"Con fundamento en el artículo 21 y 27 de la ley federal del derecho de autor y como titular de los derechos moral y patrimonial de la obra titulada "WHEN THE EARTH TREMBLES, AUTOMATIC DAMAGE ASSESMENT USING AERIAL IMAGERY", otorgo de manera gratuita y permanente al instituto tecnológico autónomo de México y a la biblioteca Raúl Baillères Jr. Autorización para que fijen la obra en cualquier medio, incluido el electrónico y la divulguen entre sus usuarios, profesores, estudiantes o terceras personas, sin que pueda percibir por la divulgación una contraprestación"

Amaury Gutiérrez Acosta

FECHA

FIRMA

Contents

Foreword

It is an inspiring moment for the field of computer vision; machines are now capable of performing tasks that we thought were impossible, reaching new milestones each year. It has been a long time since computers were just large furniture in cold university rooms. Computer power has grown exponentially since those days. In recent years, techniques initially discarded because they were computation intense are now being unburied and have been showing incredible results in today machines.

The objective of this work was to explore the possibilities that these techniques can offer in classic problems such as landcover classification. In particular, we wanted to teach a neural network how to recognize a specific element in aerial imagery and then use the tailored features as input to train classic classifiers such as random forest or support vector machines.

This preface serves as an introduction to this work. It gives a short review of the contents of each chapter, and shows how is this dissertation structured.

We explore our motivations in Chapter 1. We expose why our work is essential, and we give a clear explanation about the objective of the experiment. We offer a historical review on the grounds of several natural disasters that have occurred in Mexico, focusing mainly on earthquakes. Additionally, we explore the scope of the project by mentioning limitations and objectives.

We show an extensive literature review in Chapter 2. All the way back to the well-known technique to analyze handwritten digits with the convolutional architecture that started the revolution. A review of modern applications in more complex situations such

as object recognition in images. We explore methods to perform damage assessment in the aftermath of natural disasters as it was the primary motivation for this study. We unveil the mathematical details that make this technique to work, diving into the process of backpropagation. We talk about last layer activation functions such as the Sigmoid and the Softmax.

We elaborate on the architecture of our pipeline in chapter 4. This process includes data gathering, data curation, the training of the network and the prediction. We divide the details of the implementation into two parts, client and server side. We delve into the reasoning behind some decisions as the flux of the project suggested new ideas and custom tools emerged. We show our resulting map, and how did we obtain it.

Finally, in chapter 5, we talk about future work, and what are the main conclusions to which this practice led us. We include several improvements that we can address to obtain better results.

This work was the result of an internship spent on the Stevens Insitute of Technology (SIT) in Hoboken, New Jersey, during the summer of 2017. I worked under the supervision of Andrea Garcá Tapia, José Emmanuel Ramirez Marquez from the SIT, and Raul Sierra Alcocer from the National Commission for the Knowledge and Use of Biodiversity (CONABIO).

Chapter 1

Introduction

> Good work is not done by 'humble' men. (...) A man's first duty, a young man's at any rate, is to be ambitious.
>
> A Mathematician's Apology
> G. H. Hardy

Earthquakes are unpredictable phenomena, which damaging capabilities can be catastrophic. Given the limited nature of resources, its correct allocation is vital to mitigate the damage in the aftermath. Technology makes the labors of logistics and rescues a lot easier. This chapter explores the motivation, history, objective and scope of the present work.

1.1 Motivation

Massive collaboration proved to be a fundamental resource to face the aftermath caused by the earthquake of September 19, 2017, in Mexico City. The use of social networks allowed communication between rescue, logistics, and civil society. We learned lessons about the scope and limitations of this association.

However, what happens when the conditions and technological infrastructure of large cities do not exist? This work explores other possibilities in which current technologies

can help us when the situation in which the natural disaster occurs is different. Focusing on the study of images captured by drones during the days after the earthquake of September 7, 2017, in towns of the state of Oaxaca, we propose an analytical framework that allows detecting damaged areas in an automated way.

To do this, we applied techniques that allow the use of models that have been previously trained in massive supercomputers, adjusting them to our particular problem. This process reduces the number of resources needed, in both time and infrastructure, to obtain results with high accuracy. In the future, this will allow allocating efforts in a more agile and efficient manner.

1.2 History

Given to the particular geographical conditions, Mexico is very prone to seismic activity. The Cocos and the Rivera plates subduct below the North American plate, and the Pacific plate separates from the North American plate along the Baja California Gulf [**?**].

According to historical research, there has been a registry of these natural disasters since the Pre-Columbian age. The level of material damage and the death toll has been increasing ever since as the population and the cities grew. In Figure 1.2 we can see a pictogram that according to [**?**] means "in the year 11 rabbit the earth trembled during the night".

The use of cartography to place the damage information also is useful not only to allocate resources in the aftermath of the disaster but also serves as historical evidence. In figure 1.3 taken from [**?**] we can see the buildings damaged by an earthquake known as the *San Juan de Dios* earthquake [1]. This event dates back to March 8, 1800, years before the Mexican Independence, during an age of economic and social prosperity.

[1] Back then the earthquakes where name accordingly to the day in which they happened.

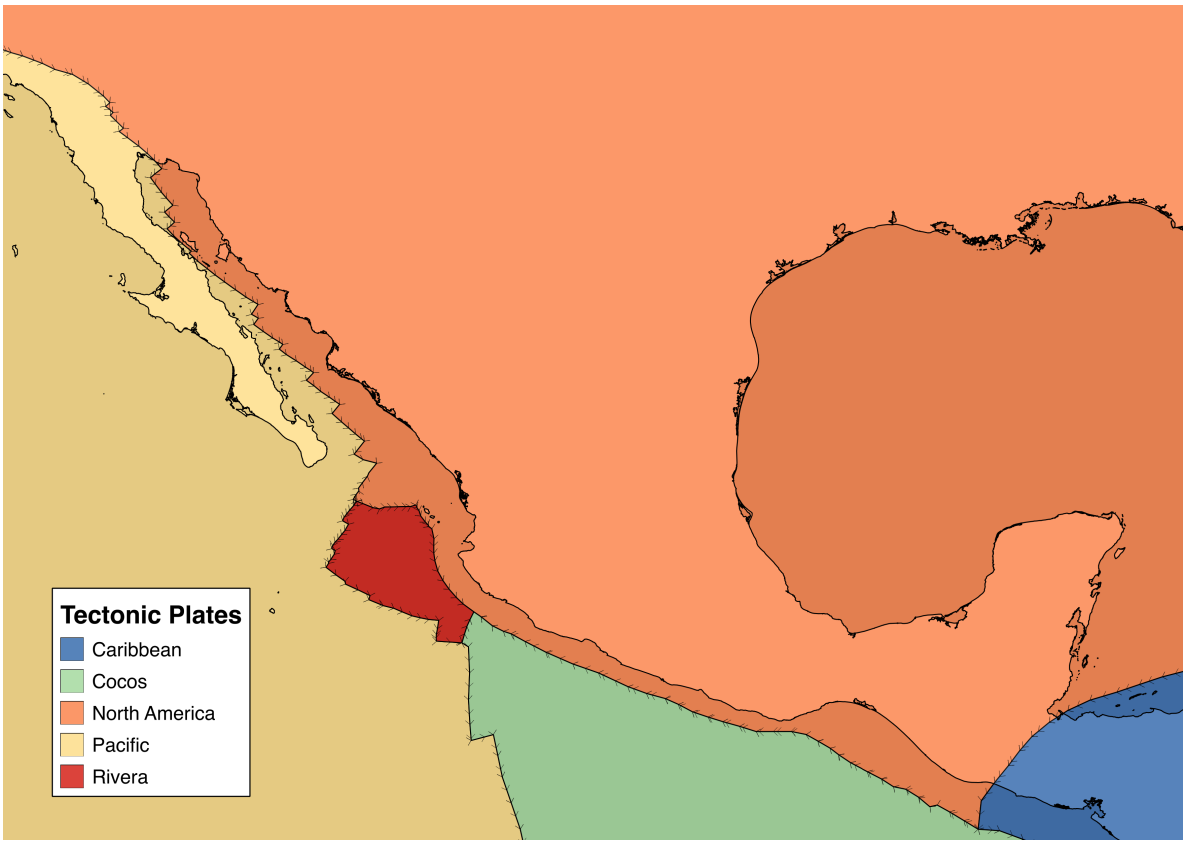


Figure 1.1: We can see the five tectonic plates meeting near Mexico.

According to historical records, a critical earthquake occurred in the year 1787, causing a massive tsunami that affected the coasts of Oaxaca along 450 kilometers. Paradoxically, this catastrophic event didn't produce as much damage as recent ones due to the lack of established cities in the state back then.

Years 1845 and 1858 also had significant earthquake events that destroyed infrastructure in Mexico city. We can find a mapping of the damaged buildings in [**?**]. The fall of the iconic Angel of Independence was the result of an earthquake on July 28, 1957, in an event that left dozens of deaths.

Nevertheless, the real breaking point in the Mexican seismological history came on September 19, 1985. That morning, an 8.1 magnitude earthquake stroke the city collapsing many buildings and leaving a death toll measured by thousands. Next day, the aftershock collapsed even more buildings, damaged the day before. The destruction

and chaos that the earthquake provoked still lingers in the memory of the people that witnessed such a terrifying event.

The way people reacted after September 19, 2017, was because they grew up in an ambient of constant fear to the quakes.

Two earthquakes took place during September 2017. While the second one devastated Mexico City, attracting help from all over the world, the first one was less known even though it was the earthquake with the highest magnitude that has hit the country in the last century. Both were catastrophic for the state of Oaxaca. The locality of Juchitan de Zaragoza was one of the most affected, buildings collapsed, and several people died.

1.3 Objective

It is not coincidental that our brief historical summary focused mainly in Mexico City. The lack of infrastructure and the distance from large cities make it harder to reach certain towns with resources and aid. We want to explore the use of new technologies to focus our efforts and use them better.

The National Center for Prevention of Disasters (CENAPRED) provided us with imagery taken on the days after the Chiapas earthquake took place. They flew drones over the towns of Juchitan de Zaragoza, Santa Maria Xadani, and Union Hidalgo. We propose to use those images to train a model that lets us geolocate collapsed buildings and create a map with them. In the case of another catastrophic event of similar nature, drones can be sent to fly over the affected area, and our proposed tool can be used to narrow dramatically the places which assessment teams must visit. This would reduce the amount of resources and time needed to correctly asses the damage in the earthquake aftemath.

In Mexico, CENAPRED is in charge of channeling financial resources in the case of

a natural disaster.

We want to explore the use of Convolutional Neural Networks (CNNs) in this context. We believe that this field is useful for disaster assessment, and will bloom in the coming years.

1.4 Scope

We don't pretend to provide a perfect mapping of every single collapsed building. We understand the inherent limitations of automatic methods, but we believe that we can achieve greater things working together with this new tools. Machine learning methods are not a panacea by no means, but a useful device that can help us reach places we could just imagine before.

We have to keep in mind that this is only the beginning and that there is vast room for improvement. The literature review in which we will dive more in-depth in the next chapter suggests ways of improving the results that we got, but its implementation is out of the scope of this work.

It would be far too ambitious to cover every topic that is involved in the process of the classification using CNNs. We want to explain how do the networks work to a certain extent, but it is not in the scope of this work to untangle every single detail.

As we already mentioned, the field of Computer Vision is in its climax. Reviewing every single article written on the topic would be a daunting task. We offer a brief literature review that gives some context about the state-of-the-art, and we hope to build upon ideas and efforts done by a myriad of people. We aimed to create a system capable of processing imagery and that allows an institution such as CENAPRED to deploy it into a cluster for efficient computation.

