentation of aerial imagery, also related to decreasing model size while simultaneously improving accuracy, involves the use of rotation equivariant CNNs (Marcos et al., 2018). Noting that in remote sensing images the absolute orientation of objects is arbitrary, this approach attempts to encode rotation equivariance directly in the neural network, through the use of rotating convolutions as building blocks and passing only the values corresponding to the maximally activating orientation throughout the network in the form of orientation encoding vector fields. This way, models do not need to learn specific (and redundant) weights to address rotated versions of particular semantic classes, thus improving accuracy even when using very small architectures.

Finally, for future work, it also would be interesting to try the idea proposed by Zhang et al. (2019) in order to explore long-range contextual information. Their Dual Graph Convolutional Network (DGCNet) models the global context of the input feature by modelling two orthogonal graphs in a single framework. The first component models spatial relationships between pixels in the image, whilst the second models inter-dependencies along the channel dimensions of the network's feature map. With this approach, the authors obtained state-of-the-art results in multiple datasets. It would be interesting to see if this approach could help improve the obtained results in the task of automatically creating flooding masks.

Bibliography

Agarap, A. F. (2018). Deep learning using rectified linear units (ReLU). *CoRR*, abs/1803.08375.

Altenberger, F. and Lenz, C. (2018). A non-technical survey on deep convolutional neural network architectures. *CoRR*, abs/1803.02129.

Antoniou, V., Fonte, C. C., See, L., Estima, J., Arsanjani, J. J., Lupia, F., Minghini, M., Foody, G., and Fritz, S. (2016). Investigating the feasibility of geo-tagged photographs as sources of land cover input data. *ISPRS International Journal of Geo-Information*, 5(5).

Audebert, N., Saux, B. L., and Lefèvre, S. (2018). Beyond RGB: very high resolution urban remote sensing with multimodal deep networks. *ISPRS Journal of Photogrammetry and Remote Sensing*, 140(1).

Avgerinakis, K., Moumtzidou, A., Andreadis, S., Michail, E., Gialampoukidis, I., Vrochidis, S., and Kompatsiaris, I. (2017). Visual and textual analysis of social media and satellite images for flood detection. In *Proceedings of the MediaEval Workshop*.

Badrinarayanan, V., Kendall, A., and Cipolla, R. (2017). SegNet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(12).

Bello, I., Zoph, B., Vaswani, A., Shlens, J., and Le, Q. V. (2019). Attention augmented convolutional networks. *CoRR*, abs/1904.09925.

Birant, D. and Kut, A. (2007). ST-DBSCAN: An algorithm for clustering spatial-temporal data. *Data Knowl. Eng.*, 60(1).

Bischke, B., Bhardwaj, P., Gautam, A., Helber, P., Borth, D., and Dengel, A. (2017a). Detection of flooding events in social multimedia and satellite imagery using deep neural networks. In *Proceedings of the MediaEval Workshop*.

Bischke, B., Helber, P., Schulze, C., Srinivasan, V., Dengel, A., and Borth, D. (2017b). The multimedia satellite task at MediaEval 2017. In *Proceedings of the MediaEval Workshop*.

Bischke, B., Helber, P., Zhengyu, Z., de Bruijn, J., and Borth, D. (2018). The multimedia satellite task at MediaEval 2018. In *Proceedings of the MediaEval Workshop*.

Cai, B. Y., Li, X., Seiferling, I., and Ratti, C. (2018). Treepedia 2.0: Applying deep learning for large-scale quantification of urban tree cover. In *Proceedings of the IEEE International Congress on Big Data*.

Caliva, F., Iriondo, C., Martinez, A., Majumdar, S., and Pedoia, V. (2019). Distance map loss penalty term for semantic segmentation. *arXiv preprint arXiv:1908.03679*.

Chaudhary, P., D'Aronco, S., de Vitry, M. M., Leitao, J. P., and Wegner, J. D. (2019). Flood-water level estimation from social media images. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 4(2).

Chen, G., Zhang, X., Wang, Q., Dai, F., Gong, Y., and Zhu, K. (2018a). Symmetrical dense-shortcut deep fully convolutional networks for semantic segmentation of very-high-resolution remote sensing images. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 11(5).

Chen, W., Zhang, Y., He, J., Qiao, Y., Chen, Y., Shi, H., and Tang, X. (2018b). W-Net: Bridged U-Net for 2D medical image segmentation. *CoRR*, abs/1807.04459.

Chen, Y., Fang, H., Xu, B., Yan, Z., Kalantidis, Y., Rohrbach, M., Yan, S., and Feng, J. (2019). Drop an octave: Reducing spatial redundancy in convolutional neural networks with octave convolution. *arXiv preprint arXiv:1904.05049*.

Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Cian, F., Marconcini, M., and Ceccato, P. (2018). Normalized difference flood index for rapid flood mapping: Taking advantage of EO big data. *Remote sensing of environment*, 209.

Dong, R., Pan, X., and Li, F. (2019). DenseU-net-based semantic segmentation of small objects in urban remote sensing images. *IEEE Access*, 7(1).

