

# Automatic Building Damage Assessment Using Deep Learning and Ground-Level Image Data

by

**Karoon Rashedi Nia**

B.Sc., Sharif University of Technology, 2015

Thesis Submitted in Partial Fulfillment of the  
Requirements for the Degree of  
Master of Science

in the  
School of Computing Science  
Faculty of Applied Science

© Karoon Rashedi Nia 2017  
SIMON FRASER UNIVERSITY  
Spring 2017

All rights reserved.

However, in accordance with the *Copyright Act of Canada*, this work may be reproduced without authorization under the conditions for “Fair Dealing.” Therefore, limited reproduction of this work for the purposes of private study, research, education, satire, parody, criticism, review and news reporting is likely to be in accordance with the law, particularly if cited appropriately.

# Approval

**Name:** Karoon Rashedi Nia  
**Degree:** Master of Science  
**Title:** *Automatic Building Damage Assessment Using Deep Learning and Ground-Level Image Data*  
**Examining Committee:** **Chair:** Greg Baker  
Senior Lecturer

**Greg Mori**  
Senior Supervisor  
Professor

---

**Anoop Sarkar**  
Supervisor  
Professor

---

**Ze-Nian Li**  
Internal Examiner  
Professor  
School of Computing Science

---

**Date Defended:** 20 January 2017

---

# Abstract

We propose a novel damage assessment deep model for buildings. Common damage assessment approaches require both pre-event and post-event data, which are not available in many cases, to classify damaged areas based on the severity of destruction. In this work, we focus on assessing damage to buildings using only post-disaster data in a continuous fashion. Our model utilizes three different neural networks, one network for pre-processing the input data and two networks for extracting deep features from the input source. Combinations of these networks are distributed among three separate feature streams. A regressor summarizes extracted features into a single continuous value denoting the destruction level. To evaluate the model, we collected a small dataset of ground-level image data of damaged buildings. Experimental results demonstrate that models taking advantage of hierarchical rich features outperform baseline methods.

**Keywords:** Building Damage Assessment; Regression with Neural Networks; Hierarchical Models

# Dedication

Dedicated to my mother, father, and sister.

# Acknowledgements

Firstly, I would like to express my sincere gratitude to my senior supervisor Dr. Greg Mori for the continuous support of my Masters study, for his patience, motivation, and demonstrating how to manage every aspect of life as efficiently as possible.

I would also like to thank Dr. Ze-Nian Li, Dr. Anoop Sarkar, and Dr. Greg Baker for serving as my thesis committee.

In addition, I would like to thank Dr.-Ing. Ali Asghar Nazari Shirehjini who provided me with opportunities to further my capabilities.

I would like to thank my brilliant friends all around the world, including Mahfam (her name describes her nicely), Armaghan, Mehrnoosh, Nazanin, Saman, Shokoofeh, Kourosh, Saeed, Hossein, Sepehr, Mani, and Mehran.

Finally, I would like to thank my mother, father, and sister, who always supported me, whatever path I took. You were always there for me.

# Table of Contents

<b>Approval</b>	<b>ii</b>
<b>Abstract</b>	<b>iii</b>
<b>Dedication</b>	<b>iv</b>
<b>Acknowledgements</b>	<b>v</b>
<b>Table of Contents</b>	<b>vi</b>
<b>List of Tables</b>	<b>viii</b>
<b>List of Figures</b>	<b>ix</b>
<b>1 Introduction</b>	<b>1</b>
<b>2 Previous Work</b>	<b>4</b>
2.1 Damage Detection Using Aerial/Satellite Imagery . . . . .	4
2.2 Object Recognition and Classification . . . . .	5
2.2.1 Object Classification Datasets . . . . .	5
2.2.2 Object Recognition Methods . . . . .	7
2.3 Semantic Segmentation . . . . .	12
2.3.1 Semantic Segmentation Datasets . . . . .	12
2.3.2 Semantic Segmentation Methods . . . . .	14
<b>3 Proposed Approach</b>	<b>18</b>
3.1 Color Image Feature Stream . . . . .	20
3.2 Color Mask Feature Stream . . . . .	20
3.3 Binary Mask Feature Stream . . . . .	21
3.4 Regression . . . . .	22
<b>4 Experiments</b>	<b>24</b>
4.1 Dataset . . . . .	24
4.1.1 Data Collection . . . . .	24