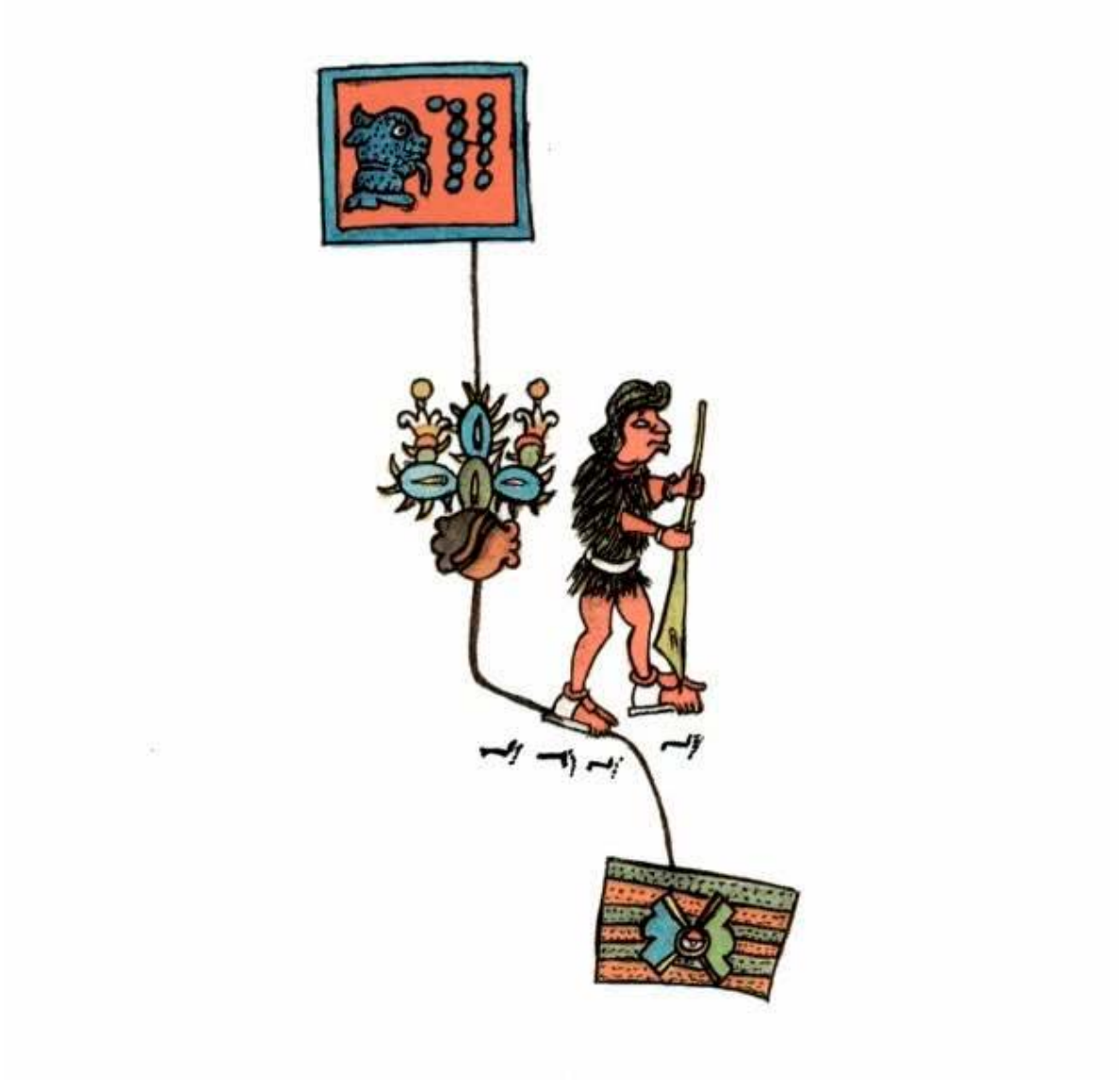


Figure 1.2: Pictogram depicting an earthquake.

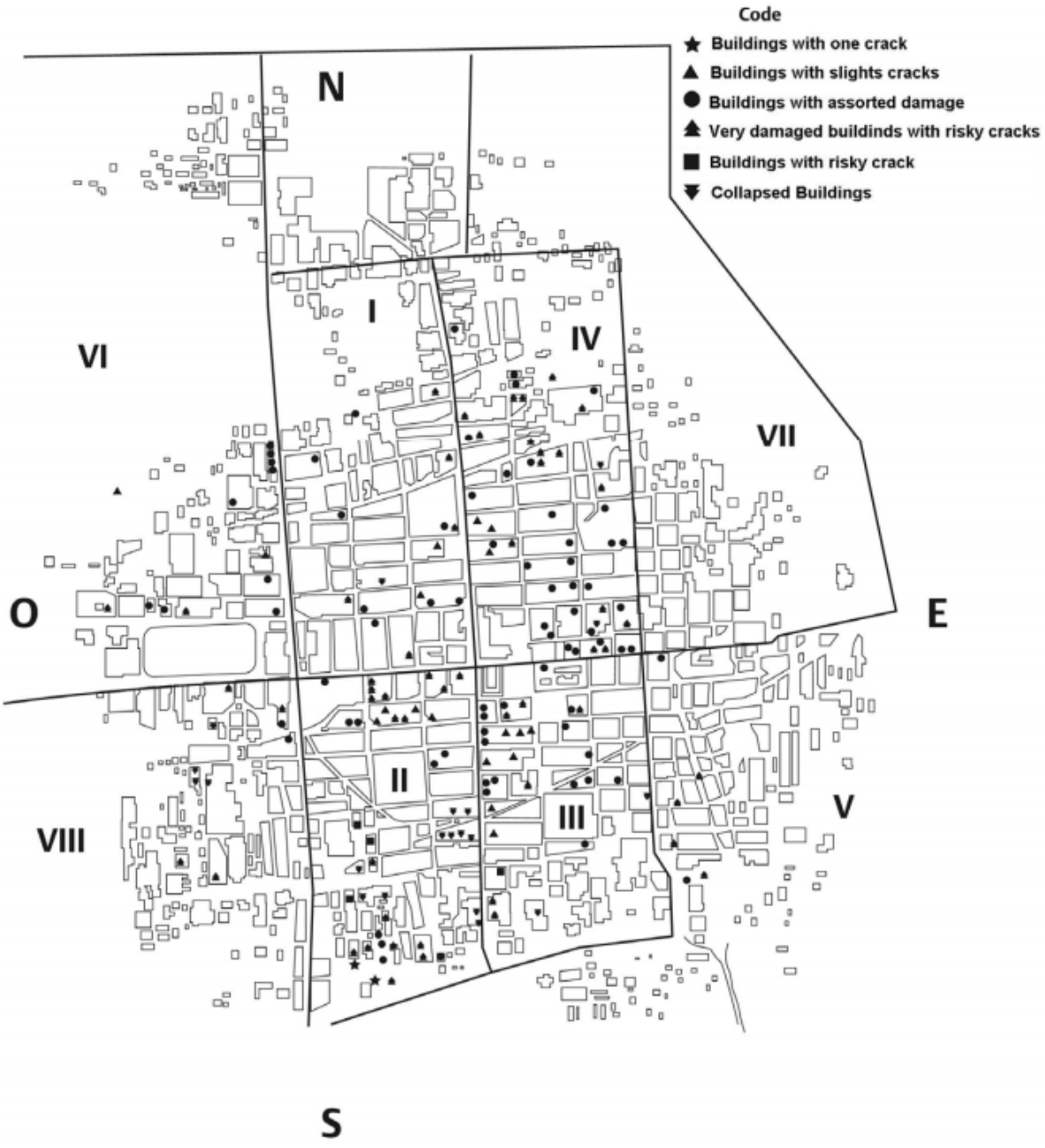


Figure 1.3: A damaged building map for the *San Juan de Dios* earthquake in Mexico City.

Chapter 2

Literature review

> If I have seen further it is by
> standing on the shoulders of giants.
>
> Letter from Sir Isaac Newton to
> Robert Hooke

In this chapter, we talk about the state of the art in computer vision and how it has been used for remote sensing problems. We also give a brief account of natural disaster assessment, and how are these machine techniques applied in this sense. We use the Chiapas earthquake that happened on Septmber 7, 2017 as a study case because of the data that was publicly made available by the CENAPRED. In this chapter the mathematical details of the CNN are exposed. How does backpropagation works, and differences between multiclass and multilabel classification.

A wonderful introduction to neural networks in the context of remote sensing can be found in [**?**]. It reviews all the necesary mathematical tooling to deal with the processing image algorithms, and implements the algorithms in Python. A survey on disctint aspects of how transfer learning is used can be found in [**?**].

In the context of natural disaste rs other options have been considered. For example Kryvasgeteue et. al. use the twitter activty during Hurricane Sandy to build a model. They found that social media activity is related with the proximity of the region in which the Hurracane hits[**?**]. Nevertheless, the mention that the nature of the data

available both in quantity and quality is rather unususal in part because of the severity and magnitude of damage that the Hurricane Sandy brought about.

The idea of using features crafted by a neural network has been widely explored. Jeff Donahue et. al. developed a framework which they called Deep Convolutional Activation Features (DeCAF) [**?**]. They feed traditional methods with the features extracted from the CNN obtaining accuracies compared to the state of the art at the time.

They took the activations from the n^{th} hidden layer, the layer previous to the fully connected one, and use them to train two classifiers: a logistic regression, and a support vector machine, in four different environments: object recognition, domain adaptation, subcategory recognition and scene recognition. Object recognition tested the ability of the classifier to correctly assign a class to a given image. They showed that the depth of the layer dramatically improves the performance of the features. Domain adaptaion consist of testing the robustness of the classifier when the source of the image varies, in this case they use a testing set consisting of images of office objects taken from Amazon, a webcam and a Dslr camera. The classifiers where able to cluster the objects across domains regardless of the origin of the images. In subcategory recognition, the task at hand is to differenciate individual categories inside a super-category, for example discriminate among two types of birds using a training set consisting only of images of birds. They found that without further fine tuning, the features extracted from the neural network achieved accuracies far superior to the ones available in the literature at the time. Finally, they tested the extracted features at the task of semantical classification. Instead of requiring the classifier to give a category of the object in the picture, the classifier must semantically classify the scene instead. This is considered a very difficult task, and the previous approach at the moment was to use hand-engineered features and a multi-kernel learning baseline. While the accuracies reported by this task where quite low, they improved the existing benchmark which provides strong evidence that the features extracted from the deeper layers of a CNN excel at extracting information from the images even though the network was not design for any of these tasks. Additionaly they developed an open source framework that later matured into Caffe [**?**], a framework designed to train CNN models.

In another attempt that adds to the evidence that features engineered by the Neural Network work pretty good off the shelf, Razavian *et al.* [**?**] use features extracted from the CNN `OverFeat` model which was made public by Le Cun *et al.* [**?**]. In the original article, they exploring the advantages of training a network simultaneously for several tasks in this case: classify, locate and detect objects in images. Razavian et al. experiment with these features to perform classification in tasks that gradually move away from the one for what `OverFeat` excell. Thet tested the features extracted from the CNN model together with a traditional support vector machine in the tasks of: image classification, fine grained recognition, attribute detection, and visual instance retrieval. They found that the CNN proved to be a strong competitor against more sofisticated methods involving handmade features.

Transfer learning consists of using the knowledge adquired by a model during a training phase to another domain of knowledge. It is explored by Yosinski *et al.* in [**?**]. They propose to use an already trained architecture in new tasks by replacing diferent layers and retraining only the topmost layers. An experiment was designed to test the transfereability of different layers. They found strong evidence that sugest that even thought the training sets are of different nature, features transfered from different tasks improve the performance of the CNN against random weights. To perform their experiment, they used ImageNet [**?**] spliting the dataset obtaining two different groups of images. They also found that that the transfereability is negatively affected by the deepness of the layers, mening that higher layer features are specialized to the given task. This ideas gave us a clear starting point for the task in our hands.

Image segmentation is another task that has been explored using neural networks. The idea is to obtain a semantic partition of the images. In our case it would be desirable to have a heat map that can separate the damaged buildings from the rest of the elements of the image. In [**?**] they use the features extracted from the CNN to segment such an image. The architectures of several classification networks were modified into fully convolutional networks, and the fetures learned were fine-tuned for the segmentation task. This process is very promising for the field of Remote Sensing

where thec classification features are engineered by using traditional methods such as manipulating the image information without taking into account the semantic context. This topic has also been explored in other topics such as biomedicine [**?**].

The possibility of having a single model that can perfom correctly in many different tasks is explored in [**?**]. Kaiser *et al.* aimed to create a single model capable of performing well in very different tasks. They noticed that while CNNs give impresive results in a variaty of tasks, the training efforts are to be repeated for every situation. They created an architecture capable of receiving training data from 8 different corporea. Ranging from image classification to language translation and natural language processing, the tasks in hand are far appart from each other. Given the very different nature of the training data, enconders needed to be created to transform it into a joint representation space. Several tasks such as translation from English to German and translation from French to English shared the same encoders promoting stronger generalization of the features. While not achieving state of the art results in any of the tasks, they found that it is indeed possible to train a universal CNN for different tasks. The domains in which the data is limited are benefited with the joint training while the other tasks suffer of little or no loss in acurracy at all.