Doshi, J. (2018). Residual inception skip network for binary segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Doshi, J., Basu, S., and Pang, G. (2018). From satellite imagery to disaster insights. In *Proceedings of the AI for Social Good Workshop at NeurIPS*.

Feng, Y. and Sester, M. (2018). Extraction of pluvial flood relevant Volunteered Geographic Information (VGI) by deep learning from user generated texts and photos. *ISPRS International Journal of Geo-Information*, 7(2).

Fu, X., Bin, Y., Peng, L., Zhou, J., Yang, Y., and Shen, H. (2017). BMC at MediaEval 2017 Multimedia Satellite Task via regression Random Forest. In *Proceedings of the MediaEval Workshop*.

Ghosh, S., Das, I., Das, N., and Maulik, U. (2019). Understanding deep learning techniques for image segmentation. *ACM Computing Surveys*.

Gleeson, T., Moosdorf, N., Hartmann, J., and van Beek, L. P. H. (2014). A glimpse beneath Earth's surface: GLobal HYdrogeology MaPS (GLHYMPS) of permeability and porosity. *Geophysical Research Letters*, 41(11).

Guan, Q., Huang, Y., Zhong, Z., Zheng, Z., Zheng, L., and Yang, Y. (2018). Diagnose like a radiologist: Attention guided convolutional neural network for thorax disease classification. *CoRR*, abs/1801.09927.

Hazirbas, C., Ma, L., Domokos, C., and Cremers, D. (2017). FuseNet: Incorporating depth into semantic segmentation via fusion-based CNN architecture. In Lai, S.-H., Lepetit, V., Nishino, K., and Sato, Y., editors, *Asian Conference on Computer Vision*. Springer International.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Hu, J., Shen, L., and Sun, G. (2018). Squeeze-and-excitation networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Huang, G., Liu, Z., van der Maaten, L., and Weinberger, K. Q. (2017). Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Huang, X., Wang, C., and Li, Z. (2018). A near real-time flood-mapping approach by integrating social media and post-event satellite imagery. *Annals of GIS*, 24(2).

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the International Conference on Machine Learning*.

Katharopoulos, A. and Fleuret, F. (2019). Processing megapixel images with deep attention-sampling models. *CoRR*, abs/1905.03711.

Khan, S., Rahmani, H., Shah, S., and Bennamoun, M. (2018). *A Guide to Convolutional Neural Networks for Computer Vision*. Number 1 in Synthesis Lectures on Computer Vision. Morgan & Claypool Publishers.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations*.

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

Leung, D. and Newsam, S. D. (2010). Proximate sensing: Inferring what-is-where from georeferenced photo collections. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Liu, Y., Minh Nguyen, D., Deligiannis, N., Ding, W., and Munteanu, A. (2017). Hourglass-shape network based semantic segmentation for high resolution aerial imagery. *Remote Sensing*, 9(6).

Maggiori, E., Tarabalka, Y., Charpiat, G., and Alliez, P. (2017). Can semantic labeling methods generalize to any city? The inria aerial image labeling benchmark. In *Proceedings of the IEEE International Symposium on Geoscience and Remote Sensing*.

Malinowski, R., Groom, G., Schwanghart, W., and Heckrath, G. (2015). Detection and delineation of localized flooding from WorldView-2 multispectral data. *Remote sensing*, 7(11).

Marcos, D., Volpi, M., Kellenberger, B., and Tuia, D. (2018). Land cover mapping at very high resolution with rotation equivariant CNNs: Towards small yet accurate models. *ISPRS journal of photogrammetry and remote sensing*, 145.

Mou, L. and Zhu, X. X. (2018). RiFCN: Recurrent Network in Fully Convolutional Network for semantic segmentation of high resolution remote sensing images. *CoRR*, abs/1805.02091.

Mouratidis, A. and Ampatzidis, D. (2019). European digital elevation model validation against extensive global navigation satellite systems data and comparison with SRTM DEM and ASTER GDEM in central Macedonia (Greece). *International Journal of Geo-Information*, 8.

N. Tkachenko, A. Z. and Procter, R. (2017). WISC at MediaEval 2017: Multimedia satellite task. In *Proceedings of the MediaEval Workshop*.

Narayanan, R., Vm, L., Rao, S., and Sasidhar, K. (2014). A novel approach to urban flood monitoring using computer vision. In *Proceedings of the International Conference on Computing, Communication and Networking Technologies*.

Newsam, S. and Leung, D. (2019). Georeferenced social multimedia as volunteered geographic information. In Wang, S. and Goodchild, M. F., editors, *CyberGIS for Geospatial Discovery and Innovation*. Springer Netherlands.

Nogueira, K., Fadel, S., Dourado, I., Werneck, R., A V Mñoz, J., A B Penatti, O., Calumby, R., Li, L., A Dos Santos, J., and Torres, R. (2017). Data-driven flood detection using neural networks. In *Proceedings of the MediaEval Workshop*.