4.1.2	Data Annotation . . . . .	25
4.2	Training Procedures and Results . . . . .	27
4.2.1	Data Preparation . . . . .	27
4.2.2	Evaluation . . . . .	28
4.2.3	Best-Performing Models . . . . .	31
<b>5</b>	<b>Conclusion and Future Work</b>	<b>35</b>
5.1	Limitations and Future Directions . . . . .	35
	<b>Bibliography</b>	<b>37</b>

# List of Tables

Table 4.1	Classification of damage to buildings. . . . .	26
Table 4.2	Evaluation of proposed models. All the VGG networks are pretrained on ImageNet [28] and LeNets are pretrained on MNIST. The dimension of color image and color mask features is 4096 and the dimension of binary mask features is 500. . . . .	29
Table 4.3	Training settings. “lr” stands for learning-rate and “fc” stands for fully-connected layer. “base lr” applies to all the layers, except the last fc layer. Settings are ranked relatively based on their lr. . . . .	32
Table 4.4	Results of further experiments conducted on our two best-performing models, pretrained VGG on ImageNet and fine-tuned on color image and color mask training sets as the best-performing model and its combination with pretrained LeNet on MNIST and fine-tuned on binary mask training set as the second best-performing model. We trained them using different learning rates described in Table 4.3. . . . .	33
Table 4.5	Examples of our best-performing model, color image + color mask feature streams trained by medium lr in Table 4.4. Left two columns are inputs to the algorithm, the third column is ground-truth, and the last column is the estimated damage. . . . .	34



# List of Figures

Figure 1.1	Given only post-disaster data we aim to output a continuous value representing the damage severity. First column represents the input to our algorithm which is non-temporal post-disaster images. The second column denotes extracted features which will be used by the regressor (the third column) to output a continuous damage level. .	2
Figure 2.1	Using pair of images to extract features, and having human-observer to provide labels for training a linear classifier [19] (Copyright © 2015, IEEE). . . . .	4
Figure 2.2	Comparison of the "cat" and "cattle" subtrees between ImageNet and ESP [44]. Number of images for the largest node of each subtree is shown [28] (Copyright © 2009, IEEE). . . . .	7
Figure 2.3	Scale-space extrema detection by comparing a pixel (marked with X) to its 26 neighbors [29] (Copyright © 2004, Kluwer Academic Publishers). . . . .	8
Figure 2.4	$4 \times 4$ gradient window around keypoints is taken and histogram of gradients in 8 directions is computed. Image from: Jonas Hurrelmann.	8
Figure 2.5	Inception module, naïve version [42] (Copyright © 2015, IEEE). . .	10
Figure 2.6	The approximation of a sparse structure with spatially repeated dense components, convolutional filters with different sizes, and using dimension reduction by $1 \times 1$ convolutions to keep the computational complexity in bounds [42] (Copyright © 2015, IEEE). . . . .	11
Figure 2.7	Residual learning. left: plain net, right: residual net [20] (Copyright © 2016, IEEE). . . . .	11
Figure 2.8	Comparison of number of annotated images per category for COCO [12] and PASCAL [13], [12] (Copyright © 2014, Springer International Publishing Switzerland). . . . .	13
Figure 2.9	Annotation of objects and objects parts in ADE20K dataset. The first row shows the sample image, the second row shows the annotation of objects, and the third row shows the annotation of object parts [47] (arXiv preprint arXiv:1608.05442, 2016). . . . .	13