The use of active learning together with semisupervised learning tenchniques is explored in [**?**]. Zhou and Prasad used an idea known as active learning to improve the performance of a classifier taking the most of small labeled datasets.

2.1 Computer vision

Le Cun *et al.* [**?**] propose to use an architecture of a multilayer neural network that was able to learn directly from the data with no prior feature extraction. In contrast to the usual path that was used in the context of pattern classification, they created an architecture that was able to automatically extract the features directly from the date without prior manipulation. Instead of using a fully connected network, they proposed a locally connected net. It was capable of extracting local features and passed them

down to the subsequent layers in what they called a *feature map*. Each unit took the information of a 5×5 neighborhood of the pixel in the previous layer. The last layer of the architecture consisted of ten units that represented each of the possible digits. This architecture was trained using backpropagation are now known as Convolutional Neural Networks (CNNs). The big leap forward of their result was that their architecture needed very little information about the task it was performing, they were able to extend the use of their method to other symbols, however, they state that the method was not able to be applied to very complex objects.

Before CNNs became widely used in computer vision tasks, the use of neural networks to automatically detect roads was already being explored [**?**]. They proposed a simple architecture with a single hidden layer. They use the now standard procedure of cropping small patches from the complete image and worked on the RGB color space. Aerial imagery in addition to vector information on roads is used as training data. Instead of manually tag the images they proposed a model that asumes certain amount of thickness in the roads, which are tipicaly with no dimension. To reduce the input dimensionality, they applied Principal Component Analysis keeping the most informative principal components. As a method of post-processing, they use a CNN to reduce the noise in the images. Effectively getting rid of false positives and false negatives by using context information. Of the important lessons learned by this experiment is the value of the random rotations in the training data. It is common that road networks in cities form grids, they realised that a model trained with information from a particular city will perform poorly if data from a new city is shown to it. They found that this can be relived if random orientations are applied to data as when we are dealing with birdeye sight, there is no correct orientation for images. They also found that adding pre-procesing such as edge detection techniques showed no improvement to the performance of their pipeline. This was attributed to the fact that the neural network crafts features of this nature when learning to perform the task of recognizing roads.

There have been several attempts to use features extracted from a CNN to be used in a different context. Michael Xie *et al.* [**?**] examine this approach by training a CNN on top of the well-known `VGG F` model. First, they replace the fully connected layers on top of that model with a convolutional layer. Then they re-train the features

the model learned from ImageNet with aerial images and nighttime images gather from the National Oceanic and Atmospheric Administration (NOAA). While they get nice accuracy results from using daytime images to predict nighttime light, it was not the purpose of their research. Instead, features crafted by the network are extracted and used to train a model to predict poverty from satellite imagery. In order to do so, they use these features as input for a logistic regression classifier. To compare their model, they train four other models extracting features from a survey, features from ImageNet itself, features from the nighttime light intensities and features from ImageNet and nighttime light intensities at the same time. The transfer model outperforms every model except for the one based on survey data. This strongly suggests that the transfer learning technique is actually extracting complex information from the aerial scenes. They mention that this approach can be useful when conducting surveys is prohibitively expensive.

With the tremendous advances that computer power has suffered in the late years, this has been proven to be incorrect. In 2009 a big image database was gathered and published [**?**]. Ever since this database became the defacto dataset to test classification methods. A few years later, in 2012 Krizhevsky *et al.* [**?**] proposed the use of CNNs in this daunting task.

2.2 Remote sensing

In late years groundbreaking advances in computer vision have led to tremendous advances in other science fields. In particular, we are interested in landcover classification.

The use of CNNs in the context of landcover classification was explored by Kussul *et al.* [**?**]. They used an ensemble of CNNs to obtain state of the art results in the classification of different types of crops using multitemporal and multisensor satellite data. They explore 2 approaches, first they use a 1-D CNN to perform the convolutions in the spectral domain by stacking the different bands from the Sentinel-1 A and Landsat-8 scenes. This process outputs a pixel-wise classification, then they perform a traditional 2-D CNN on the scenes. In order not to lose resolution with the 2-D CNN, they use a

sliding window approach assigning the class to the center pixel of the sliding window. Finally, they ensemble both opinions and filter the result to improve the quality of the map.

The usual approach with landcover classification is the use of classical classification methods such as support vector machines (SVM) and random forests (RF). In order to improve the performance, features must be handcrafted from the original bands. In [**?**], Grant *et al.* explore the use of Transfer Learning and Data Augmentation in the context of remote sensing images. By exploring well-known high-resolution datasets, they obtain state of the art results.

Segnet papers: [**?**] in [**?**] they enhace their approach by extending their own architecture to include a Bayesian approach. The idea is to add a model of uncertainty to the CNN and use this information to get more acurrate guesses on each of the pixels. They report that this feature adds some improvement in the level of accuracy for many types of architectures not only their own SegNet.

2.2.1 Data augmentation

Data augmentation is a technique used to artificially increment the size of the training dataset by applying an affine transformation to the images. It is often used when tagged data is scarce and difficult to obtain. The usual transformations include rotations and reflections. When using this technique we should be careful about the orientation of the objects, for example, a building upside down makes no sense, so there is no use to make the network learn features on objects that it won't see in the wild. Fortunately, aerial imagery doesn't present this problem. There is no particular orientation that can be considered correct when the pictures are taken from above. This means that we can dramatically boost the size of our dataset.

The reasoning behind this idea is that when we see a picture, our brain automatically orients it into its correct position. By showing the network with different positions and orientations of an object we enrich its knowledge about it.

We can think of the neural network as a newborn kid, in the beginning, they expe-

rience its environment for the first time.

2.3 Damage Assessment

2.4 TensorFlow

TensorFlow is a machine learning system developed by the Google Brain Team to supersede its first-generation system, DistBelief. It was built on top of the lessons learned during DistBelief development. One of the fundamental concepts that lead the team to create a new system from scratch was the need for flexibility. TensorFlow was thought as a way to expressing machine learning algorithms using a common interface with implementations targeting a wide range of devices. Complex models that where first implemented in DistBelief such as Inception got a performance boost of a factor of x6 [**?**] when it was ported to the new system. TensorFlow was opened sourced on November 9, 2015 under the Apache 2.0 license.

In this section, we will untangle some of the details of how TensorFlow and its graph model work [**?**].

It uses an elegant data-flow system in which both the operations and the state of the algorithm are represented as nodes and edges in a directed graph. This lets the system to pre-calculate an optimal sub-graph before starting the calculations.

With flexibility in mind, this system lets the user define new operations and register them to use within the framework. Additionally, it was developed to target different platforms, from machine clusters to mobile devices, these implementations are known as *kernels*. Using the same programming model, TensorFlow decides in runtime which pieces to use. This is useful when you take into account the whole development process of a data product and how it evolves. We can think of a common scenario, first, the developer experiments with data in a single computer before deploying the system to train with a larger data set in a cluster of computers, when the model is trained, it

can be deployed to an online service which will run on a single computer or it can be implemented to be used in a mobile device for offline use. In each of these steps the underlying environment is completely different, however, TensorFlow adapts automatically to each situation.

As a common interchange data format, it uses tensors. With machine learning algorithms, it is often the case to have sparse data, encoding it as dense tensors is a clever way to save space. As we mentioned before, TensorFlow uses a graph to represent both the state and the operations. Nodes represent operations. Edges represent inputs and outputs between these operations. The system takes its name from the tensors flowing through this pipes. Although it is not of particular importance to our experiment, it is worth mentioning that TensorFlow supports algorithms with conditional and iterative control flow, which means that it can be used, without further tuning, to train Recurrent Neural Networks which are very important in fields like speech recognition and language modeling.

The system also provides a library that allows symbolic differentiation. As many machine learning techniques rely on Stochastic Gradient Descent to train a set of parameters, this feature makes easier to explore new techniques as the backpropagation code is automatically produced for any combination of operation nodes.

TensorFlow was built with huge data-sets in mind. It provides an intern library that allows the distribution of datasets that would be to large to fit in RAM. Instead, data can be sliced, taking advantage of how some algorithms work. Additionally, comunication between nodes use lossy compression, taking advantage of the fact that some of the machine learning algorithms are tolerant to reduced precision arithmetic. It is important to mention that it is possible to extract state and information from any particular node in the graph. This fact is very important to our study because we are interested in the features that the system craft to perfom the given task. We want to teach the system to excel at our task of interest and then use its knowledge to improve another task.

Another feature that TensorFlow provides is its ability to prune the execution graph before starting its computations. Usually several sets operations are repeated along the graph, by detecting this, the system can automatically replace all incidences of each repeated sub-graph with a single one thus, saving memory and time. The same case happens with communication nodes. If several nodes from a single device are consuming the same data from another device, the system is prepared to detect this and ensure that the data transfer occur only once.

The framework code is deeply optimized. Implementations of the same interface target the different dispositives that the code can run in. It is built upon known mature frameworks such as cuDNN, a library for deep neural networks that targets NVIDIA GPUs, and Eigen, a C++ library for linear algebra, which was extended to offer tensor arithmetic support. On top of this infrastructure, TensorFlow offers a Python client which is very convenient for fast development.

An interesting tool that is packed with the framework is TensorBoard. With the huge complexity that machine learning models offer at this scale, it is important to know what is happenining at any point of the training process. TensorBoard offers a glimpse of how does the architecture for a particualar computation graph looks like. This will become handy later when describing our model.

There exist several systems that offer similar features to TensorFlow. Theano, Torch, Caffe, Keras to name a few. The purpose of this work is by no means to study the advantages or disadvantages of these systems nor to create a benchmark on their perfomance. As the pipeline was already written in Python, we choose TensorFlow to minimize the gluing code. Among the examples that come with the system there is one end to end example of how to transfer learining from one trained net to another task.

Chapter 3

System architecture

> We may say, roughly, that a mathematical idea is 'significant' if it can be connectted, in a natural and illuminating way, with a large complex of other mathematical ideas.
>
> A Mathematician's Apology
> G. H. Hardy

To analyze a huge amount of images requires a better way of handling and sorting them than just storing them into directories. A set of scripts where built arround a database to let us create trainging and testing sets easilier. Borrowing ideas on how is the transfer learning process made by the engineers in the tensorflow team, our catalog system grew to answer to these needs. In a later stage the analysis process was also included into the system spawning what could become a framework to analyze visual patterns in sets of images with many different purposes.

The development process showed the need of a better set of tools to overcome several difficulties that were found. The training stage involved a repetitive process consisting of manually inspecting the available imagery. To this end, a web application was built on top of our catalog system facilitating this process. The process of continuously showing and tagging images proved to be error prone, each mistakenly tagged picture, required

to log into the database and delete the wrong entry after maching encoded file names. So a new feature was implemented into our tool, it shows the stored images and their respective tag and lets the user either delete or edit the classification. Finally, when the model was already trained and it was predicting on the orthorectified rasters, the ability to geolocate the predicted damaged areas and place them in a map was very convenient.

As it was just exposed, the application was built thinking about the end user. Even though the correct implementation from a design perspective is out of the scope of this work, it was an interesting thing to explore as this would be the kind of problems to be solved in the case of getting such a system into production. The last few paragraphs explained how the system grew to a full server-client architecture to respond to the need of processing data and visualizing the results in a meaningful way.

The purpose of this chapter, is to talk about the implementation of our experiment. Details of the pipeline architecture, and the techniques used to obtain and curate data are unveiled.

3.1 Backend

To understand why some design decision where made we need to understand the nature of our data. Data came in directories taken from the drones and splitted by flying dates. The quantity and naming of the images was not uniform across towns. Additionaly, drone imagery provide metadata that is useful to geolocate the images. However, this information is limited as it only offers the place where the image was taken but gives no information about the image resolution. This difficulty is overcome with specialized software that takes in the set of images and creates a mosaic using the images using a process known as orthorectification which corrects the distortions caused by the angle in which the image was taken. CENAPRED gave us both resources, the raw drone images and the ortorectified rasters. So the first part of the application was to transform these data into a way that would be easier to manage.

