"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 "CUANDO LA TIERRA SE SACUDE, EVALUACIÓN AUTOMÁTICA DE DAÑOS CON BASE EN IMÁGENES AÉREAS", 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

Resumen

Los terremotos son un fenómeno impredecible y de inevitable naturaleza cuyo nivel de destrucción puede llegar a catastrófico. Debido a esto, buscar alternativas que ofrezcan una respuesta efectiva y rápida es una prioridad tanto para la óptima canalización de recursos como en el proceso de mitigación de las secuelas.

Tras el sismo del 19 de septiembre de 2017 en la Ciudad de México la colaboración masiva demostró ser un recurso fundamental para afrontar el desastre. El uso de las redes sociales permitió la sinergia entre labores de rescate, logísticas y las acciones de la sociedad civil. Sin embargo, ¿qué ocurre cuando no existen las condiciones ni la infraestructura tecnológica de las grandes urbes? Este trabajo propone un modelo aplicable a estos casos.

Durante los días posteriores al sismo del 7 de septiembre de 2017, con epicentro en el Golfo de Tehuantepec, drones del Centro Nacional para la Prevención de Desastres (CENAPRED) inspeccionaron las zonas afectadas sobre tres pueblos en el estado de Oaxaca. Las imágenes obtenidas fueron liberadas y sirven como objeto de estudio para esta investigación. Partiendo de una red neuronal convolucional entrenada con un banco de imágenes diseñado para la investigación en el campo de reconocimiento de objetos, se ajustó un modelo que detecta edificios dañados en las fotos aéreas. Con ayuda de técnicas de georectificación, el modelo predictivo se usó para generar un mapa de daños potenciales. Los resultados sugieren que este proceso podría servir para generar herramientas que ayuden a la correcta asignación de los recusos para minimizar costos y facilitar su operación.

Stay lucky

- To my late mom, and to my dad, the most amazing person I have ever met. I am so lucky to have you.
- To María José, for all the countless nights, the countable weeks, and the adventures yet to come.
- Thanks to the colorful bunch, Teresa and Julián, for the suggestions and good times.
- Andrea Garcia-Tapia, thanks for the idea for this project and for keep pushing me forward, I wish you all the best.
- Raúl Sierra, thanks for your valuable time and patience.
- Last but not least, to everyone involved in this project to some extent. In strictly random order: José Ramirez-Marquez, Everardo Robredo, Zaira Medoza, Oscar Zepeda, Guillermina Montanari, Camilo Abboud, Edgar Villeda, Ollin Langle, Loïc Dutrieux, Michael Schmidt, Fernando Esponda, Rodrigo Murillo, Luis Felipe González, Carlos Pérez.

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 computationally 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 problems such as damage assessment. In particular, we wanted to teach a neural network how to recognize damaged buildings in aerial imagery. We obtained imagery captured by drones after a massive earthquake stoke south of Mexico in September 2017.

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.

In Chapter 1 we explore our motivations. 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.

In Chapter 2 we elaborate an extensive literature review . 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 appli-

cations 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.

In Chapter 3 we discuss on the architecture of our pipeline. 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.

In Chapter 4 we examine the development and outcome of our experiment. We evaluate our model against classical computer vision techniques, and we propose a validation scheme for the model using the data at our hands.

Chapter 5 dives into the implementation in real-world setting. We use the trained model to predict on orthorectified areas. In the end, the predictions are geolocalised and transformed into human readable adresses, and persisted in a database.

Finally, in Chapter 6, we talk about future work, present our observations and discussion points and draw conclusions. We include several improvements that we can address to obtain better results.

This work was based on an internship spent on the Stevens Insitute of Technology (SIT) in Hoboken, New Jersey, during the summer of 2017. The project focused on flood detection usign drone imagery captured after Hurricane Sandy, under the supervision of Andrea García Tapia, and José Emmanuel Ramirez Marquez from the SIT. After the events of September, we adapted the pipeline to fit this circumstances. In this new context, I was advised by Raul Sierra Alcocer from the National Commission for the Knowledge and Use of Biodiversity (CONABIO).

Chapter 1

Introduction

> All palaces are temporary palaces.
>
> Robert Montgomery

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. This tragic event made us learn lessons about the scope and limitations of the association bewteen technology and people.

Mexico City is a large city, with good communications, and technological infrastructure. What happens when conditions and technological infrastruc-

ture 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 three towns in 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 Context

Due to its particular geographical conditions, Mexico is very prone to seismic activity [6]. 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 as can be seen in Figure 1.1. This characteristic has made the country suffer from many disasters along its history.

1.2.1 History

There has been a registry of these natural disasters since the Pre-Columbian age. The level of material damage and the death toll have been increasing as the population and the cities grow. In Figure 1.2 we can see a pictogram that according to [41] means "in the year 11 rabbit the earth trembled during the night".

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.

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

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 [12]. The fall of the iconic Angel of Independence was the result of an earthquake on July 28, 1957, 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 struck the city collapsing many buildings and leaving a death toll measured by thousands. The 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. Constant fear of quakes became part of Mexicans lives.

Two earthquakes took place during September 2017. While the second one

devastated Mexico City with a magnitude of 7.1[3], attracting help from all over the world, the first one was less known even though it was strongest earthquake to hit the country in the last century, its magnitude was estimated to be 8.2 [2]. Both events 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.

1.2.2 Cartography

Maps are flat representations of reality, and cartography is the study and practice of making them. In our contex we will use them to place the points of interest in a visual tool that help us with the resource allocation.

The use of cartography to map the damage information is useful to allocate resources in the aftermath of the disaster, and to serve as historical evidence. In figure 1.3 taken from [12] 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.

1.3 Objective

It is not coincidental that our brief historical summary focused mainly in Mexico City. Centralization has always been an inherent characteristic of Mexico. 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 resources better.

Data science arises from the need to process and filter large amounts of information. Without this organization, all this flow of data becomes just noise, transforming it into useful information is the work of data scientists. Using powerful mathematical tools implemented in state of the art technology, we propose a pipeline that lets us transform raw data in the form of drone imagery

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

into information useful to allocate the resources in a more efficient way during a crisis.

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 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.

The main focus of this project was to create a data product, and we went through the different stages involved in that process. We focused not only on the use of statistical tools to extract insights but also in the process of making this extraction reproducible. It is not only doing the analysis but also creating the tools to make the analysis easier to repeat. Additionally, we needed a baseline to test if our methodology performs better than other techniques. We trained and evaluated a method from classical computer vision, and compared it with ours. Although it was difficult to analyze both methods fairly it gave us an idea of which one is better suited to our objective.

Neural Networks are computational models inspired by the way biological nervous systems process information [7]. Although they might appear to be a recent development, they were established before the advent of the powerful computers we have nowadays. They are a collection of connected nodes called neurons that given an input, often a real number, propagate a signal if a certain threshold is crossed when applying a function to the input. Several architectures of neurons have been tested through the years with Convolutional Neural Networks (CNNs) having the most notable performance for the task of computer vision. We want to explore the use of CNNs in the context of disaster assessment, and we believe that it will bloom in the coming years.

The literature review in which we will dive more in-depth in the next chapter sets the foundation for our ideas. The techniques that we use were tested in other contexts before. This fact allowed us to focus on the implementation

alone confident that the experiment would lead to good results. Nevertheless, it would be far too ambitious to cover every topic that is involved in the process of 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. 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.

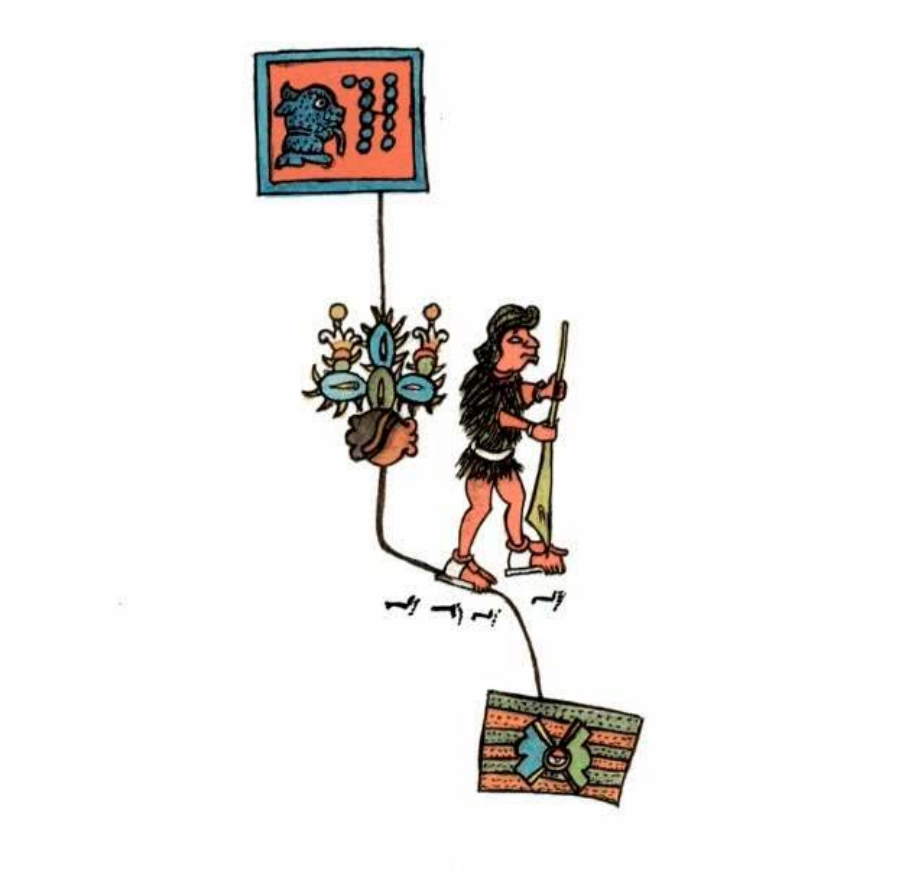


Figure 1.2: An earthquake happened during the year 11 rabbit according to an aztec glyph.