Figure 2.10	Frequency of object parts per each category [47] (arXiv preprint arXiv:1608.05442, 2016). . . . .	14
Figure 2.11	MCG performs hierarchical segmentation on different image resolutions and produces a ranked list of object candidates by combining these hierarchies [30] (Copyright © 2014, IEEE). . . . .	15
Figure 2.12	Overview of Simultaneous Segmentation and Detection algorithm [17] (Copyright © 2014, Springer International Publishing Switzerland). . . . .	15
Figure 2.13	Transforming fully connected layers into convolutional layers enables a classification net to output a heatmap [36] (Copyright © 2015, IEEE). . . . .	16
Figure 2.14	A fully convolutional network structure is employed to extract features, a bilinear interpolation stage up-samples the feature map to the image resolution. A fully-connected CRF refines the segmentation [8] (arXiv preprint arXiv:1606.00915, 2016). . . . .	17
Figure 3.1	Overview of our proposed model. (1): Color image feature stream (section 3.1). A deep structure directly analyzes the input image data. (2): Color mask feature stream (section 3.2). A deep structure analyzes color masks of the image data. (3): Binary mask feature stream (section 3.3). A Different deep structure is employed on the binary masks of the input data. The regressor utilizes extracted features to estimate the level of destruction. . . . .	19
Figure 3.2	VGG [37] network structure with some modifications. The network contains 13 convolutional layers followed by 3 fully-connected layers. A Sigmoid and a Euclidean loss layer are appended as the regressor. . . . .	20
Figure 3.3	The first row: raw images which are used by the color image feature stream pipeline. Second row: corresponding color masked images as the input to the color mask pipeline. Instances of greenery and the sky are omitted. Hence, the model is able to focus on relevant parts. . . . .	21
Figure 4.1	An example of an inconsistency in Virtual Disaster Viewer labels. The left is labeled as total destruction, however, the right one, which looks more damaged, is labeled as moderate damage. We manually corrected all of such examples. . . . .	25

# Chapter 1

## Introduction

Numerous tragic and hazardous natural disasters threat vulnerable areas of the world each year. Earthquake represents the highest mortality rate among natural disasters [4]. Although developing early warning systems is not feasible using current technologies, it is possible to utilize advanced computer vision approaches to manage post-event rescue and reconstruction plans efficiently. Computer vision based automatic damage assessment systems rely on optical data, which are processed to detect damaged areas and assess the level of destruction. According to the available data, these approaches either use only post-event data or both pre- and post-event images. In this work, we concentrate on the former.

Popular approaches that address automatic damage assessment use both pre-event and post-event aerial data to train classifiers (e.g. [5, 19]). These algorithms classify damages into pre-defined categories. However, in our case we are interested in carrying out building damage assessment in a continuous fashion rather than classifying it, using only post-disaster data. To this end, we require a dataset of damaged buildings annotated with the level of damage, a feature extraction method to extract robust features out of post-disaster image data, and a regressor to summarize all the extracted features into a single continuous value representing the level of destruction. In the following sections of this chapter, we describe how we handle aforementioned challenges. An overview of our model is illustrated in Fig. 1.1, in which the model estimates the damage level using non-temporal data.

To our knowledge there is no publicly available dataset of ground-level images of areas affected by natural disasters and annotated with consistent labels. To deal with this problem, we collected a small dataset and annotated each instance with a label denoting the damage severity. In order to train and evaluate our proposed models, the dataset is split into two distinct sets, training and testing sets.

Having the dataset of damaged buildings, we extract features from the training set and train a regressor to perform the damage assessment. Features are the essential factor to our system, as Girshick et al. [11] state “features matter”, and we have to collect them