Given to previous experience in developing similar projects, it was decided that the system was to be built on top of a Python Web framework. It offers many solutions out of the box, including a familiar line interface set of commands and an object relational mapper that makes the database integration easier. The feature of being ready to offer a web interface was a great plus.

3.1.1 Data model

We took advantage of the object relation mapper system that Django offers. In the figure [**?**] we show a subset of the database diagram. Tables inherent to the features of the Django framework are not shown as they where not modified.

The database design was based on reproducible reaseach. We wanted to keep track of the charactistics of the models trained. Also we wanted to know which where the training images used for each model. It was also desirable to be able to reuse models that where trained with different sets of images to benchmark and finally use the best model to actually predict on the orthorectified raster.

In order for the tagging application to work, images must be ingested into the system so every time a new sample is tagged we populate the database with information about the original image and the coordinates relative to the original image. In the final stage of the process we produce a list of potential damaged buildings which are also inserted into the database with geografical information and human readable address. We think that this can be helpfull to allocate resources in the most efficient way.

We built a system to ingest the images from the NOAA service. It lazily downloads the images by checking first if the file is already present in the temporary folder. If the file does not exist it downloads it, then the system tries to add it to the database and persistent file system. To maintain a coherent one to one mapping between the database and the file system, the process of adding a new scene must be successful both in the database and in the filesystem, otherwise, the file is erased from both, and the state of the system remains as it was before the ingestion attempt.

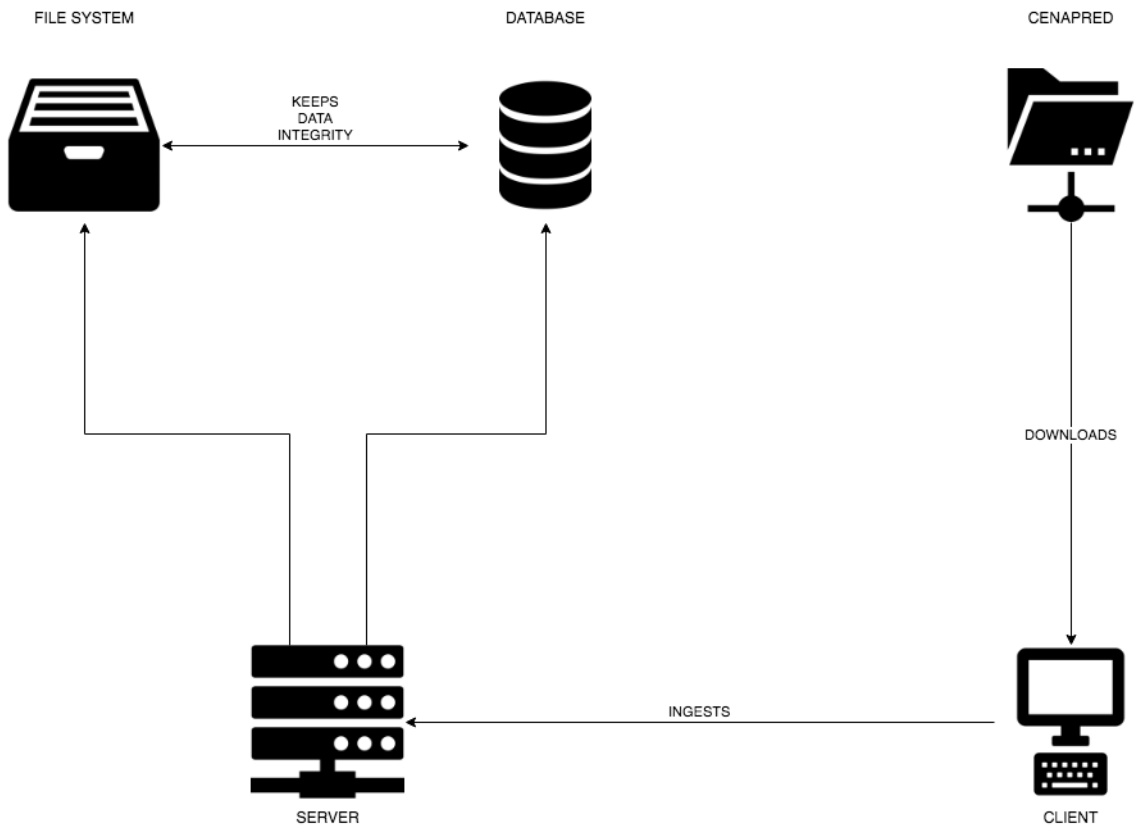


Figure 3.1: We download images from a ftp that CENPRED gave us access to.

3.1.2 Tag

Aerial tagged data is scarce. In particular, for the purpose of our experiment, we don't have any useful metadata on the images. We propose a method to tag samples of the scenes using crowd sourcing. We built a service that crops samples from the images and exposes them to an online application that lets any user with access to tag an image. We have three categories: the image has water in it, the image does not have water in it, and it is not possible to tell. When a positive answer is obtained, the system persists de image in the data base with the information of from which scene was it extracted.

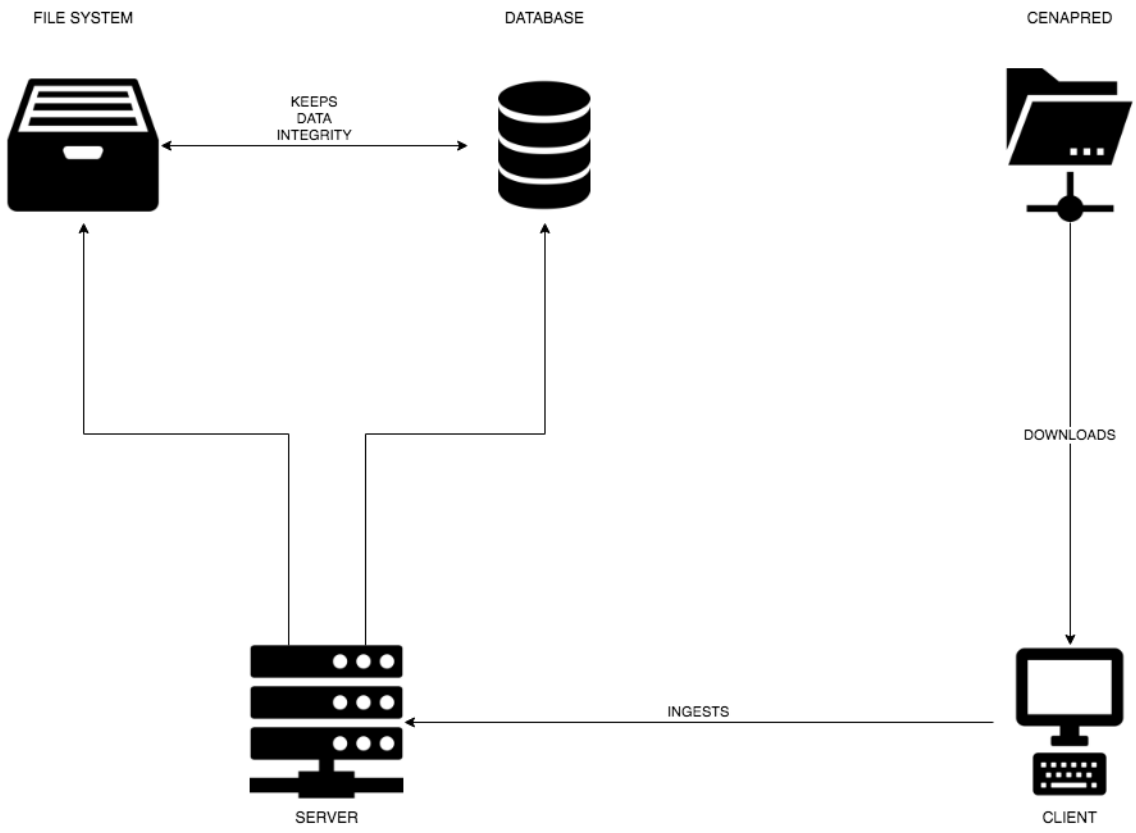


Figure 3.2: Ingest process.

3.2 Data augmentation

Given the nature of our task, it is hard to acquire the tagged images. To increment the size of our training dataset, even more, we use a technique known as data augmentation. It relies on the fact that affine transformations do not change the content of the scenes, however, a transformed scene appears as a completely new one to the classifier.

The images were rotated by 10 degrees, and reflected by the x-axis and the y-axis this gives us a $x144$ factor, this means that for each tagged image, the training corpus is incremented by 144 images. The problem with this approach is that when a square image is rotated, some information on the corners is lost so we have to adjust the original image so that we can still crop a complete square from the desired size from it. For our experiment, the input size for the neural network is 227×227 pixels, so the original images must be at least $\sqrt{2}$ times 227 on each side. This way no matter how we rotate

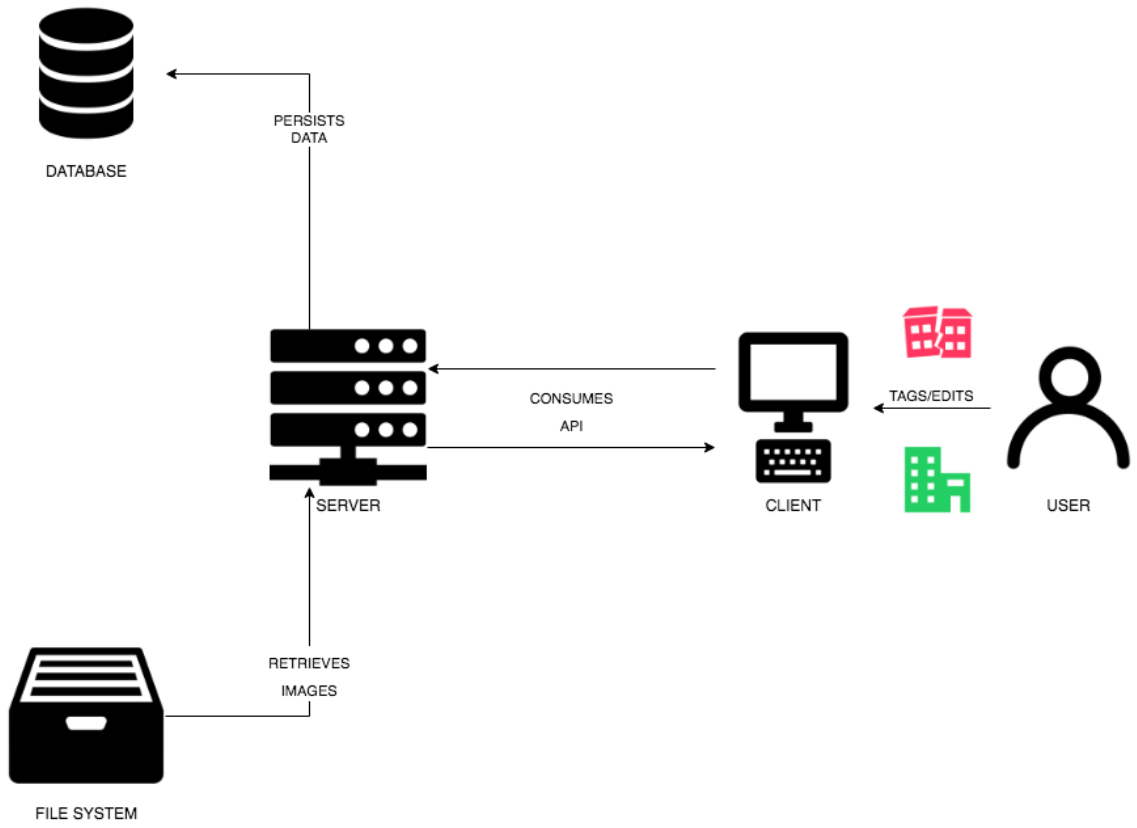


Figure 3.3: Tagging process.

the original image, we can still crop a 227×227 from the center of the rotation without losing any data.

3.3 Train

We needed data to train our model. Raw dron images where obtained from the National Center for Disaster Prevention (CENAPRED). Drons flew over three towns in the state of Oaxaca producing 3733 images during several days. Images contain gps information about the place where they where taken, however, it is not posible to produce a one to one mapping from the pictures to georeferenced points. With this limitation it was not possible to locate possible damaged buildings from the raw images. However, the model does not need any geografical information to be trained as it relies only in the pixel intensities.