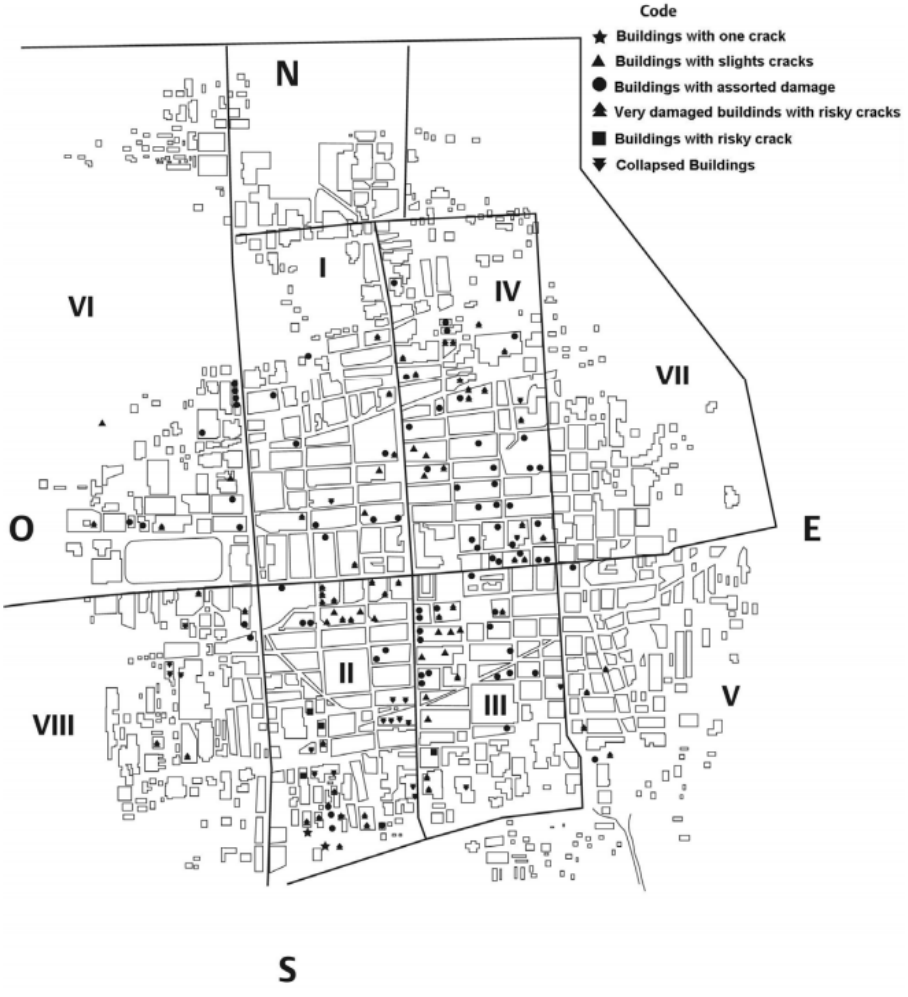


Figure 1.3: A damaged building map for the *San Juan de Dios* earthquake in Mexico City.

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 state-of-the-art technologies in computer vision and how we have used them 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 September 7, 2017, as a study case because of the data that was publicly made available by the CENAPRED.

An excellent introduction to neural networks in the context of remote sensing is in [9]. It reviews all the necessary mathematical tooling to deal with the processing image algorithms and implements them in an interpreted high leve language called Python. A survey of distinct aspects of how to transfer learning from separate domains is in [31].

2.1 Neural Networks

This section will briefly explain how neural networks work and some of the important concepts behind them. The discussion is based on notes by Andrej Karpathy [21], an introduction to neural networks by Christos Stergiou and Dimitrios Siganos [39], and an online publication by Michael Nielsen [30].

An artificial neuron is a function that was conceived based on the idea of biological neurons. It has one or more inputs and sums them into an output. In order to continue with the biological analogy, we say that the neuron is active when a certain threshold is attained after applying a function to the linear combination of inputs. This function is called activation function and should be non-linear. So an artificial neuron has a general form like the following, where x_is are the inputs, w_{ik} are weights, and ϕ is the activation function:

$$y_k = \phi\left(\sum_{i=0}^{n} w_{ik}x_i\right)$$

By design, several artificial neurons can be plugged together to form a network. A set of neurons at the same level from the input is called a layer. It is important to highlight that without the non-linearity of the activation function, several layers of neurons could be simplified to a single layer because a linear combination of linear functions will remain linear.

Many architectures exist for networks of neurons, the simplest one and the one that we will focus on is called feed-forward neural network. In this particular architecture, information flow through the neurons in a unidirectional way. When the output of a neuron is connected to the input of other neurons we call it a hidden layer neuron. It was proven by G. Cybenko [11] that neural networks with at least one hidden layer and with a specific class of activation functions are universal approximators, this means any continuous function can be arbitrarily approximated with a neural network. While this result proves the existence of such a neural network, it does not provide an algorithm of how to build it, however, it shows the representational power if the tool.

Other two main parts of neural networks are the score function and the loss function. While our data points (x_k, y_k) are given, we can choose the

weights w_i to make the output of our model resemble our data. Intuitively, the score function transforms raw data into class scores so that we can choose the correct class given an input. On the other hand, the loss function gives us a measure of how different our output is compared to the ground truth labels.

To be used as a classification tool, the neural network receives the data as input and produces information that allows to predict a class. Broadly speaking, this is done through linear combinations and activation functions. The score function produces a result that is evaluated by the loss function. A way to modify the weights is necessary to improve the performace of the network so it is useful in practice. Fortunately, there are different methods to train neural networks that have added to their increasing popularity and development. One is these methods is known as backpropagation.

As already mentioned, the loss function measures how close our results are to our data. It would be desirable to minimize this function by modifying the weights of the units in the network. To do this we can apply optimization methods such as Gradient Descent. By construction, the neural network is a continuous function except perhaps in some finite number of points. This means that we can calculate the direction of maximum descent for the loss function and use this information to change the weights of the neurons connected directly to it. Subsequently, the new weights are used to recalculate the weights of the neurons connected to these and so on. The algorithm ends when there are no more neurons left to update. This algorithm gets its name from the way the gradient signal propagates through the network in the opposite directions as the information flow in the forward pass.

There are many details involved in the types of functions used in each step of the process described. There is is a lot of ongoing research about the best ways to exploit the neural networks. A deeper explanation is out of the scope of this work.

2.2 Damage Assesment

In the field of natural disasters, we found many references to work that takes advantage of datasets in many different forms. For example Kryvasheyeu *et*

al., in 2016, used the twitter activity during Hurricane Sandy to build a model. They found that social media activity correlates with the proximity of the region in which the Hurricane hits [24]. Nevertheless, they mention that the nature of the data available both in quantity and quality is unusual in part because of the severity and magnitude of damage that the Hurricane Sandy brought about.

A close approach to what we propose is the work of Rashedi and Mori [29]. They gathered a dataset of damaged building images from different sources and gave them a numerical degree of damage. They used these data to train a model consisting of three different pipelines that extracted features of diverse natures, and then used this features to obtain a single continuous value using a sigmoid function. This technique is useful in the sense that it does not require to have images of the state of the building previous to the disaster.

2.3 Convolutional Neural Networks

We found ongoing research about the idea of using features crafted by a neural network in different domains. We took inspiration from some of these techniques for our research framework. This field is still novel and will be expanding in future years.

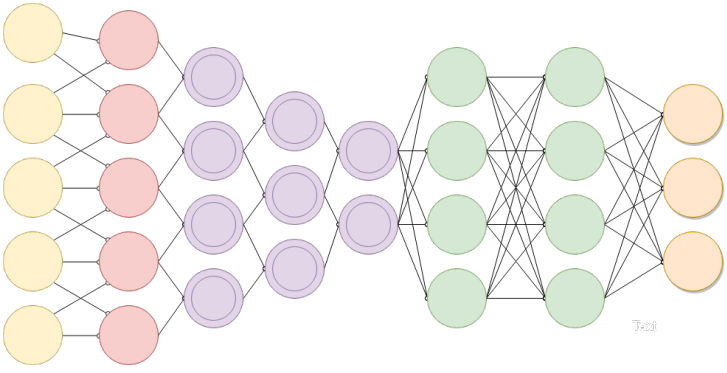
Jeff Donahue *et al.* developed a framework which they called Deep Convolutional Activation Features (DeCAF) [14]. They feed traditional methods with the features extracted from the CNN obtaining accuracies compared to state of the art methods at the time. They took the activations from the n^{th} hidden layer, the layer previous to the one that is fully connected, 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 assign a class to a given image correctly. They showed that the depth of the layer dramatically improves the performance of the features. Domain adaptation consists 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 were able to cluster the objects across domains

regardless of the origin of the images. In subcategory recognition, the task at hand is to differentiate 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 for the object in the picture, the classifier must semantically classify the scene instead. The task above is considered extremely difficult, and the previous approach was to use hand-engineered features and a multi-kernel learning baseline. While the accuracies reported by this task were 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 original design of the network was different. Additionally, they developed an open source framework that later matured into Caffe [19], a framework designed to train CNN models.