Oñoro Rubio, D. and Niepert, M. (2018). Contextual hourglass networks for segmentation and density estimation. *CoRR*, abs/1806.04009.

Pereira, J., Dias, M., Monteiro, J., Estima, J., Silva, J., Moura Pires, J., and Martins, B. (2019a). A dense U-Net model leveraging multiple remote sensing data sources for flood extent mapping. Under Review for International Journal of Geographical Information Science.

Pereira, J., Monteiro, J., Estima, J., and Martins, B. (2019b). Assessing flood severity from georeferenced photos. Under Review for GIR Workshop.

Pereira, J., Monteiro, J., Silva, J., Estima, J., and Martins, B. (2019c). Assessing flood severity from crowdsourced social media photos with deep neural networks. Under Review for Journal Multimedia Tools and Applications.

Petrasova, A., Mitasova, H., Petras, V., and Jeziorska, J. (2017). Fusion of high-resolution DEMs for water flow modeling. *Open Geospatial Data, Software and Standards*, 2(1).

Reuter H.I., A. Nelson, A. J. (2007). An evaluation of void filling interpolation methods for SRTM data. *International Journal of Geographic Information Science*, 21(9).

Ronneberger, O., P.Fischer, and Brox, T. (2015). U-Net: Convolutional networks for biomedical image segmentation. In *Proceedings of the Conference on Medical Image Computing and Computer-Assisted Intervention*.

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6).

Roy, A. G., Navab, N., and Wachinger, C. (2018). Concurrent spatial and channel Squeeze & Excitation in fully convolutional networks. In Frangi, A. F., Schnabel, J. A., Davatzikos,

C., Alberola-López, C., and Fichtinger, G., editors, *Medical Image Computing and Computer Assisted Intervention*. Springer International Publishing.

Ruder, S. (2016). An overview of gradient descent optimization algorithms. *CoRR*, abs/1609.04747.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge (ILSVRC). *International Journal of Computer Vision*, 115(3).

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and Chen, L.-C. (2018). MobileNetV2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

See, L. M. (2019). A review of citizen science and crowdsourcing in applications of pluvial flooding. *Frontiers in Earth Science*, 7:44.

Selvaraju, R. R., Das, A., Vedantam, R., Cogswell, M., Parikh, D., and Batra, D. (2016). Grad-CAM: Why did you say that? Visual explanations from deep networks via gradient-based localization. *CoRR*, abs/1610.02391.

Shen, X., Wang, D., Mao, K., Anagnostou, E., and Hong, Y. (2019). Inundation extent mapping by synthetic aperture radar: A review. *Remote Sensing*, 11(7).

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556.

Smith, L. N. (2017). Cyclical learning rates for training neural networks. In *Proceedings of the IEEE Winter Conference on Applications of Computer Vision*.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1).

Srivastava, S., Muñoz, J., Lobry, S., and Tuia, D. (2018). Fine-grained landuse characterization using ground-based pictures: a deep learning solution based on globally available data. *International Journal of Geographical Information Science*.

Szegedy, C., Ioffe, S., Vanhoucke, V., and Alemi, A. A. (2017). Inception-v4, inception-ResNet and the impact of residual connections on learning. In *Proceedings of the Association for the Advancement of Artificial Intelligence Conference*.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Tadono, T., Ishida, H., Oda, F., Naito, S., Minakawa, K., and Iwamoto, H. (2014). Precise global DEM generation by ALOS PRISM. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 2(4).

Tan, M., Chen, B., Pang, R., Vasudevan, V., Sandler, M., Howard, A., and Le, Q. V. (2019). MnasNet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Tan, M. and Le, Q. V. (2019). EfficientNet: Rethinking model scaling for convolutional neural networks. *CoRR*, abs/1905.11946.

Volpi, M. and Tuia, D. (2017). Dense semantic labeling of subdecimeter resolution images with convolutional neural networks. *IEEE Transactions on Geoscience and Remote Sensing*, 55(2).

Wang, B., Glossner, J., Iancu, D., and Gaydadjiev, G. N. (2019). Feedbackward decoding for semantic segmentation. *CoRR*, abs/1908.08584.

Xie, J., Hes, T., Zhang, Z., Zhang, H., Zhang, Z., and Li, M. (2018). Bag of tricks for image classification with convolutional neural networks. *CoRR*, abs/1812.01187.

Xu, G., Zhu, X., Fu, D., Dong, J., and Xiao, X. (2016). Automatic land cover classification of geo-tagged field photos using deep learning method. In *Proceedings of the Fall Meeting of the Association of American Geographers*.

Zhang, L., Li, X., Arnab, A., Yang, K., Tong, Y., and Torr, P. H. S. (2019). Dual graph convolutional network for semantic segmentation. *CoRR*, abs/1909.06121.

Zhao, H., Shi, J., Qi, X., Wang, X., and Jia, J. (2017). Pyramid scene parsing network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Zhen, M., Wang, J., Zhou, L., Fang, T., and Quan, L. (2019). Learning fully dense neural networks for image semantic segmentation. *CoRR*, abs/1905.08929.