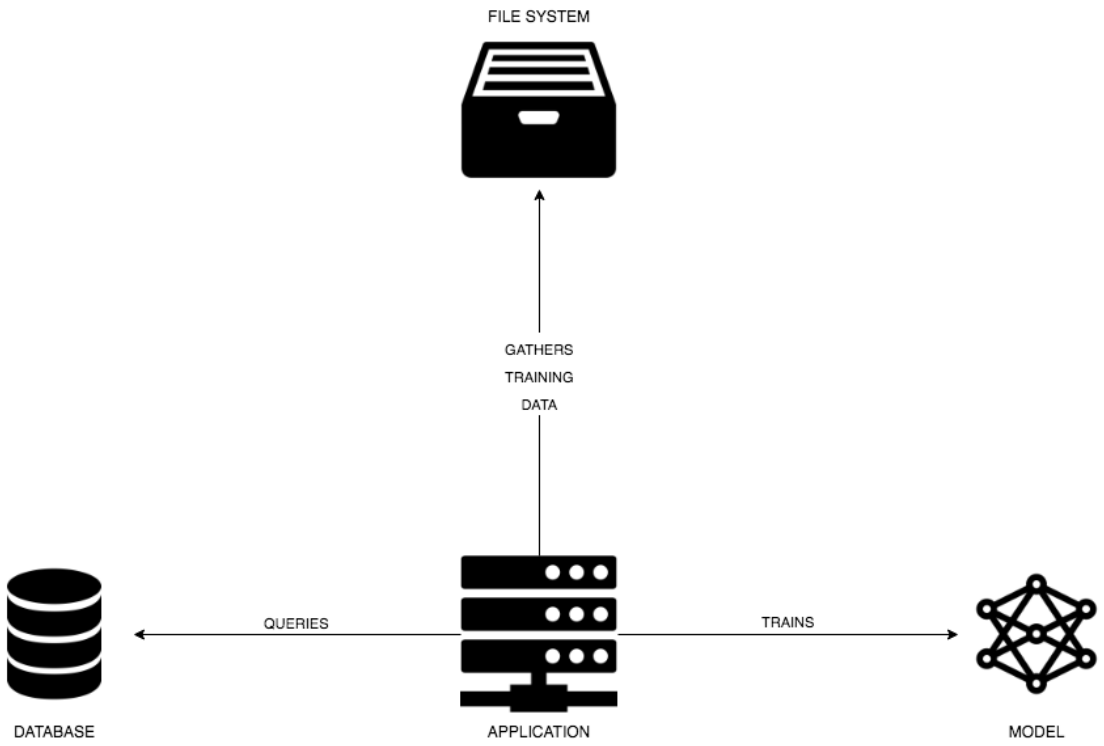


Figure 3.4: Tagging process.

Cropping and tagging manualy a large number of samples from the images was prohibitive. In order to overcome this obstacle an online tagging application was developed for this purpose. The idea was to decentralize the tagging procedure by giving an easy to use tool that was able to run from any browser. This way the cumbersome task of tagging the images can be crowdsourced.

By consuming the REST api described in the previous section, a simple web client was developed using jquery and openlayers. The interface is an image viewer with a selection and a button to submit an opition on that the higlighted area. The user will select an appropiate section of the given image, tag it, and then submit the section. In the backend, the image is cropped and ingested into the database.

An additional button is given that lets the user see how well is the current best model performing. The process is quite similar to the previous one. In the server, the image is cropped and then the thumbnail is exposed to the model and the result is written back to the client through the REST api.

During the process, we noticed that the tagging process can lead to errors. In order

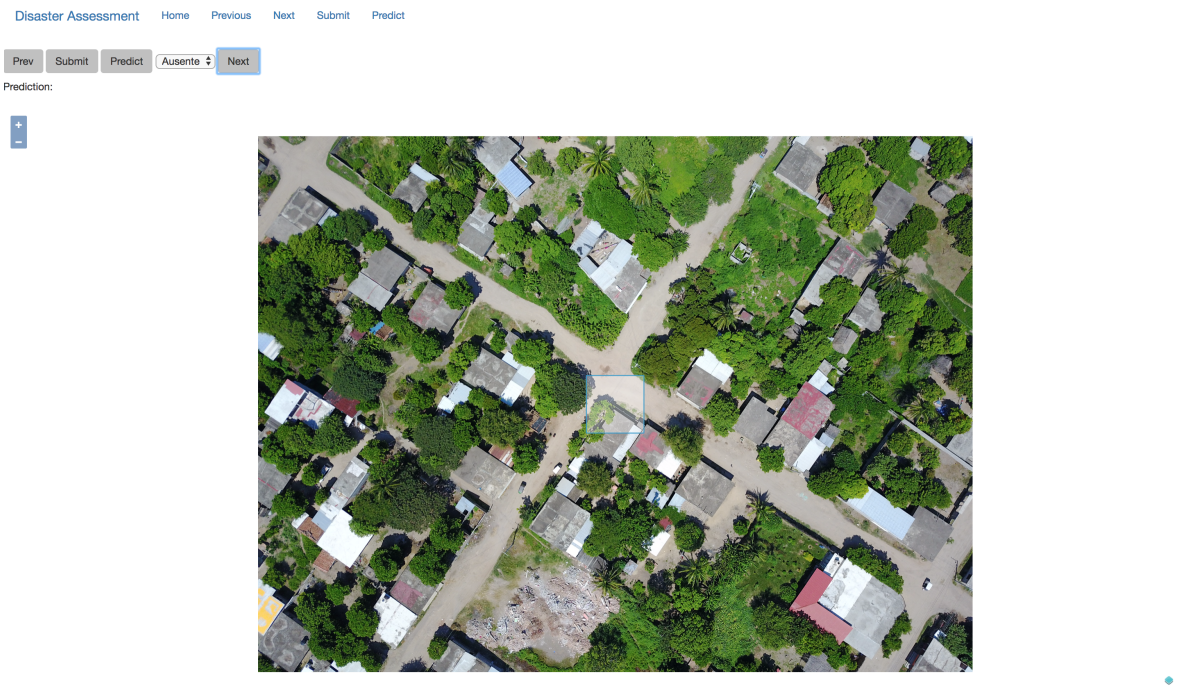


Figure 3.5: Visualisation process.

to deal with those mistaken tagged images, a visualiser for the tagged images was developed. It consist in a simple interface where the last tagged images are shown with their respective tag. The interface offers a way to delete a given image or just edit the previously assigned tag.

Finally, we developed a view for the points that the algorithm finds in the ortorectified image. This view was implemented with open layers and shows information about the location of the potential damaged building in a human readable form. This is done by querying the google api with the latitude and longitude points extracted from the map.

3.4 Predict

Another simple front end application was developed to predict on new features, it is very similar to the tagging application. Instead of asking the user about the correct tag, it queries the model and exposes the answer to the front end.

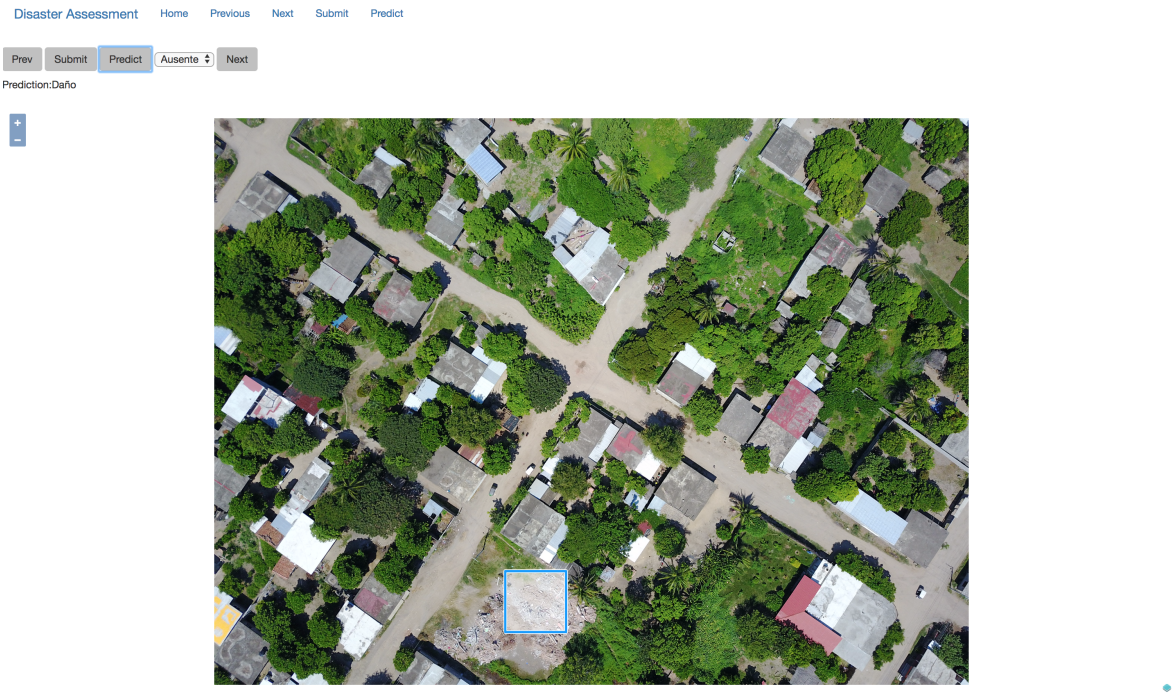


Figure 3.6: Predict interface.

3.5 Model creation

In order to create the model we need to select a sample from the thumbnails that we extracted from the original images. We wanted to create a process that was easily reproductible so we would compare models in a simple fashion. To this end the sample must be random each run and we need to split the images in three sets: training, validation and testing.

Tensorflow provides a script to retrain the last layer of inception by conecting the extracted features into a sofmax layer, and then training this classifier on the given set. It requires a directory layout tailored built to this purpose. The script was modified to fit our database design in order to make as easy as possible to train several models with homogeneous training, validation, and testing sets.

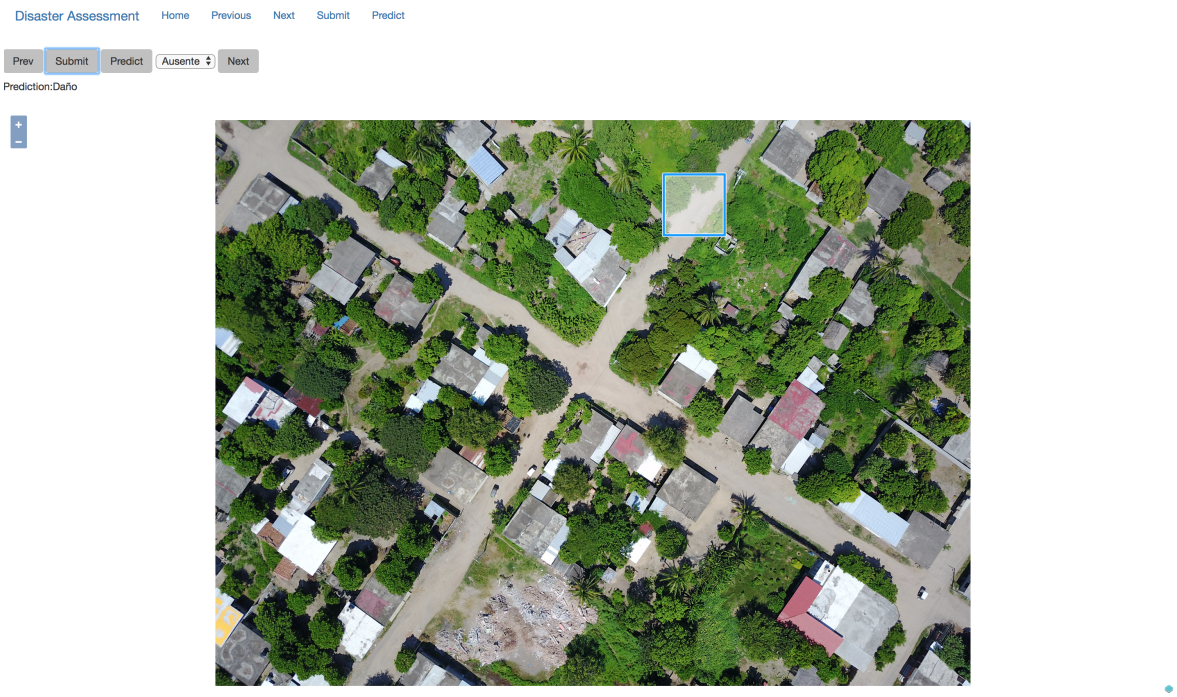


Figure 3.7: Submit interface.