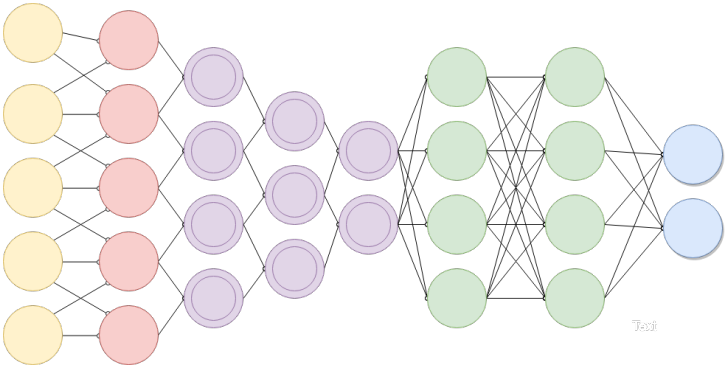
Another attempt that adds to the evidence that features engineered by the Neural Network work pretty good off the shelf, is Razavian `et al.` [33]. They use features extracted from the CNN `OverFeat` model which was made public by Le Cun *et al.* [38]. In the original article, they explore the advantages 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` excel. They 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 sophisticated methods involving handmade features.

Transfer learning consists of using the knowledge acquired by a model during a training phase to another domain of knowledge. It is explored by Yosinski *et al.* in [43]. They propose to use an already trained architecture in new tasks by replacing different layers and retraining only the topmost layers. An experiment was designed to test the transferability of different layers. They found strong evidence that suggests that even though the training sets are of different nature, features transferred from different tasks improve the performance

of the CNN against random weights. They used ImageNet [13] splitting the dataset obtaining two different groups of images. They also found that the transferability is negatively affected by the deepness of the layers, meaning that higher layer features are specialized to the given task. This idea gave us a clear starting point for the task in our hands.



(a) Task A



(b) Task B

Figure 2.1: A graphical representation of how transfer learning works. The last layer of the architecture is changed while leaving the rest without modification.

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 [26] Long *et al.* use the features extracted from the CNN to segment such an image. They modified the architecture of several classification networks into fully convolutional networks, and the features learned were fine-tuned for the segmentation task. This process is very promising for the field of Remote Sensing where we engineer classification features by using traditional methods such as manipulating the image information without taking into account the semantic context. Other fields, such as biomedicine, also explored this topic [35].

The possibility of having a single model that can perform well in many different tasks is explored in [20]. Kaiser *et al.* aimed to create a single model capable of performing well in very different tasks. They noticed that while CNNs give impressive results in a variety of functions, they need to repeat training efforts for every situation. They created an architecture capable of receiving training data from 8 different corpora, ranging from image classification to language translation or natural language processing. Given the very different nature of the training data, they needed to create encoders 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. Finally, they found that the domains in which the data is limited are benefited with the joint training while the other tasks suffer from little or no loss of accuracy at all. While not achieving state of the art results in any of the domains, they found that it is indeed possible to train a model to learn many different objectives.

Zhou and Prasad used an idea known as active learning to improve the performance of a classifier taking the most of small labeled datasets [44]. Active learning is based on the observation that the traditional approach of randomly selecting training samples leads to redundancy in the training sets. The approach is to enable the learner to select the most informative training data. In their research, they explore the use of active learning together with semi-supervised learning techniques.

2.4 Computer Vision

Neural networks appeared as function approximators that could be adjusted to simulate the behavior of linear functions. As they grew in sophistication, new architectures were capable of performing well in more complex tasks. The pioneer in using them for image recognition was Yann LeCun in 1990.

Le Cun *et al.* [10] proposed to use an architecture of a multilayer neural network that was able to learn directly from the data with no prior feature extraction. 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 classes that represented each of the possible digits. This design, which was trained using backpropagation, is what now we know as CNNs. The reason that their research was so groundbreaking 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 technique was not able to be applied to very complex objects. With the tremendous advances that computer power has suffered in the late years, this has been proven to be incorrect. In 2009 a prominent image database was gathered and published [13]. Ever since this happened, this database became the defacto dataset to test classification methods. A few years later, in 2012 Krizhevsky *et al.* [23] proposed the use of CNNs in the daunting task of classifying the image dataset.

The use of neural networks to automatically detect roads was explored already before CNNs became widely used in computer vision tasks [28]. 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. They used aerial imagery in addition to vector information on roads as training data. Instead of manually tagging the photos they proposed a model that assumes a certain amount of thickness in the streets, which usually have no dimension. To reduce the input dimensionality, they applied Principal Component Analysis (PCA), keeping the most informative principal components. As a method of post-processing, they use a neural network to reduce the noise in the images, efficiently getting rid of false

positives and false negatives by using context information. One of the valuable lessons learned from this experiment is the value of the random rotations in the training data. It is common that road networks in cities create grids, they realized that a model trained with information from a particular town would perform poorly if data from a new one is shown to the model, to put it in another way, the model generalized poorly. They found that we can relive this if we applied random orientations to the data as when we are dealing with bird-eye sight, and there is no correct orientation for images. They also found that pre-processing the pictures with methods such as edge detection showed no improvement in the performance of their pipeline, they attributed this to the fact that the neural network crafts features of this nature when learning to perform the task of recognizing roads.

Another attempt to use features extracted from a CNN in a different context is in Michael Xie *et al.* [42]. They 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 excellent 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. 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 strengths at the same time. The transfer model outperforms every model except for the one based on survey data, which strongly suggests that the transfer learning technique is extracting complex information from the aerial scenes. They clarify that although this approach can be useful when conducting surveys is prohibitively expensive.

2.5 Remote Sensing

In recent years groundbreaking advances in computer vision have led to tremendous advances in other science fields. These algorithms are behind many of

the tools that we use every day, and that we take for granted. Nowadays face recognition techniques are so advanced that we can use them as an authentication method. Autonomous vehicles are soon to be a reality, reshaping the way we move. We can build three-dimensional models from hundreds of overlapping images [40]. In particular, we are interested in the field of remote sensing.

Remote sensing is a field that studies the energy emanating from the earth's surface and captured by a sensor mounted in an aerial vehicle or a satellite [34]. Landcover classification is one of the most important disciplines in the field. In the case of digital imagery, we want to classify pixels into clusters of interest to extract information from the raw data that the sensors acquire. In our case, we want to detect pixels that represent damaged buildings automatically.

Kussul *et al.* [25] explored the use of CNNs in the context of land-cover classification. 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 two 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 sensors. This process outputs a pixel-wise classification; they then use a traditional 2-D CNN on the images. To not 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). To improve the performance, we must handcraft features from the original bands. In [37], Grant *et al.* explore the use of the mentioned above method of transfer learning and data augmentation in the context of remote sensing images. They tested their methods with well-known high-resolution datasets, and they obtained state of the art results.

In [22] and [8], Kendall *et al.* propose and enhance their own approach by extending their 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 accurate 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 SegNet.

2.6 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. When tagged data is scarce and difficult to obtain this method is a good choice. Frequent alterations that the photos are subjected to include rotations and reflections. When we choose to use 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 create the network learn features on objects that it will not see in a real situation. Fortunately, aerial imagery does not present this problem. No particular orientation is considered correct with aerial pictures. This case means that we can dramatically boost the size of our dataset without the cumbersome process of manually tagging.

The reasoning behind this idea is that when we see a picture, our brain automatically orients it into its correct position. Showing the network an image with different positions and orientations of an object we enrich its knowledge about it, we can see an example in Figure 2.2.

Figure 2.2: Same image with different rotations. A way to boost the size of the training set.

2.7 TensorFlow

TensorFlow is a machine learning system developed by the Google Brain Team to supersede its first-generation system, DistBelief. 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 lets us express machine learning algorithms using a standard interface with implementations targeting a wide range of devices. Complex models that first implemented in DistBelief such as Inception got a performance boost of a factor of x6 [4] when they got moved 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 [5]. It uses an elegant data-flow system in which both the operations and the state of the algorithm get represented as nodes and edges in a directed graph. This lets the system build 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 at runtime which pieces to use; this is useful when taking into account the whole development process of a data product and how it evolves. For example, when a developer experiments with data in a single computer before deploying the system to be trained with a more massive data set in a cluster of computers. Another example comes when the model training process, it can be deployed to an online service which will run on a single computer, but then it can be implemented in a mobile device for offline use. In each of these steps the underlying environment is entirely different. However, TensorFlow adapts automatically to each situation.

As a standard 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 smart 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 op-

erations. 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 we can use it, 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 more comfortable to explore new techniques as the framework produces backpropagation code for any combination of operation nodes.

They built TensorFlow with considerably large data-sets in mind. It provides an intern library that allows the distribution of datasets that would be too large to fit in RAM. Instead, data can be sliced, taking advantage of how some algorithms work. Additionally, communication between nodes uses 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 fundamental to our study because we are interested in the features that the system craft to perform 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 repeat 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 occurs only once.