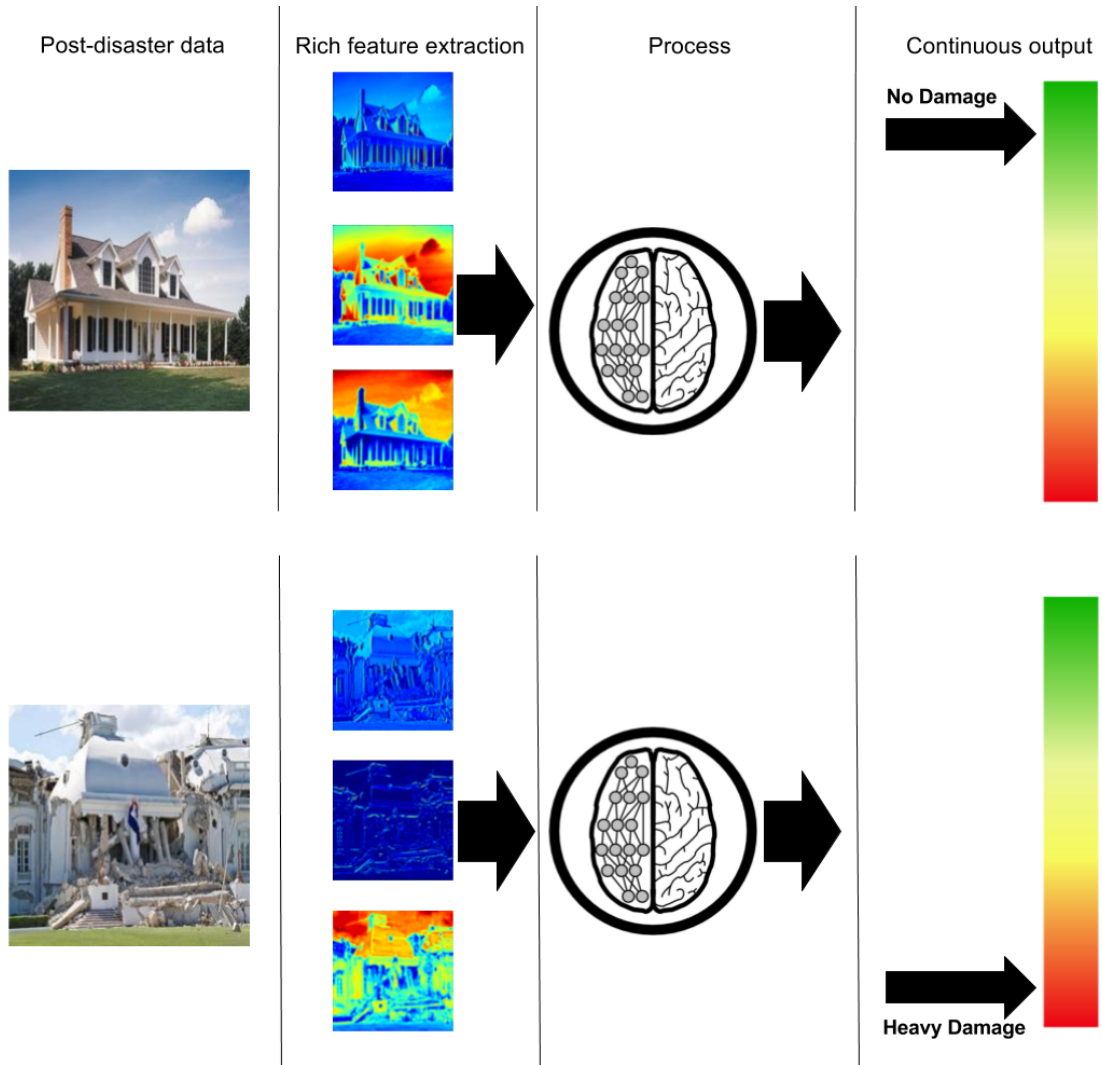


Figure 1.1: Given only post-disaster data we aim to output a continuous value representing the damage severity. First column represents the input to our algorithm which is non-temporal post-disaster images. The second column denotes extracted features which will be used by the regressor (the third column) to output a continuous damage level.

properly. Inspired by the recent achievements of deep structured models in different areas of computer-vision, we consider utilizing deep networks to extract robust features from the input source.