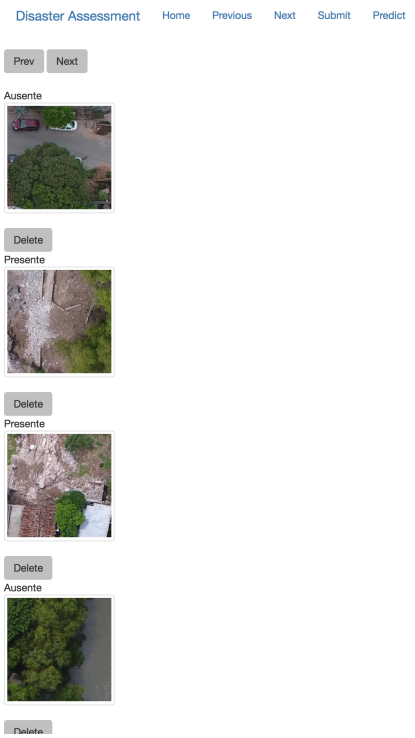


Figure 3.8: Edit interface.

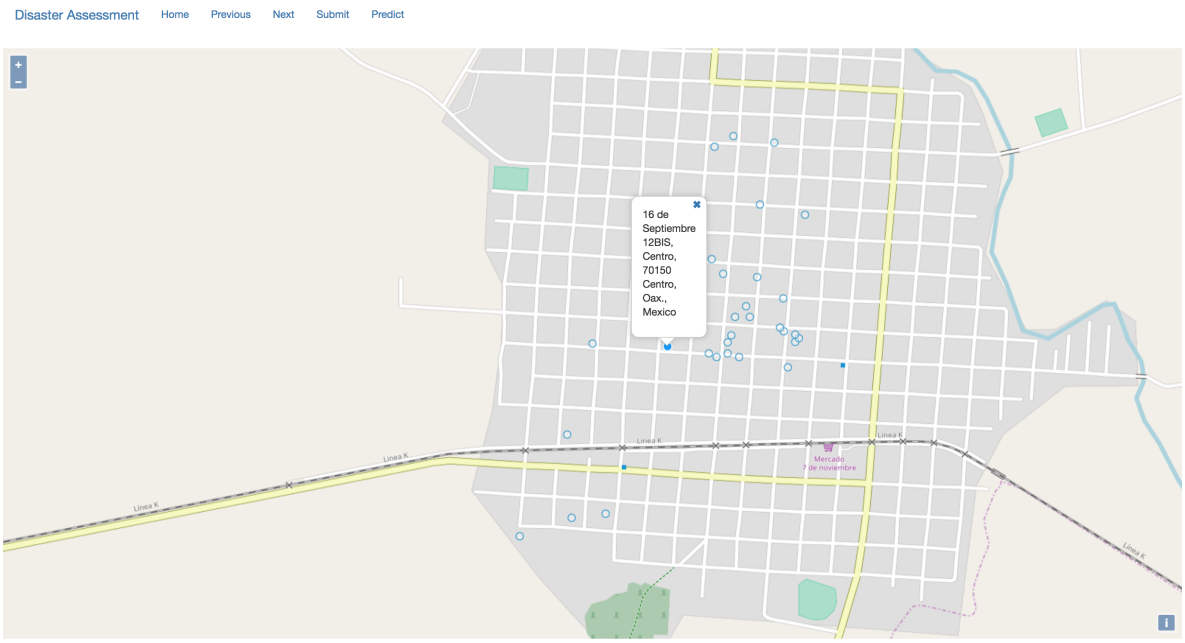


Figure 3.9: Predict interface.

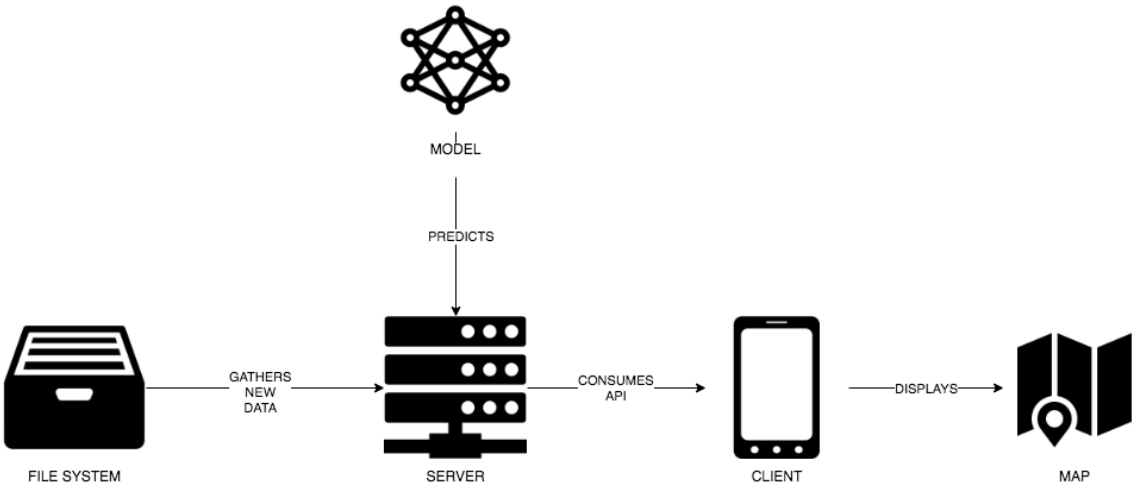


Figure 3.10: The system is prepared to predict on new data and display a map with the places in which it founds damaged buildings.

Chapter 4

Data analysis

> A mathematical proof should resemble a simple and clear-cut constellation, not a scattered cluster in the Milky Way.
>
> A Mathematician's Apology
> G. H. Hardy

The process that our experiment undertook was iterative. Obstacles in the analysis showed the need to develop tools that helped us to make the process less cumbersome. In the same way that in the previous chapter we explored the flow of ideas that led to our pipeline design, here we will explain how data shaped our experimental process.

The purpose of training a model, is to provide a faithful representation of the studied phenomena, but how do we know that this description is correct? We need to validate the relationship between the outcome of the model and the data that we have at hand. This process is complicated. Validation must be carefully crafted to obtain valid results.

To have a complete picture of how does our approach compares with techniques from classical computer vision, we trained other models to test their performance against our novel approach. This exercise gave us some insight on the feasibility of a real world implementation.

4.1 Exploratory analysis

During the week following the Chiapas earthquake of September 7, 2017, several drones captured images from three different towns in the state of Oaxaca. We obtained those images from CENAPRED. The ensemble consisted of 1134 images from Juchitán de Zaragoza (Figure 5.3), 727 images from Santa María Xadani (Figure 5.4), and 1872 images from Unión Hidalgo (Figure 5.5).

As we can see from Figures 5.3, 5.4, and 5.5, drones fly in regular patterns forming a lattice of points, these clouds of information distribute spatially around the towns of interest. We wanted to show that this clustering translates to the space of information that the images contain. To do this, we used a technique known as t-Distributed Stochastic Neighbor Embedding (t-SNE), introduced by Laurens van der Maaten [**?**], as an alternative to traditional methods that were difficult to optimize given the computational costs.

The interest of being able to represent the images in a two-dimensional structure comes from the idea of simplifying things for better understanding. High dimensional data usually lies in hidden low dimensional manifolds, techniques like t-SNE gives us insights about this underlying structure. The reason that we chose this method instead of a Principal Component Analysis was that t-SNE is particularly well suited for embedding a high dimensional space into a two dimensional one.

We expected the images of the different towns to cluster in this low dimensional representation. We thought that pictures taken under in the same town would be closer than the ones taken in a different setting because of the light conditions during the exposure tend to have less variance among similar times and places.

One limitation that we found was that the size of the images[1] prevented us from computing the t-SNE for them. To overcome this problem, the information from the pixels of each image was flattened into a vector comprising the means and standard

[1]Raw images are 4000×3000 pixels, and three color channels each.

Figure 4.1: Juchitán de Zaragoza.

deviations for each color channel. This simple dimensionality reduction technique allowed us to use the aforementioned technique. As we expected, images form natural clusters depending on the town that they depict. We show our results in Figure 4.4. This outcome adds to evidence that supports a methodology of using images from one town and try to predict on others to see how well does the model generalizes.

Figure 4.2: Santa María Xadani.

Figure 4.3: Unión Hidalgo.

4.2 Experiment

We needed a benchmark to test the effectiveness of our method. To this end, we used classic computer vision methods on our dataset. We extracted simple features from

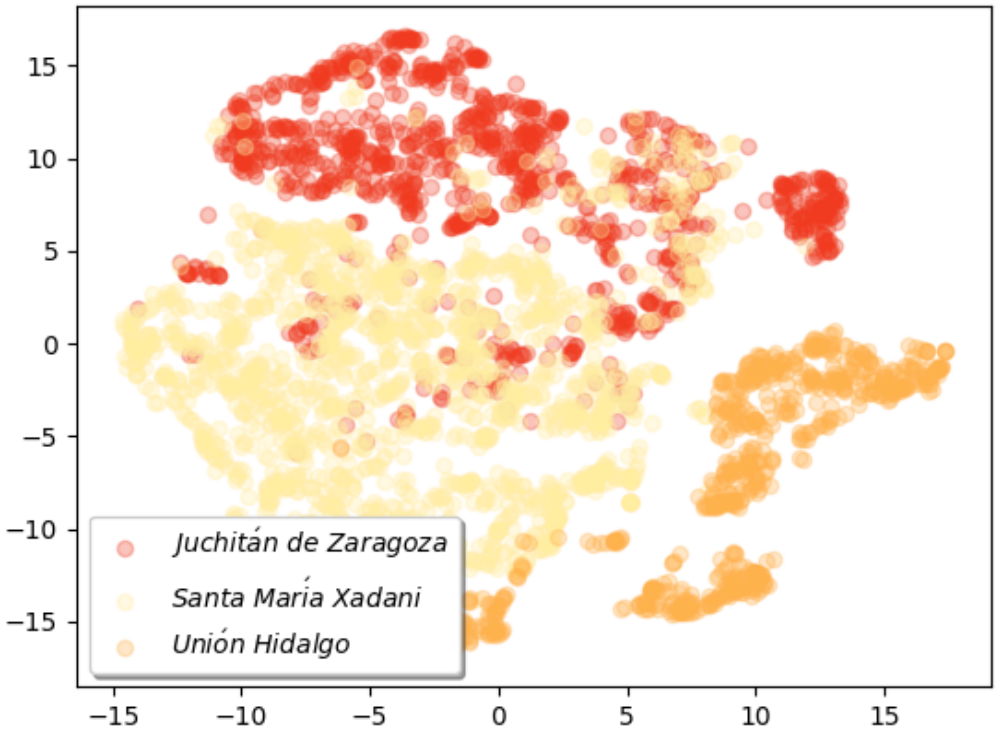


Figure 4.4: T-sne diagram.

the images, and then we trained a classification model on the resulting data[2]. The final purpose was to show how the methods compare out of the box with little extra effort.

We considered three types of features. First, we used a method known as Histogram of Oriented Gradients (HOG), introduced by William T. Freeman and Michal Roth [**?**]. It consists of studying the slopes of the intensity of the pixels in a dense grid of the image. The feature space that this method output depends on the size of the lattice. Then, to test more straightforward features, we extracted the means and the standard deviations from each of the color channels in the same fashion that we did with t-SNE in the exploratory analysis. We additionally decided to consider the color channel means alone as a different feature set. Finally, we trained a random forest for

[2]Spending more time on feature engineering on the pictures would undoubtedly lead to better results. However, it was not the scope of the present work.

each of these three sets of attributes.

Figure 4.5: Sample images from the damaged training set.

We reduced the problem to binary classification by selecting only two labels; a *damage* tag when the samples showed clear infrastructure damage, and a *no damage* tag when we saw no visible damage. We show some examples for both categories in Figures 4.5 and 4.6. We used the application detailed in the previous chapter to crop and classify 200 square patches from the images in each of the towns. To keep the classes balanced, we chose 100 images for each class. Every sample was 327×327 pixels, and we assigned a tag manually for each of them. After the tagging process, we had 600 labeled images that we will refer to as *our dataset* through the rest of this chapter. The information was stored in the relational database and reviewed for data consistency.