The framework code is deeply optimized. They built it 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 ex-

tended 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 comes with the framework is TensorBoard. With the immense complexity that machine learning models offer at this scale, it is essential to know what is happening at any point in the training process. TensorBoard offers a glimpse of how the architecture for a particular computation graph looks. This tool will become handy later when describing our model.

There are several alternatives 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 for their performance. We choose TensorFlow because the pipeline was already in Python, to minimize the gluing code. Among the examples that come with the system, there is an end to end example of how to transfer learning from one trained net to another task.

2.8 Summary

An extensive literature review gave us the tools to plan our methodology, based on the ideas that we thought were promising for our objective. Given the results reported in [43], we decided to use transfer learning for our task. We decided not to explore the concept of data augmentation because we had the ability to control the size of the training set. We took the tool that TensorFlow provides to perform transfer learning, using Inception as the architecture to start with. Based on [33], we choose to compare the method using transfer learning with a more classical approach, in our case we chose a random forest classifier. To keep the scope of our work not too ambitious we left out the possibility of collating our results with official data as suggested on [24].

Chapter 3

System architecture

> Those who can imagine anything,
> can create the impossible.
>
> Alan Turing

Analyzing a massive amount of images requires a better way of handling and sorting them than just storing them into directories. Borrowing ideas of how is the transfer learning process made by the engineers in the Tensorflow team; our catalog system grew to comply with those 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 for a better set of tools to overcome several difficulties. The training stage involved a repetitive process consisting of manually inspecting the available imagery. To this end, we built a web application on top of our catalog system facilitating this process. The action 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 encoded file names. So a new feature was implemented into our tool, it shows the images and their respective tag and lets the user either delete or edit the classification. Finally, when we had the trained model, and it was predicting on the orthorectified rasters, the ability to geolocate the

damaged areas and place them on a map was very convenient.

As it was just exposed, we built the application thinking about the final user. Even though the correct implementation from a design perspective is out of the scope of this work, it was exciting to explore it as this would be the kind of problems that arise 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. We unveil the techniques used to obtain and curate data, and details of the pipeline architecture.

3.1 Backend

To understand why some design decision were made we need to understand the nature of our data. Data came in directories taken from the drones and split by flying dates. The quantity and naming of the images were not uniform across towns. Additionally, drone imagery provides 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. 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, we decided to build the system on top of a Python web framework called Django [1]. It offers many solutions out of the box, including a familiar line interface set of commands and an object-relational mapper (ORM) 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 Django's ORM. It creates Python classes that represent tables in the database and manipulates them as Python objects. In the figure 3.1 we show a subset of the database diagram. We do not show tables inherent to the features of the Django framework as they were not modified.

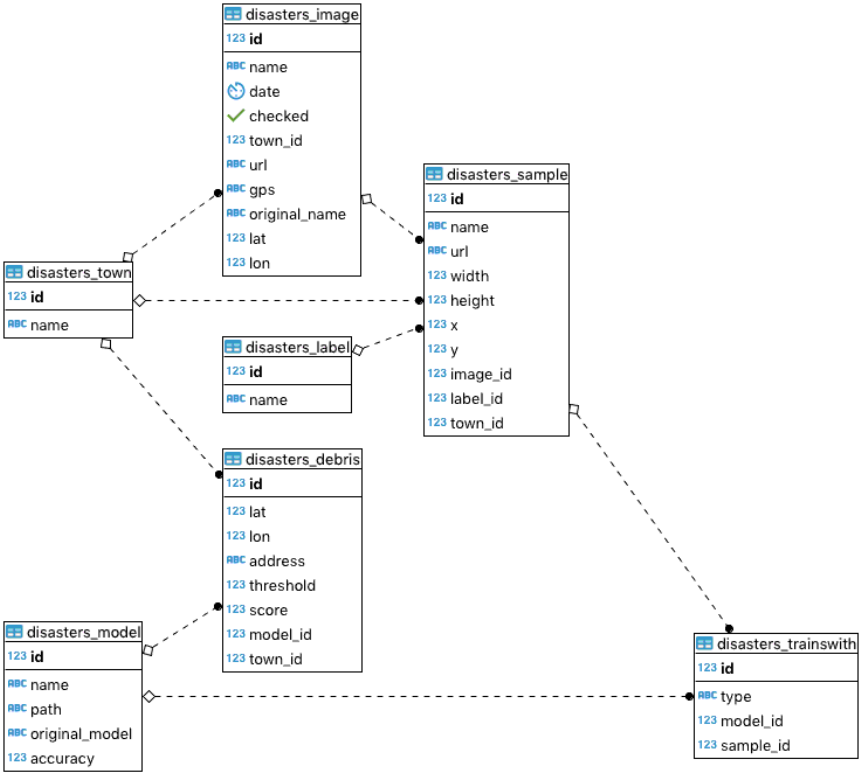


Figure 3.1: Database schema.

We based our database design on reproducible research. It was important

to keep track of the characteristics of the models trained as well as the training images used for each model. It was also desirable to be able to reuse models trained with different sets of images for benchmarking. Finally, we wanted to use the best model to predict on the final rasters.

3.1.2 Ingestion

We built a process to ingest the images from CENAPRED's FTP. It lazily downloads the images by checking first if the file is already present in the temporary folder, if the file does not exist already, we download it. The system then tries to add it to the database and copies it to the 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, we erased the file from both, and the state of the system remains as it was before the ingestion attempt. In Figure 3.2 we show a diagram of this process.

3.1.3 Tagging

Aerial tagged data is scarce. In particular, for our experiment, we do not have any useful metadata on the images. The idea behind our design was to crowdsource this effort. 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 two categories: the image shows damage, and there is no visible damage in the image. When we obtained a positive answer, the system persists the image in the database with the information about the original image and the coordinates relative to it. In Figure 3.3 we show a diagram of this process.

3.1.4 Training

We need to be able 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 could compare models in a simple fashion. To this end, a random sample is created every time the application runs and the image set

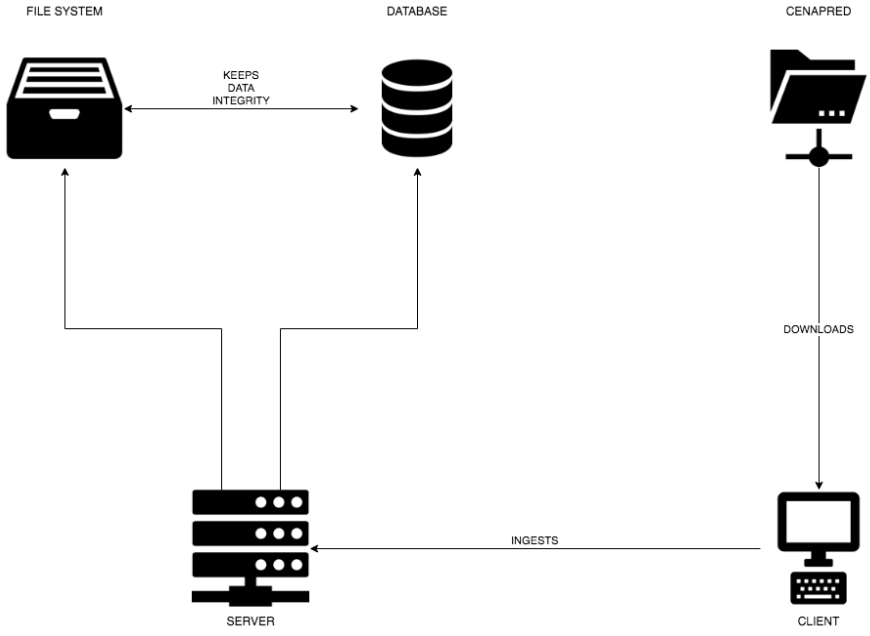


Figure 3.2: We download images from a ftp that CENPRED gave us access to.

is splitted in three sets: training, validation, and testing.

The model does not need any geographical information in its training as it relies only on the pixel intensities. We took this to our advantage as we can use the raw drone images for the training stage while predicting on orthorectified mosaics. Tensorflow provides a script for retraining 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. We modified and integrated the script into our framework to fit our database design. This process made easier to train several models with similar training, validation, and testing sets. We show a diagram of our process in Figure 3.4.

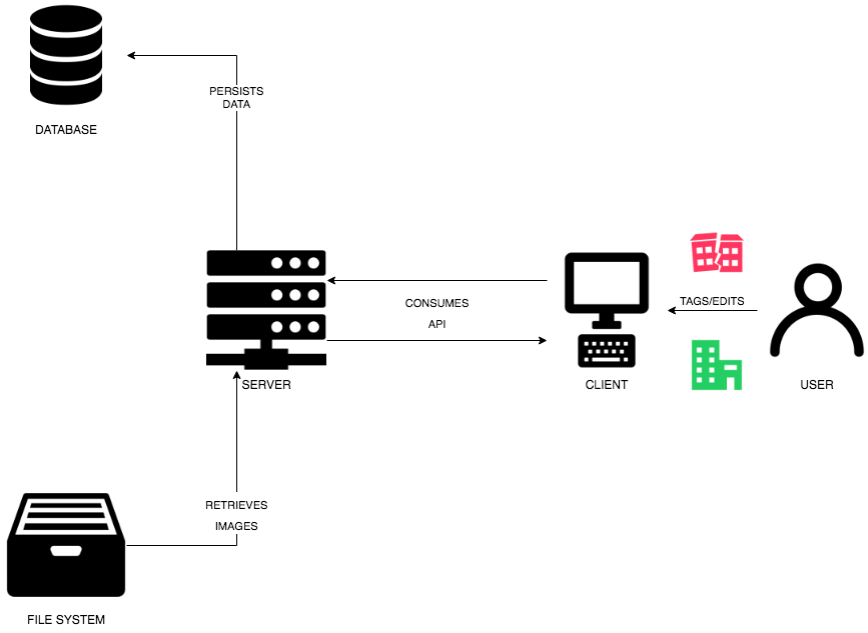


Figure 3.3: Tagging process.