We construct three separate pipelines using different convolutional neural networks (ConvNets). Each pipeline is designed to perform a specific visual analysis. The first pipeline analyzes raw input images using a ConvNet and extracts features from the color images. Two other pipelines require a pre-processing step on the input. The input may contain irrelevant information to the context of our task, such as greenery, pedestrians, and cars. These distractions affect the performance of damage assessment system negatively. In the second pipeline, we attempt to focus only on the advantageous segments of the input. For this purpose, we define a set of objects which we consider as relevant, such as wall, roof, and doors, and employ a deep segmentation method to output all the instances of our defined set and consider the rest as background. Afterward, a different ConvNet is employed on the color masks, which hold the objects of interest, and extracts features from these objects. Moreover, in the third pipeline, we provide additional features on how intact and destructed buildings look like. To this end, in the third feature stream, the same segmentation method produces binary mask of objects appear in our defined set and a different ConvNet traverses these binary masks and extracts features. The intuition behind learning the shape of objects is LeNet [27], which recognizes handwritten digits using the binary representation of them. At last, the regressor, which is stacked fully-connected layer, a Sigmoid function, and a cost function, is employed on the extracted features of three streams and outputs a continuous value representing the level of damage.

The main contribution of this thesis is the study of segmentation methods as an advantageous factor involved in the damage detection and evaluation systems. In this work, we consider utilizing semantic segmentation paradigm to improve the performance of proposed algorithm. Beside directly analyzing post disaster data as the only input source to the algorithm, a semantic segmentation method is employed on these data to collect relevant objects to building damage assessment. Results indicate that semantically segmenting input data into objects of interest as foreground and the rest as background affects the performance positively.

# Chapter 2

## Previous Work

Damage detection and assessment is a well-studied topic and has been explored in multiple research areas. Solutions to computer vision problems, such as classification and segmentation, are the essential components of functional damage assessment systems. There is substantial literature in aerial image analysis, object recognition, and semantic segmentation. In section 2.1 we review computer vision-based algorithms performing damage assessment. These algorithms analyze aerial images as a source providing information on destruction level. In section 2.2 we go over some of the object classification datasets as well as algorithms processing them. In section 2.3 we cover segmentation datasets and algorithms attempting to segment these datasets semantically. The objective of this work is to assess damages to buildings of areas undergoing a catastrophe, such as an earthquake or a typhoon.

### 2.1 Damage Detection Using Aerial/Satellite Imagery

Each year, several natural disasters, such as an earthquake or a typhoon, impact broad areas of the world. Immediately responding to these catastrophes plays a significant role in rescuing affected population. Due to inaccessibility of such areas, satellite and aerial imagery are a valuable source of data for estimating the impact of calamity.

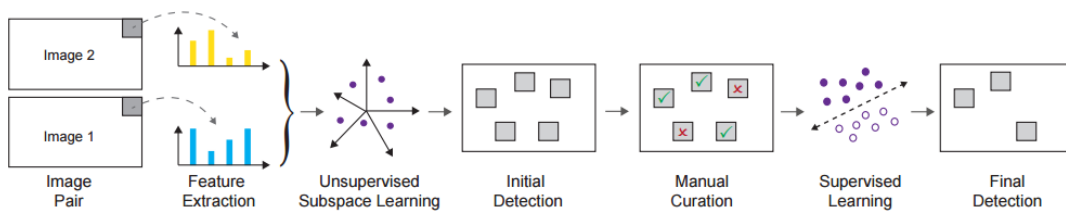


Figure 2.1: Using pair of images to extract features, and having human-observer to provide labels for training a linear classifier [19] (Copyright © 2015, IEEE).

Gueguen et al. [19] propose a semi-supervised framework to detect large-scale damages caused by catastrophes. They collected a dataset of 86 pairs of pre-event and post-event satellite imagery of impacted areas. For each pair, they extract features of  $50 \times 50$  window by using tree-of-shapes [31] as a descriptor of shape. Based on these features the algorithm clusters areas, which then used by human observers to obtain feedback on them. They use this feedback to train a linear Support Vector Machine (SVM). The algorithm is illustrated in Fig. 2.1.