As mentioned earlier, we studied three different towns, so we assembled different datasets using a different permutation each time. In the case of the classic computer vision methods, we used two towns to train the model and the remaining one to test the trained model. In the case of the transfer learning model, we used the images of one town to retrain the model, the images of another town to validate the model during the training stage, and the remaining town to test the trained model. In all cases, we tested every single image in the testing town against the model.

Figure 4.6: Sample images from the non damaged training set.

One of the difficulties that we found during our research process was that several images might depict the same collapsed building. This event could happen when the drone flew over the same territory in different routes. During the tagging stage, we inspected the images for damaged areas, however, despite our efforts, it was impossible to obtain a set of pictures in which every single one was completely independent of the rest. This fact shaped our validation process.

A classical approach would be to divide the dataset into two parts; training and testing sets, the model is trained with the former, and then tested against the latter. This method is not used in practice because data is usually scarce and we would like to take the most of it. A better option is n-fold crossed-validation in which we repeat a similar approach n times and average the outcomes. This process makes more likely to train the model with every data point and average out errors that may appear by chance. However, n-fold crossed-validation does not suit our case because of the nature of our dataset. As a consequence of the patterns in which the drones overfly the area, some buildings appear several times in the dataset even though the original pictures were not the same. If we apply a technique such as cross-validation, it would be very likely to have the same building both in training and in the testing sets. This situation

would lead to having very high accuracies for our model but not such good performance in a real setting.

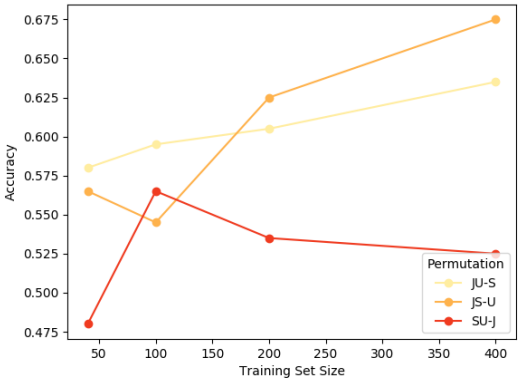
Another thing to be considered is that while traditional supervised classification methods need to divide the dataset into two parts to perform the validation, machine learning methods often require three sets instead, namely train, validation, and test. The validation set is used during the training stage to fine tune the hyperparameters of the model, and then the testing set is used to see how our model deals with previously unseen data. We manage to elegantly avoid the problem of having repeated data on each of the datasets by using the fact of having information from three completely independent settings. We trained models feeding them with different training set sizes and testing them against a set of a fixed size. In summary, we trained a model for each combination of a method, a training set size, and town permutation.

4.2.1 Classic Computer Vision

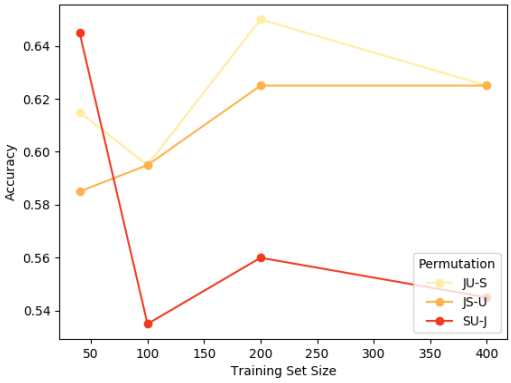
We extracted the HOG features using an implementation from the Python package scikit-image [**?**]. We used eight orientations, and a 16×16 pixel window. Each town was selected once for testing, in table 4.2.1, we show the codes that we use in Figure 4.7 to show the results that we obtained.

Train Set	Test Set	Code
Juchitán de Zaragoza, Unión Hidalgo	Santa María Xadani	JU-S
Juchitán de Zaragoza, Santa María Xadani	Unión Hidalgo	JS-U
Santa María Xadani, Unión Hidalgo	Juchitán de Zaragoza	SU-J

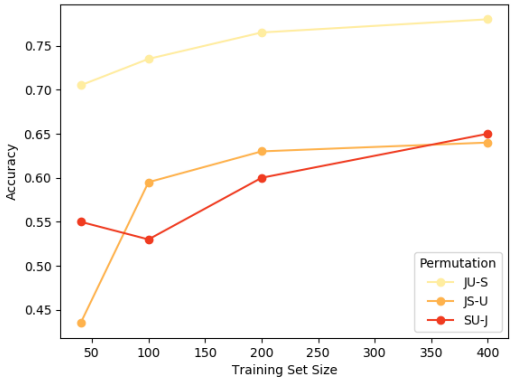
It was surprising that the model that showed the best performance was the one in which we considered means and standard deviations. We expected that the specialized features of the HOG model to perform better than a simple mean and standard deviation extraction. We think that the reason behind this was that the HOG features were designed to find objects in images, but they are not well suited for aerial images and the textures that characterize debris. As we expected, increasing the size of the training set results in improving the performance of the trained model. In particular, we



(a) Histogram of Oriented Gradients



(b) Means



(c) Means + Standard Deviation

Figure 4.7: Training example using classic computer vision models.

noticed that the worse performing models were the ones using Juchitán de Zaragoza as the training set, a hypothesis is that the pictures from Santa María Xadani and Unión Hidalgo do not generalize so well. The best performing model achieved 78% accuracy and used Santa María Xadani as the testing set and all of the remaining 400 images to train. The worse performing model achieved 43.5% and used Unión Hidalgo to test with only 40 images in the training set which is particularly unfortunate considering we are dealing with binary classifiers [3].

[3] A binary classifier which scores less than 50% accuracy is worse than flipping a coin and using that flip to decide the class.

4.2.2 Transfer Learning

In the case of transfer learning, we have three types of sets and three towns. This fact makes six different possible configurations. Thus six different models for each training set size. We retrained the Inception model using our dataset and 1000 training steps[4].

Train Set	Validate Set	Test Set	Code
Unión Hidalgo	Juchitán de Zaragoza	Santa María Xadani	U-J-S
Unión Hidalgo	Santa María Xadani	Juchitán de Zaragoza	U-S-J
Juchitán de Zaragoza	Unión Hidalgo	Santa María Xadani	J-U-S
Juchitán de Zaragoza	Santa María Xadani	Unión Hidalgo	J-S-U
Santa María Xadani	Unión Hidalgo	Juchitán de Zaragoza	S-U-J
Santa María Xadani	Juchitán de Zaragoza	Unión Hidalgo	S-J-U

4.2.3 Threshold Selection

A binary classifier assigns a real number in the interval $[0, 1]$. To decide which tag to assign to each prediction we must choose a decision threshold. A tool that is often used to select such threshold is a Receiver Operating Characteristic curve (ROC curve). It was first proposed during World War II to measure the ability of a receiver operator to detect an enemy ship, where it took its name [**?**].

When dealing with a binary classifier, we want to predict a condition p. If the outcome of our classification is p and the real value is also p we call that a true positive (TP), if the actual value is instead n, we call that a false positive. Conversely, we assign names for a true negative (TN) and a false negative when we predict n as the outcome for our classifier. We call true positive rate (TPR) or *sensitivity* to the ratio between the outcomes correctly predicted as p and the number of elements that are indeed p. We are also interested in the false positive rate (FPR) or *specificity*, which is the proportion of incorrectly assigning p given that the condition is n. In a ROC curve, we plot the sensitivity as a function of the specificity; it is used to select a decision threshold and have a tradeoff between detecting as many positives and having as little

[4]We empirically noticed that further increase in the number of training steps did not improve the final accuracy of the model.

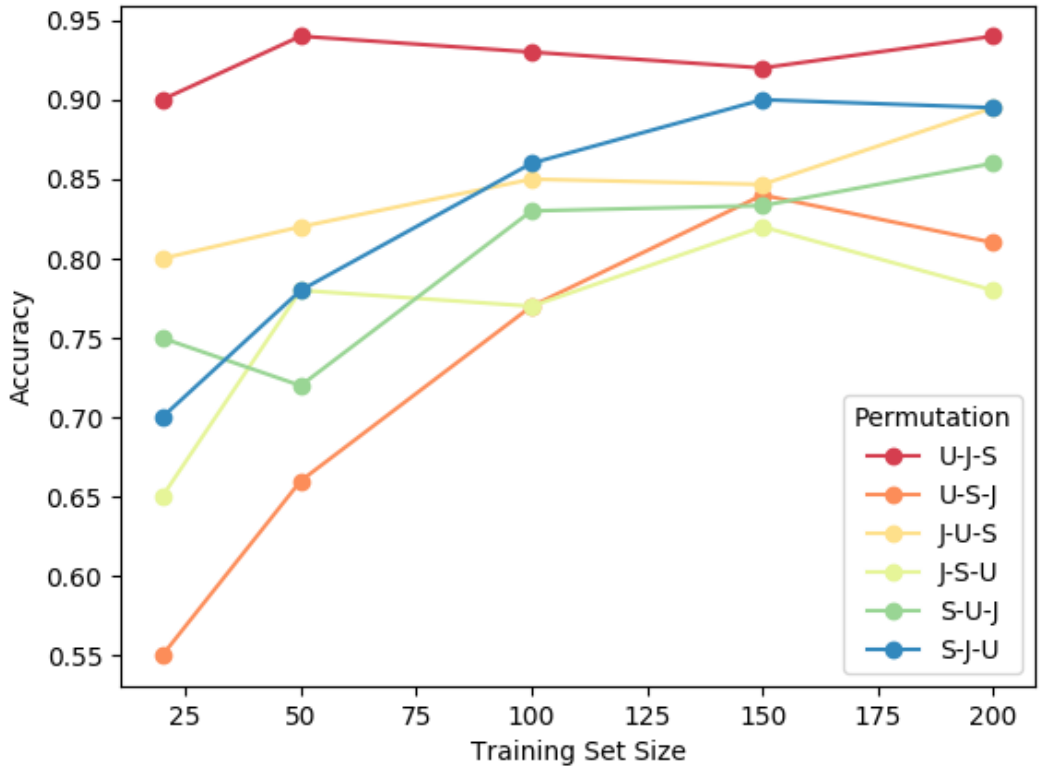
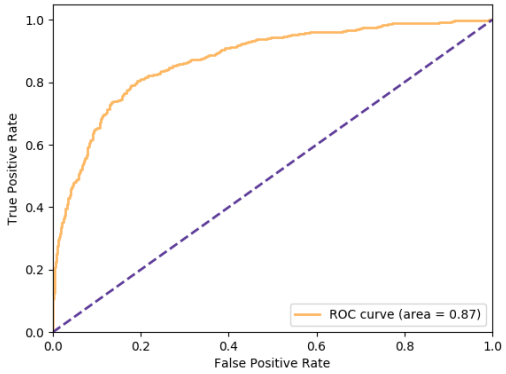


Figure 4.8: Results of our experiment. As we expected, the graph shows a positive correlation between the accuracy and the number of training samples.

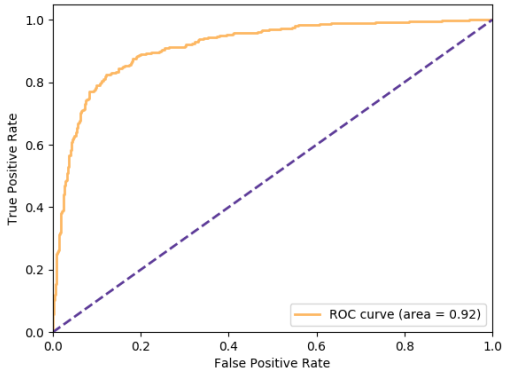
false alarms as possible. There is an additional feature of interest, the area under the ROC curve (AUC) is often used for model comparison, the closer the value is to 1 the better.