3.1.5 Prediction

Although this topic will make more sense once we learn about our experiment results in next chapter, it is worth explaining the process here. Once we have a trained model, we can use it to classify new incoming data and show it to the final user. Once the rasters are subject to the predicting process, results are persisted in the database and the information may be accessible through a client. We show how we ensvision this process in Figure 3.9.

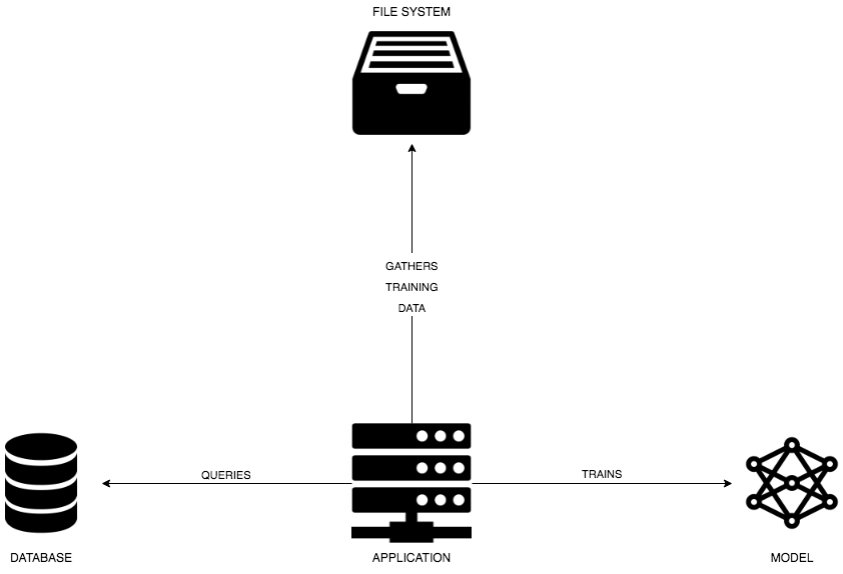


Figure 3.4: Training process.

3.2 Frontend

In the previous section, we talked about the backend components that perform the data governance. At the beginning of the process, we used these components to start our research. However, we soon noticed that it was challenging to manipulate data directly from the file system and the database. To alleviate this inconvenience, we built a set of tools that helped us in this process. This tools were developed using HTML, Javascript and CSS, using a set of common frameworks: Open Layers [1], Bulma [2], and jQuery [3]. While making a full account of the details involved in a web application goes beyond the scope of the present work, we offer a brief description of our tool.

[1] https://openlayers.org/
[2] https://bulma.io/
[3] https://jquery.com/

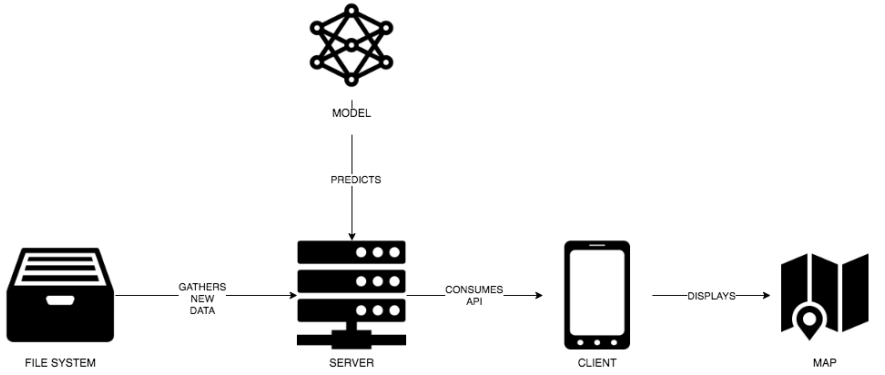


Figure 3.5: The system is prepared to predict on new data and display a map with the places in which it founds damaged buildings.

3.2.1 Data Visualization

First of all, we needed to know the number of images in the database as well as their geographical location. We developed a view in which we show a map with all the ingested images separated by the town they belong. We also show how many tagged samples have we submitted so far. This view updates each time we add new information to the database. In Figure 3.6 we see the individual views for each of the towns.

3.2.2 Tagging

Cropping and tagging a large number of samples manually from the images was prohibitive, to overcome this obstacle, we developed a unique view 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 could be crowdsourced.

A web client was developed using jquery and OpenLayers, and it consumes the endpoints described in the previous section. The interface is an

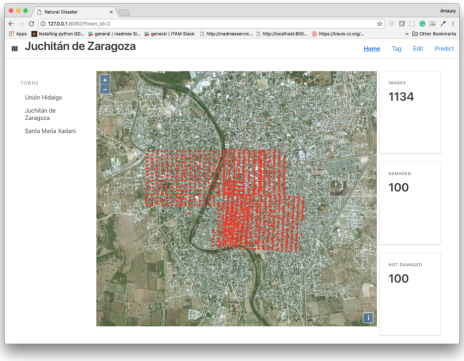
image viewer with a selection and a button to submit an opinion about the highlighted area. In the backend, the image is cropped and ingested into the database assigning the tag that the user selected. In Figure 3.7 we see an example of this view.

3.2.3 Editing

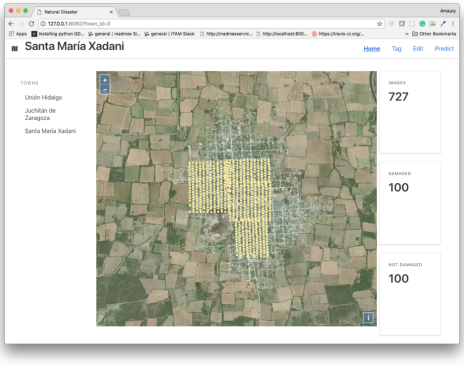
We noticed that the tagging process was error-prone. In some cases, an image was given an incorrect label, and the error was immediately spotted. To deal with those mistakenly tagged images, we needed to inspect the database and the file system to delete the record. A visualizer for the tagged images was developed to overcome this difficulty. It consists of an interface were the tagged images appear with their respective tag. The interface offers a way to delete or edit the previously assigned tag.

3.2.4 Predict

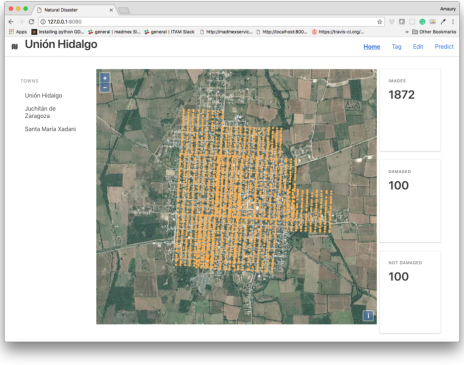
Finally, we developed a view of the points that the algorithm finds in the orthorectified raster. In the final stage of the process, we produced a list of potentially damaged buildings. Those sites also go into the database with geographical information and human readable address obtained from the Google Maps API. We think that this can be helpful to allocate resources most efficiently. Figure 3.9 shows how this view looks like.



(a) Juchitán de Zaragoza



(b) Santa María Xadani



(c) Unión Hidalgo

Figure 3.6: Information interface. We give a summary of the information ingested in the system.

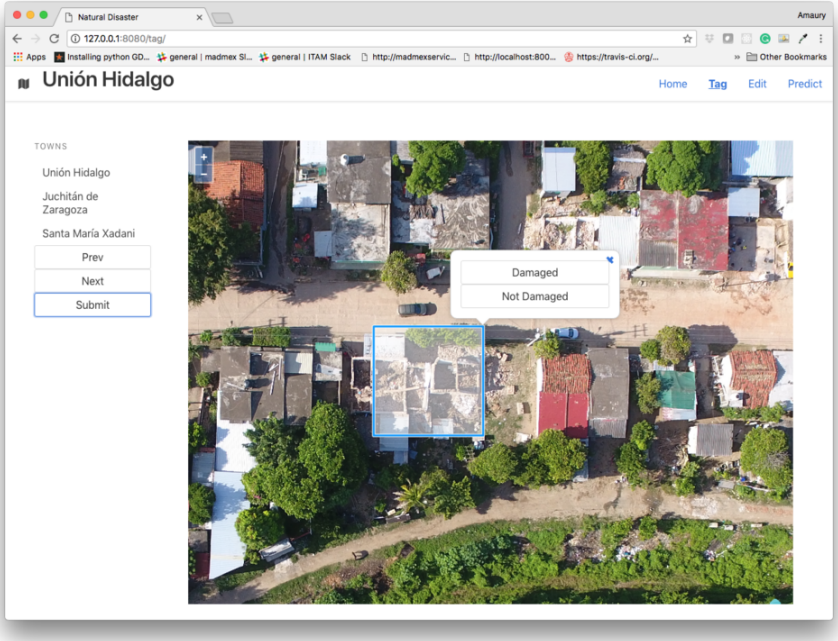


Figure 3.7: Submit interface.