Thomas et al. [5] propose an automatic damage assessment approach using pre-event and post-event aerial images. At the first stage, since the camera’s position, orientation, and field of viewed may change, they developed an algorithm to match pre/post-event images. SIFT feature detector [29] is used to extract interest points. K-Nearest Neighbor and Euclidean distance with a defined threshold are utilized to match detected points. Having corresponding points, their algorithm computed homographic transformation matrix. In order to segment buildings, they used maximum likelihood classification, and k-mean clustering to improve the results of the classifier. To estimate damage, they consider the roof structure, such as removal of tiles, holes in the roof or collapsed roofs, by using Canny edge detector to measure an increase in the number of edges, as a sign of damage. At the final stage, they use edge features and decision trees [7] to assess the damage.

## 2.2 Object Recognition and Classification

Object recognition is a widely-studied problem. Several attempts have been made to address this problem. Extracting features using feature descriptors, such as SIFT [29] and HOG [10], and applying classifiers, such as SVM, have brought advancement to object recognition algorithms. However, during the past few years, achievements of neural networks drew the attention of computer vision community to new research directions. In this work, by considering buildings and damages to them as objects, we utilize developed algorithms for recognition and classification to perform the building damage assessment.

### 2.2.1 Object Classification Datasets

Computer vision and machine learning research community utilize small/large-scale datasets in order to train and evaluate their proposed algorithms. Our proposed damage detection system, takes advantage of both small and large-scale datasets. We collected and annotated a small set of images of damaged structures using the experience of other researchers in collecting datasets. In the following section we review some well-labeled datasets.

## Small Image Datasets

Many of computer vision algorithms utilize well labeled small datasets as training and evaluation benchmarks. Caltech101 [14] contains 101 object categories. Each category contains 45-400 images which are collected via the Google image search engine. Caltech256 [18] is collected in a similar manner as Caltech101. It has 256 categories, and each category contains between 80 and 600 images. MSRC [33] is made up of 591 photographs of 21 object classes. The dataset is randomly broken into 3 pieces, 45% training, 10% validation and 45% test sets.

## Large-Scale Image Datasets

Due to recent advancement of computer vision research, larger and more challenging datasets are a crucial part of developing and evaluating more complex algorithms.

TinyImage [40] is obtained through the internet and image search engines. All words in WordNet [24], a lexical database of English words which organizes them into sets of synonyms called synset, are used as queries to the search engines and collected an average of 1000 images for each synset. The dataset contains 80 million  $32 \times 32$  low-resolution images, in which only 10-25% are possibly clean. Due to high level of noise and low-resolution images, the dataset is not appropriate for general purpose algorithms; however, the dataset has performed successfully in specific applications.

The ESP [44] is a game which is used to acquire a large dataset called ESP. The ESP game encourages two players, in a limited time, to independently suggest labels to one image with the goal of having as many identical words as possible. This game collected millions of labeled images (a subset of this collection is publicly available). In the limited time, players tend to propose simple and basic labels for each image. However, more specific labels (e.g. sparrow is more precise than a general category such as bird) may be necessary for some applications.

ImageNet [28] is a large-scale publicly available dataset of over 15 million accurately labeled images. ImageNet is built upon the semantic structure of the WordNet [24], synsets of words are internally linked by several types of relations like "IS-A", and covers over 5000 synsets of around 80,000 noun synsets in WordNet. ImageNet is acquired in two main phases, collecting candidate images and cleaning them. Over 10K candidate images for each synset are obtained through the several image search engines, by querying set of WordNet [24] synonyms and their translation into other languages, such as Chinese, Spanish and Dutch. Afterward, by using Amazon Mechanical Turk (AMT), human users verified whether each image comprises objects of the synset. To ensure the accuracy of labeling process, multiple users were asked to label the same image independently, and the image was considered positive if the majority of user votes were positive.