There is no correct way to select a decision threshold. It depends on the application at hand. For instance, when we are dealing with disease diagnosis, a screening test should be very sensitive whereas a confirmatory test should be very specific. In our case we want our classifier to be very sensitive because there is not as much overhead to have false alarms. In Figure 4.9 we show the ROC curves for each of the transfer learning models, to build them, the prediction the test images were recorded in each permutation among the different training set sizes. Therefore, we have a ROC curve

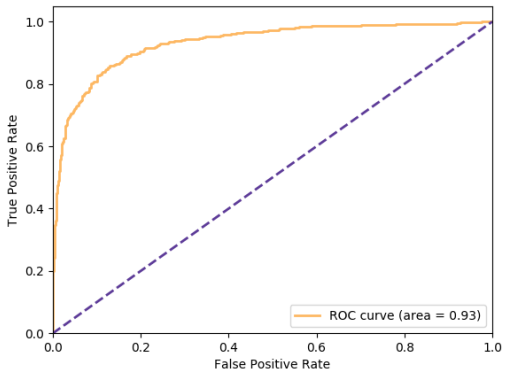
for every training set size. As we can see, the area under the curve is very close to 1 in every case, and it increases as the training set size increases.



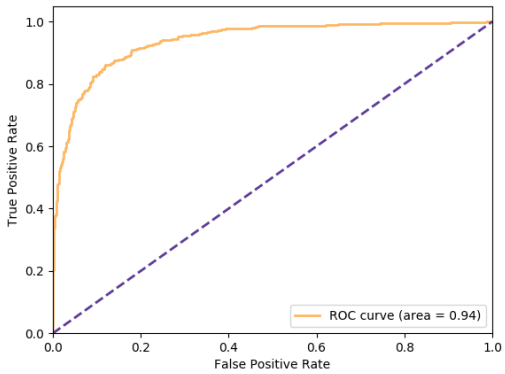
(a) Training Set Size: 20 Images



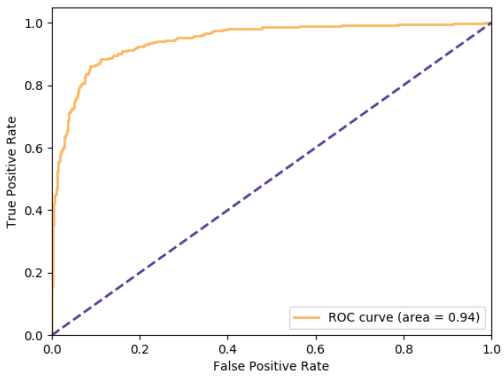
(b) Training Set Size: 50 Images



(c) Training Set Size: 100 Images



(d) Training Set Size: 150 Images



(e) Training Set Size: 200 Images

Figure 4.9: Receiver operating characteristic curves for the different training sets.

Chapter 5

Implementation

> A mathematical proof should resemble a simple and clear-cut constellation, not a scattered cluster in the Milky Way.
>
> A Mathematician's Apology
> G. H. Hardy

Even though deep learning has allowed us to do a giant leap forward on the performance of several tasks, and contrary to what some researchers claim [**?**], there exists no universal model. A good model would be able to be performant in very different settings, but it won't be able to handle every situation. We need our model to be robust, but we can not expect it to be perfect. Being aware of the limitations intrinsic to the model is the best way to alleviate possible mistakes. Although it sometimes seems the case, deep learning is not a magic box, it learns from the data that we supply. If the model is biased, in all likelihood, it means that our training set is biased. In this chapter, we briefly discuss the details of implementing our method to be used in real-world setting.

5.1 Final Model

In the previous chapter we trained and validated several models. We found that our methodology is quite effective in the task of locating damaged buildins. We now centered our efforts in creating a final model taking into account the entire dataset of 600 tagged samples. This model was not validated as there was no images left to use as testing set. This would closely resemble a real pipeline.

We used 0.95 as our decision threshold. According to the best of our models, this threshold guaratees us that we would have around a 0.47 probability assigning a damaged label when the building is damaged while only having a 0.01 probability of misslabeling a non damaged building. While this decision threshold was not necessarily valid for our final model, it gave us a hint on what decisin to make. We are sacrificing sensibility for specificity, when we predict a damage building we are confident that it will be a true positive. We could argue about the low probability of assigning a damage label but it all makes sense if we thing that we test a lot of samples for each zone.

5.2 Post-processing

So far we have centered our discussion on training a model that is able to categorize images of a particular size. Using such a model to detect collapsed buildings from the drone imagery and geolocating our findings is a completely different story. We must prost-process the information given by the model in order to tranform it into insights.

5.2.1 Overlapping

The usual methos is to use an slideing window. In this method we take a fixed window of the size that the model accepts, and we croped parts of the raster. En each step we use the model to predict on the current sample. When a prediction is positive, we save the location of the window. After the algorithm went through the whole image, we used the found positives and drew boxes over the original image. We can look at the result of this process in Figure **??**. It is evident that this approch has inherent flaws, as damaged buildings can be counted several times by the algorithm. A method to aliviate overlapping bounding boxed is known as non-maximum suppression (NMS).

It was first proposed by A. Rosenfeld and M. Thurston in the context of edge detection techniques [**?**]. It consist of checking the percentage of overlap in the set of bounding boxes and discarding those that are repeated. The final result after postprocessing looks like Figure **??**.

5.2.2 Orthorectification

Ortorectified mosaics where built by CENAPRED using the very same images taht we used to train our models. This images come in a different format than the images taken by the drones. Additional to the optical information these tif files contain geographical location and can be used to assing a point in space to each pixel in the image. This means that we can not only locate a damaged building in an image, but to link this information with a geographical location in a given projection.

5.3 Final Results

The images where divided in a regular grid of 322×322 pixels tiles with 90 pixel ovelaps. These overlaps are later postprocessed to eliminate the posibility of counting the same building twice. Each tile is exposed to the model which predicts a class on it using the previously selected threshold. When the model test a tile positive, the box is saved for postprocesing in which a technique known as non max suppression is used to eliminate boxes that represent the same object. This technique is borrowed from facial recognition algorithms. Once we have the final boxes, the center pixel of each box is transformed to world coordinates. Additionaly, this coordinates are used to query Google Maps API to obtain a human readable address for each point.

A shape file is produced which contains the results for each town given the output of the algorithm. This shape can be overlayed on top of the raster file using a Geographic Information System software such as QGIS. Additionaly the results are also exposed via the REST interface so they can be visualised in the web application.

Figure 5.1: Results of our experiment. As we expected, the graph shows a positive correlation between the accuracy and the number of training samples.

Town	Threshold	Positives	Width	Height	Seconds	Overlap	Windows
Santa Maria Xadani	0.96	11	25598	30144	16403	0.0005	45704
Juchitan de Zaragoza	0.96	562	42375	28831	21216	0.0005	72372
Union Hidalgo	0.96	73	19945	28795	11363	0.0005	34188

Table 5.1: We report details on running the algorithm on the orthorectified rasters. Width and height are in pixels. Threshold is our decision boundry to decide wether or not a window is considered to have damage. Windows are the number of predictions that the model gave us. We also report the time in seconds that each process took.

Figure 5.2: Results of our experiment. As we expected, the graph shows a positive correlation between the accuracy and the number of training samples.

Figure 5.3: Juchitán de Zaragoza.

Figure 5.4: Santa María Xadani.

Figure 5.5: Unión Hidalgo.

Chapter 6

A glimpse into the future

> A mathematican, like a painter or a poet, is a maker of patterns. The mathematician's patterns, like painter's or the poet's, must be *beautiful*; the ideas, like the colours or the words, must fit together in a harmonious way.
>
> A Mathematician's Apology
> G. H. Hardy

The product that we developed during this research might serve as a baseline in the event of new natural disaster events. It works s a tool to give an idea of where to start looking for damaged buildings so decision makers can allocate resources efficiently.

We built a data pipeline from end to end. This exercise was enlightening and showed many of the issues that can appear when we take such endeavor. In this chapter, we review the ideas that we did not inspect deeper, and some obstacles found in the process.

6.1 Drawbacks

We noticed that, even when the classifier performs well in environments different to the one we trained it with, it learns to classify dribble, not precisely damaged buildings. We noticed some examples in which the classifier correctly finds scenes with the presence of dribble, however, when we inspected the place using Google Street View, we saw that there was no building in that area. This evidence can point to two possible scenarios; there was never a building in that place, and the site was used as a disposal for dribble from other areas, or the house was not there when Google Street View took the picture. Both cases show inherent limitations of the methodology that we are proposing; we can only automate this kind of process to a certain extent. This fact reflects concerns that we already exposed in previous chapters; we can not expect an algorithm to be perfect.

One aspect of the proceeding that we did not include in this research was the post-processing of drone data. CENAPRED gave us a post-processed raster with the geo-rectification already applied. This process requires specialized software that reads the images and geolocates ground control points that act as anchors for the pictures to be geolocated. This technique requires human experts to aid the software by manually inspecting the limitations of the automatized process.

Another limitation found in the process was the lack of reliable training data. This obstacle was the reason that the author needed to tag the images. In the future, we expect to have curated data from previous events to make the process even faster. As of today, our work may serve as a base for future improvements that allow this tool to get into production.

We should also take ethical considerations into account. Does the information that the system outputs should be public? What if people use it for nefarious purposes such as looting? With the power of data follows a greater responsibility.

6.2 Future work

We divided this section in two: first, we mention which features can be improved and incorporated into our pipeline, and then we image how can these ideas impact other fields of interest. By no means, this review pretends to be exhaustive, but just to give a broader picture.

6.2.1 Areas of opportunity

An idea that was beyond the scope of this research was to incorporate a technique known as active learning. In this scheme, the algorithm keeps improving as it receives feedback from the experts. In this fashion, when the model makes mistakes, this incorrectly labeled scenes can be relabeled and input back into the system to create a new model and keep improving its performance. Although there is a limit to the extent in which the algorithm can perform, this might aid in some prominent cases that might not be present in the original training data.

We used the Inception model because it gave us the scaffolding to produce an application minimizing the development efforts. It was the principal objective to test the ability of trained CNN models to be used in other tasks that the ones for what they were built. However, it would be interesting to create a model from the ground and fine tune it for our specific task. Inception is a complex model because of its nature; its classification capabilities lie far beyond the simple job for what we are using it. This means that the resources that it takes could be dramatically downsized. A tailor fit model would be more performant, but in the other end, it would also need vast amounts of training data.

The tool does not consider the different use cases that can come up in its real environment. Site verifiers would need to have access to the interface that allows to tag new training data in the same way that the taggers would need no access to administrative duties such as user creation and full access to the REST API.

6.2.2 Remote sensing

Another aspect that would be interesting to explore is the application of the same analysis framework in another type of studies. In the National Commission for the Knowledge and Use of Biodiversity (CONABIO), we use landcover maps to analyze and assess the evolution of the environment through time. We make this possible by leveraging standard classification algorithms and a significant amount of computing power. While our efforts have been quite productive, these algorithms have certain limits; they rely on the use of the light spectrum. As a consequence, any two categories with similar spectrum footprint will, in all likelihood, confuse the classifier.

For example, crops and grasslands might seem identical to a supervised classifier. Curiously enough, humans have little problem distinguish between such a pair of categories. The human eye can spot the difference from one another. This behavior is because we are not seeing particular pixels and trying to classify them one by one. Instead, our brain takes a look at the whole picture, and we focus on segments of the image that contain most of the information, in other words, we care about context. CNNs take this into account. Each neuron of the network receives only a zone of the image when information flow through the layers of the system, individual neurons activate upon specific stimuli. This process allows the network to recognize some features that would be invisible to a standard classifier. We can think of this as if the system engineers its features on the fly.

Efforts at CONABIO focus to attain a particular objective; to build a comprehensive biodiversity monitoring system. It can be though as two independent attempts. One is the Monitoring Activity Data for the Mexican REDD+ program (MADMex) [**?**] which pretends to monitor the behavior of forest and vegetation across the country by processing satellite imagery. The other is the Mexican National Biodiversity and Ecosystem Degradation Monitoring System (SNMB) [**?**] which gathers information about species in the different ecosystems that exist in Mexico. Both projects can be benefited by this novel techniques.

6.3 Conclusion

Our efforts showed that it is possible to deliver a preliminary product with little training effort. The performance that a fully trained CNN is outstanding compared to traditional methods. The accessibility of the pieces needed to build such a system makes it possible to create a system that helps in the aftermath of natural resources with little investment.

This subject is prominent nowadays, and the time has come to create tools that make our lives better. The objective of this work was to bring an efficient tool targeting the difficulties found in the real world. We thought of a device that allowed fast and efficient allocation of resources. Aiming places in which infrastructure is not as advanced as in big cities where we can employ other methods.

Recent events taught us essential lessons about what to do and what not to do in the aftermath. Earthquakes will continue to shake our realities. Our assignment is to prepare our society to confront these events most resiliently.