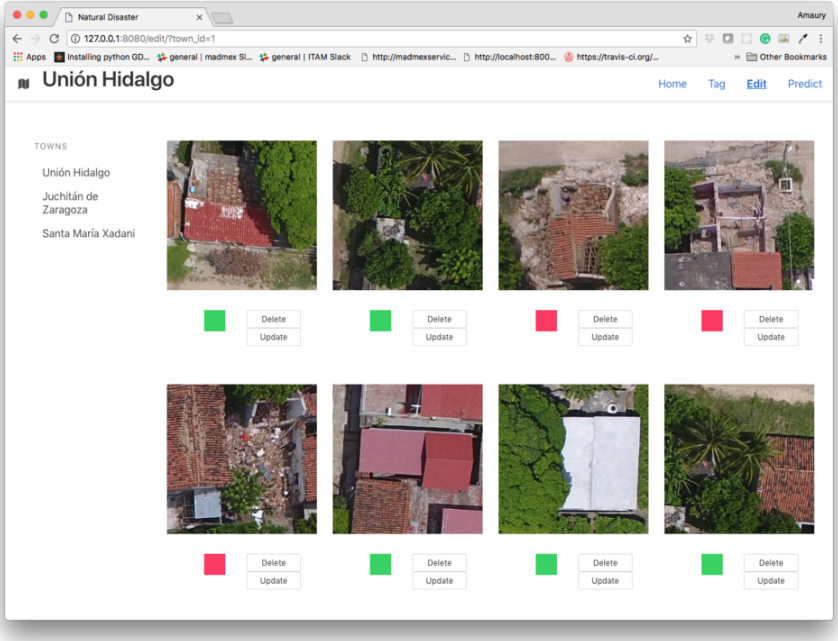


Figure 3.8: Edit interface.

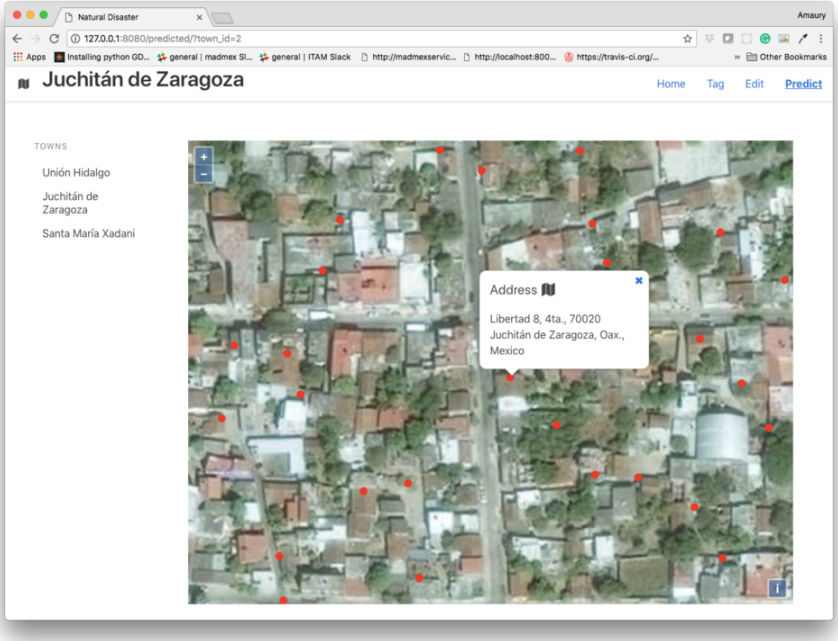


Figure 3.9: Predict interface.

Chapter 4

Data analysis

> The question of whether a computer can think is no more interesting than the question of whether a submarine can swim.
>
> Edsger W. Dijkstra

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. Validation must be carefully crafted to obtain accurate results. In other words we need to control the variables that might affect the outcome of the experiment so we can isolate the phenomena that we are studying.

To have a complete picture of how our approach is compared with other classical techniques from 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 4.1), 727 images from Santa María Xadani (Figure 4.2), and 1872 images from Unión Hidalgo (Figure 4.3).

As we can see from Figures 4.1, 4.2, and 4.3, drones flew in regular patterns forming a lattice of points, these clouds of information distribute evenly around the towns of interest. Given this distribution, it was reasonable to hypothesize that there was an underlying similarity between images of the same town. This similarity might be caused by the light conditions during the exposure that tend to have less variance among similar times and places. A limitation found was that the size of the images[1] prevented us from computing regular metrics such as the Euclidian distance between a pair of images. To overcome this problem, the information from the pixels of each image was flattened into a vector comprising the means and standard deviations for each color channel. In summary, for each image, we obtained a vector of six components.

To test our hypothesis over this new set of data, we used an algorithm known as t-Distributed Stochastic Neighbor Embedding (t-SNE), introduced by Laurens van der Maaten [27]. This technique allowed us to visualize the information in a two-dimensional plane[2]. As we expected the images of the different towns cluster in this low dimensional representation. We show our results in Figure 4.4.

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

[2] 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 give us insights about this underlying structure.

Figure 4.1: Juchitán de Zaragoza.

Figure 4.2: Santa María Xadani.

Figure 4.3: Unión Hidalgo.

The outcome of this experiment gave us a reason to model our methodology around the characteristics of the data. Using the information of a certain town and later testing the model with data from another one allowed us to evaluate the generalization capacity of the model.

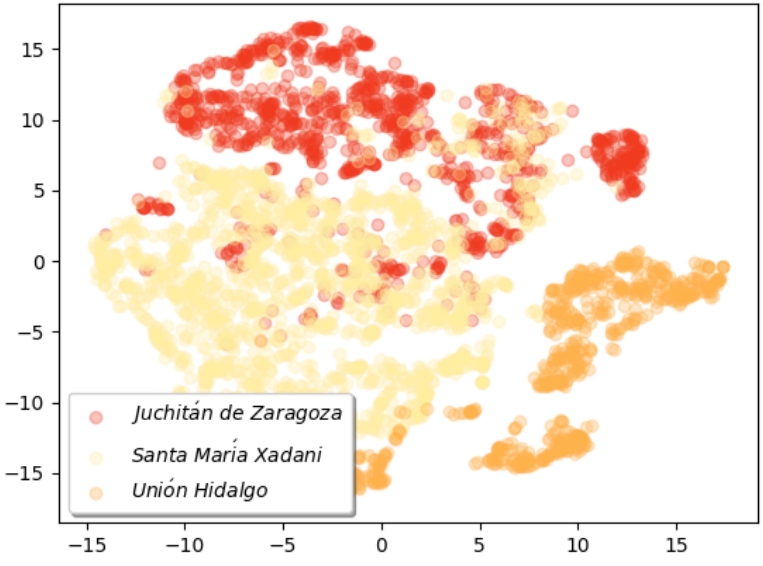


Figure 4.4: T-sne diagram.

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 the images, and then we trained a classification model on the

resulting data[3]. 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 [15]. 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 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

[3]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.

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.

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

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.

One of the difficulties that we found during our research process was that several images might depict the same collapsed building (Figure 4.7). 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 ar-

eas, 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.

Figure 4.7: An example of the same building in two diferent images. The drone flew over the same place in different moments and directions.

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

Table 4.1: Permutation codes.

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

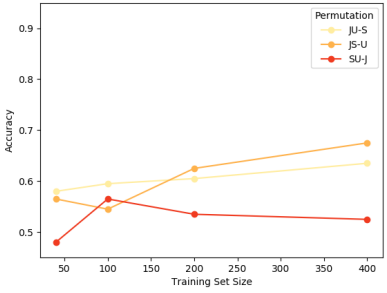
be very likely to have the same building both in training and in the testing sets. This situation would lead to having very high accuracies for our model but not such good performance in a real setting.

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.

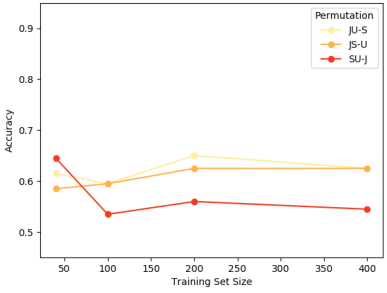
4.2.1 Classic Computer Vision

We extracted the HOG features using an implementation from the Python package scikit-image [32]. We used eight orientations, and a 16×16 pixel window. Each town was selected once for testing, in table 4.1, we show the codes that we use in Figure 4.8 to show the results that we obtained.

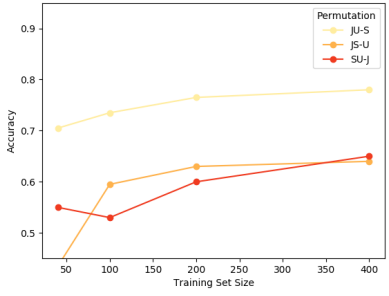
It was surprising that the model that showed the best performance was the one in which we considered means and standard deviations. We expected that the specialized features of the HOG model to perform better than a simple mean and standard deviation extraction. We think that the reason behind this was that the HOG features were designed to find objects in images, but



(a) Histogram of Oriented Gradients



(b) Means



(c) Means + Standard Deviation

Figure 4.8: Training example using classic computer vision models.

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 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.

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].

Table 4.2: Permutation codes.

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 [18].

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

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

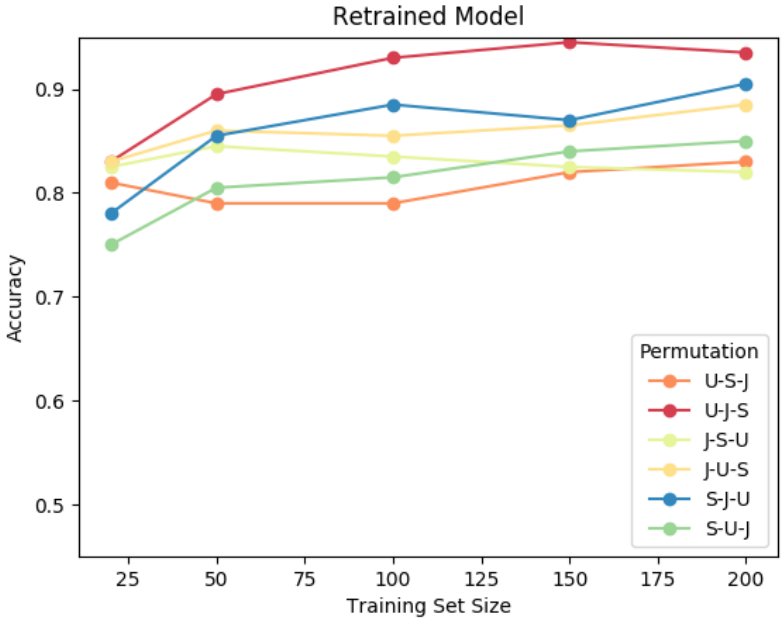
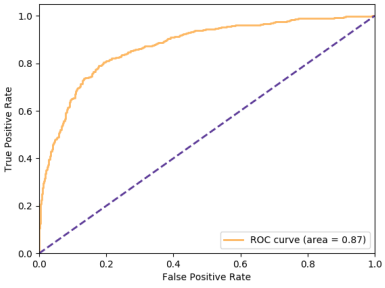


Figure 4.9: Results of our experiment. As we expected, the graph shows a positive correlation between the accuracy and the number of training samples.

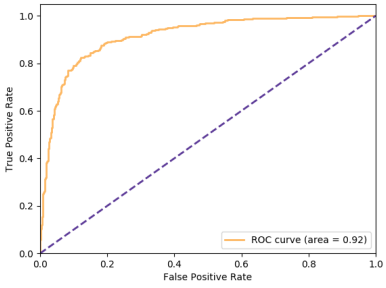
predicted as p and the number of elements that are indeed p. We are also interested in the false positive rate (FPR) or *specificity*, which is the proportion of incorrectly assigning p given that the condition is n. In a ROC curve, we plot the sensitivity as a function of the specificity; it is used to select a decision threshold and have a tradeoff between detecting as many positives and having as little 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

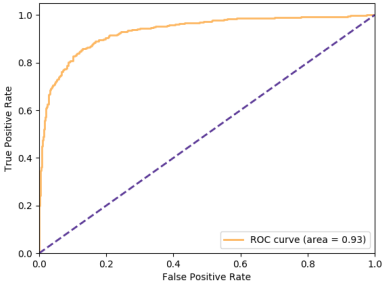
application at hand. For instance, when we are dealing with disease diagnosis, a screening test should be very sensitive whereas a confirmatory test should be very specific. In our case we want our classifier to be very sensitive because there is not as much overhead to have false alarms. In Figure 4.10 we show the ROC curves for each of the transfer learning models. To build them, the prediction for each test image was recorded for 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.



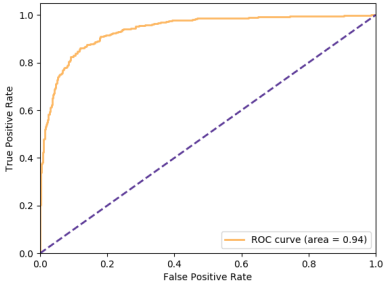
(a) Training Set Size: 20 Images



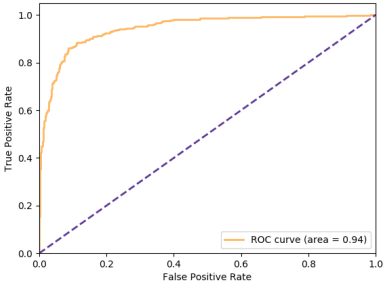
(b) Training Set Size: 50 Images



(c) Training Set Size: 100 Images



(d) Training Set Size: 150 Images



(e) Training Set Size: 200 Images

Figure 4.10: Receiver operating characteristic curves for the different training set sizes. We show the performance of the transfer learning models using Inception as the base.

Chapter 5

Implementation

JOI
Mere data makes a man. A and C and T and G. The alphabet of you. All from four symbols. I am only two: 1 and 0.

K

Half as much but twice as elegant, sweetheart.

Blade Runner 2049. Dir. Denis Villeneuve, 2017.

Even though deep learning has allowed us to take a giant leap forward on the performance of several tasks, and contrary to what some researchers claim [20], 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 useful in the task of locating damaged buildings. We now centered our efforts on creating a final model taking into account the entire dataset of 600 tagged samples. This model was not validated as there were no images left to use as a testing set. This process would closely resemble a real pipeline (Figure 5.1).

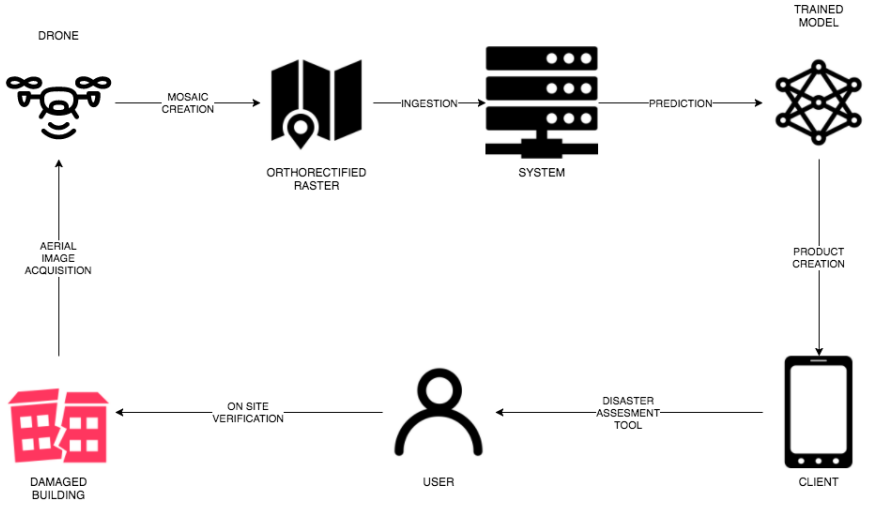


Figure 5.1: The complete cycle of the process. Drones gather data, gets preprocessed, we train the model, and the final user verifies the information extracted by the model on site.

We used 0.95 as our decision threshold. According to the best of our models, this threshold guarantees us that we would have around a 0.47 probability

assigning a damaged label when the condition is present while only having a 0.01 likelihood of mislabeling a no damaged building. While this decision threshold was not necessarily valid for our final model, it gave us a hint on what decision to make. We are sacrificing sensibility for specificity when we predict a damaged building we are confident that it will be a true positive. We could argue about the low probability of assigning a damaged label, but it all makes sense if we think that we test many samples for each zone.

5.2 Post-processing

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

5.2.1 Overlapping

We envisioned two methods to apply our predictive model to a full image. Sliding-window, in which we move a fixed squared along the x-axis and the y-axis, and for each (x, y) value the model is used to predict the condition of the particular window. Another method is to use random sampling, in which we place the window randomly in the image, and then the model is used in the same fashion using a parameter to control the density of the windows. In both cases, we saved the locations of the windows with positive outcome and painted them over the original image. We show an example of our results using the random sampling method in Figure 5.2.

A limitation of this approach is that damaged sites are counted several times by the algorithm. A method to alleviate overlapping bounding boxes is known as non-maximum suppression (NMS), a technique borrowed from facial recognition algorithms. First proposed by A. Rosenfeld and M. Thurston in the context of edge detection techniques [36], it consists of checking the percentage of overlap in the set of bounding boxes and discarding those that

Figure 5.2: An output of our process after predicting over randomly sampled patches. We see positive predictions overlapping over some regions.

repeat. The final result after postprocessing looks like Figure 5.3.

5.2.2 Orthorectification

In order to transform our images into useful information from the spatial point of view we need to put them through a process called orthorectification. It removes the effects that arise from the perspective in which the drone took the image. It also mosaics and puts the images together to create a raster that covers the region of interest. This process needs specialised software, and CENAPRED built those mosaics for us. These rasters are the ones that we use to test our final model. The mosaics gave us the ability to assign coordinates

Figure 5.3: The same region as Figure 5.2 after post-processing.

to each of the sites in which the model gave us a positive output.

5.3 Results

Using the methods described in the previous section, we applied our process to the three orthorectified rasters provided by CENAPRED. We randomly placed 322×322 pixels boxes along the raster, randomly sampling the location of each of them. The boxes with a positive outcome were saved and post-processed to find overlaps. Once we have the non-overlapping boxes, we transform the center pixel of each box to world coordinates. Finally, these coordinates are used to query Google Maps API obtaining a human-readable address for each point. In table 5.1 we show details of this exercise.

We produced a shape-file containing the results for each town given the output of the algorithm. This layer can be overlayed on top of the raster file using a Geographic Information System software such as QGIS. Figures 5.4,5.5, and 5.6 show our results. Additionally, the results are also exposed via the REST interface so we can visualize them in the web application as we saw in Figure 3.9.

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 whether 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.

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

5.4 Digest

We were able to build a process that allowed us to transform raw images taken by drones into a map of points with a high probability of damage in an automated way. Coupled with a usable graphical interface, this tool would allow acting quickly and effectively in the event of a catastrophe. The method, although focused on earthquakes, can be adapted to other situations. Our experiment proves that it is possible to use innovative technologies to aid the poorest population.

Figure 5.4: Juchitán de Zaragoza.

Figure 5.5: Santa María Xadani.

Figure 5.6: Unión Hidalgo.

Chapter 6

A glimpse into the future

> A goal without a plan is just a wish.
>
> Antoine de Saint-Exupery

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 Data Science

The product that we developed during this research might serve as a baseline in the event of a new natural disaster. The process involved the use of several techniques and tools to create a data pipeline capable of detecting damaged buildings automatically using drone imagery. The purpose was to provide the decision makers with a tool that allow them to allocate resources efficiently. 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, but a useful device that can help

us reach places we could just imagine before.

The discussion of the need for a correct use of the information extracted with our system was an additional topic that was not covered. Maps are only useful when they serve as tools to make better decisions, otherwise, they end up being worthless drawings hanging on the wall. Operations research is the discipline that can help our idea to have a real impact. For example, we envision a situation in which our map can lead to a correct allocation of people to go on site to inspect the buildings and a dispatch system that makes every second count. Another imaginable scenario would be to compare the model's predictions with the official information to spot anomalies. This would bring light to processes that are liable to be tainted by corruption. Leveraging the advances in data processing and the many sources available is key to a better future.

6.2 Drawbacks

We noticed that, even when the classifier performs well in environments different to the one we trained it with, it learns to classify debris, not precisely damaged buildings. We noticed some examples in which the classifier correctly finds scenes with the presence of debris, 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 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.

One aspect of the proceeding that we did not include in this research was the postprocessing of drone data. CENAPRED gave us a post-processed raster with the georectification 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. Should the information that the system outputs be public? What if people use it for nefarious purposes such as looting? With the power of data follows a greater responsibility.

6.3 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 these ideas can impact other fields of interest. By no means, this review pretends to be exhaustive, but just to give a broader picture.

6.3.1 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 with improved 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 for. This means that the

resources 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.

6.3.2 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 spectral signature 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) [17] 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) [16] which gathers information about species in the different ecosystems that exist in Mexico. Both projects can benefit from this novel techniques.

6.4 Conclusion

Our efforts showed that it is possible to deliver a preliminary product with little training effort. The performance that a fully trained CNN reaches 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.

Bibliography

[1] Django (version 2.0) [computer software]., 2017.

[2] Sismo de de tehuantepec 2017-09-07 23:49 (m8.2), 2017.

[3] Sismo del día 19 de septiembre de 2017 puebla-morelos (m7.1), 2017.

[4] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[5] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek Gordon Murray, Benoit Steiner, Paul A. Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zhang. Tensorflow: A system for large-scale machine learning. *CoRR*, abs/1605.08695, 2016.

[6] V. Garca Acosta. Historical earthquakes in mexico. past efforts andnew multidisciplinary achievements. *Annals of Geophysics*, 47(2-3), 2004.

[7] I. Aleksander and H. Morton. *An introduction to neural computing*. Chapman and Hall, 1990.

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

[9] M.J. Canty. *Image Analysis, Classification and Change Detection in Remote Sensing: With Algorithms for ENVI/IDL and Python, Third Edition*. Taylor & Francis, 2014.

[10] Y. Le Cun, O. Matan, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, L. D. Jacket, and H. S. Baird. Handwritten zip code recognition with multilayer networks. In *[1990] Proceedings. 10th International Conference on Pattern Recognition*, volume ii, pages 35–40 vol.2, Jun 1990.

[11] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, 2:303–314, 1989.

[12] A. Molina del Villar. 19th century earthquakes in mexico: three cases, three comparative studies. *Annals of Geophysics*, 47(2-3), 2004.

[13] Jia Deng, Wei Dong, Richard Socher, Li jia Li, Kai Li, and Li Fei-fei. Imagenet: A large-scale hierarchical image database. In *In CVPR*, 2009.

[14] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. *CoRR*, abs/1310.1531, 2013.

[15] William T. Freeman and Michal Roth. Orientation histograms for hand gesture recognition. Technical Report TR94-03, MERL - Mitsubishi Electric Research Laboratories, Cambridge, MA 02139, December 1994.

[16] Nashieli Garcia-Alaniz, Miguel Equihua, Octavio Prez-Maqueo, Julin Equihua Bentez, Pedro Maeda, Fernando Pardo Urrutia, Jos J Flores Martnez, Sergio A Villela Gaytn, and Michael Schmidt. The mexican national biodiversity and ecosystem degradation monitoring system. *Current Opinion in Environmental Sustainability*, 26:62 – 68, 2017.

[17] Steffen Gebhardt, Thilo Wehrmann, Miguel Angel Muoz Ruiz, Pedro Maeda, Jesse Bishop, Matthias Schramm, Rene Kopeinig, Oliver Cartus, Josef Kellndorfer, Rainer Ressl, Lucio Andrs Santos, and Michael Schmidt. Mad-mex: Automatic wall-to-wall land cover monitoring for the mexican redd-mrv program using all landsat data. *Remote Sensing*, 6(5):3923–3943, 2014.

[18] David M. Green and John A. Swets. *Signal Detection Theory and Psychophysics*. Wiley, New York, 1966.

[19] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[20] Lukasz Kaiser, Aidan N. Gomez, Noam Shazeer, Ashish Vaswani, Niki Parmar, Llion Jones, and Jakob Uszkoreit. One model to learn them all. *CoRR*, abs/1706.05137, 2017.

[21] Andrej Karpathy. cs231n: Convolutional neural networks for visual recognition, 2015.

[22] Alex Kendall, Vijay Badrinarayanan, and Roberto Cipolla. Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding. *CoRR*, abs/1511.02680, 2015.

[23] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[24] Yury Kryvasheyeu, Haohui Chen, Nick Obradovich, Esteban Moro, Pascal Van Hentenryck, James Fowler, and Manuel Cebrian. Rapid assessment of disaster damage using social media activity. *Science Advances*, 2(3), 2016.

[25] N. Kussul, M. Lavreniuk, S. Skakun, and A. Shelestov. Deep learning classification of land cover and crop types using remote sensing data. *IEEE Geoscience and Remote Sensing Letters*, 14(5):778–782, May 2017.

[26] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. *CoRR*, abs/1411.4038, 2014.

[27] L.J.P.V.D. Maaten and GE Hinton. Visualizing high-dimensional data using t-sne. 9:2579–2605, 01 2008.

[28] Volodymyr Mnih and Geoffrey E. Hinton. Learning to detect roads in high-resolution aerial images. In *Proceedings of the 11th European Conference on Computer Vision: Part VI*, ECCV'10, pages 210–223, Berlin, Heidelberg, 2010. Springer-Verlag.

[29] Karoon Rashedi Nia and Greg Mori. Building damage assessment using deep learning and ground-level image data. 2017.

[30] Michael Nielsen. neural networks and deep learning, 2017.

[31] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, Oct 2010.

[32] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python . *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[33] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. CNN features off-the-shelf: an astounding baseline for recognition. *CoRR*, abs/1403.6382, 2014.

[34] John A. Richards. *Remote Sensing Digital Image Analysis, Fifth Edition.* Springer-Verlag Berlin Heidelberg, 2013.

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

[36] A. Rosenfeld and M. Thurston. Edge and curve detection for visual scene analysis. *IEEE Transactions on Computers*, C-20(5):562–569, May 1971.

[37] G. J. Scott, M. R. England, W. A. Starms, R. A. Marcum, and C. H. Davis. Training deep convolutional neural networks for land-cover classification of high-resolution imagery. *IEEE Geoscience and Remote Sensing Letters*, 14(4):549–553, April 2017.

[38] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *CoRR*, abs/1312.6229, 2013.

[39] Christos Stergiou and Dimitrios Siganos. neural networks, 2017.

[40] Richard Szeliski. *Computer Vision: Algorithms and Applications*. Springer-Verlag New York, Inc., New York, NY, USA, 1st edition, 2010.

[41] Irene Mrquez Moreno y Amrica Molina del Villar Virginia Garca Acosta. *Los sismos en la historia de Mxico. Tomo II. El anlisis social*. FCE/UNAM/CIESAS, 2001.

[42] Michael Xie, Neal Jean, Marshall Burke, David Lobell, and Stefano Ermon. Transfer learning from deep features for remote sensing and poverty mapping. *CoRR*, abs/1510.00098, 2015.

[43] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? *CoRR*, abs/1411.1792, 2014.

[44] X. Zhou and S. Prasad. Active and semisupervised learning with morphological component analysis for hyperspectral image classification. *IEEE Geoscience and Remote Sensing Letters*, 14(8):1348–1352, Aug 2017.