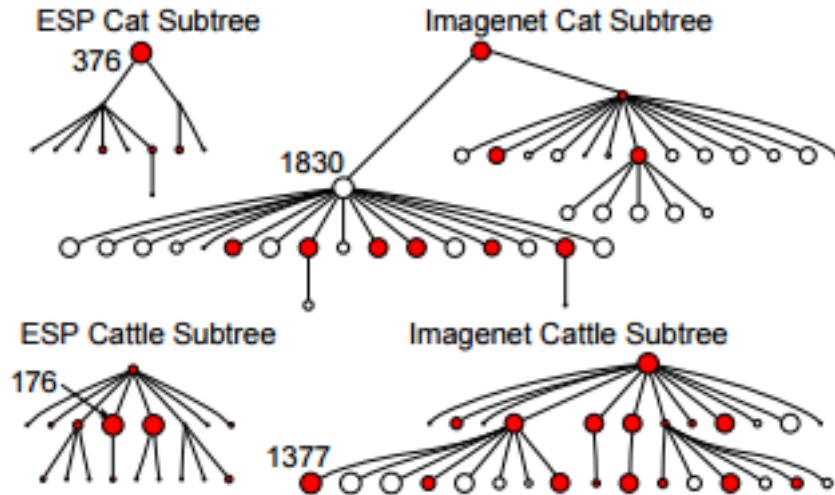


Figure 2.2: Comparison of the "cat" and "cattle" subtrees between ImageNet and ESP [44]. Number of images for the largest node of each subtree is shown [28] (Copyright © 2009, IEEE).

By this process, each synset on average contains over 600 accurately labeled images. Fig. 2.2 depicts the scale and diversity of ImageNet compare to ESP [44] for two sample categories. Several object classification approaches (e.g. [39, 37, 20]) leveraged scale, accuracy, and diversity of the ImageNet.

## 2.2.2 Object Recognition Methods

Object recognition is the convergence point of robotics, machine learning and computer vision, which enables artificial intelligence products to detect and identify objects in an image or video. This task is straightforward for humans, however, due to multiple factors, including viewpoint, illumination, scale, and occlusion, is a challenge for computer vision systems. Many approaches have been developed to address this challenge. In the following sections we review some feature-based and deep learning methods in recognition and classification.

### Feature-Based Methods

In machine learning and computer vision, defining informative, discriminating, and independent features is a crucial step toward developing algorithms in classification and regression. Sift [29] and HOG [10] are examples of such features, which enhanced computer vision algorithms in the past few years.

Scale Invariant Feature Transform (SIFT) [29] transforms image data into stable features which are invariant to image scaling and rotation, by detecting keypoints in an image and

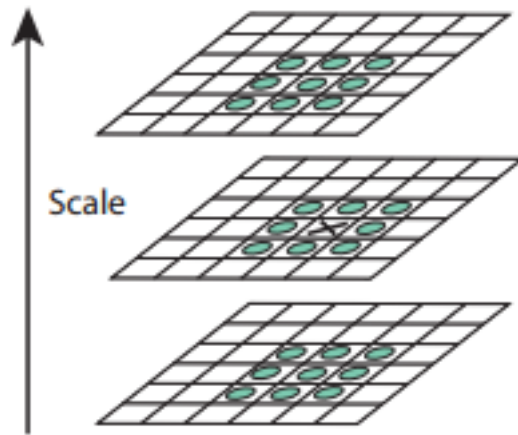


Figure 2.3: Scale-space extrema detection by comparing a pixel (marked with X) to its 26 neighbors [29] (Copyright © 2004, Kluwer Academic Publishers).

representing them by SIFT feature descriptor. The stable features are computed in four major stages, scale-space extrema detection, keypoint localization, orientation assignment and keypoint descriptor.

The method chooses keypoints by computing difference-of-Gaussian images, and detecting maxima and minima by comparing each pixel to its neighbors in scale-space as we can see in Fig. 2.3. Further refinement of detected keypoints, choosing true extrema from candidates, is done by applying Taylor expansion of the difference-of-Gaussian, taking derivative and setting it to zero. For each remaining point, the method chooses an area ( $4 \times 4$ ) around it and computes gradient magnitude and orientation (8 directions) for points fall in that area, and creates orientation histogram. Finally, a keypoint descriptor, a 128 dimensional

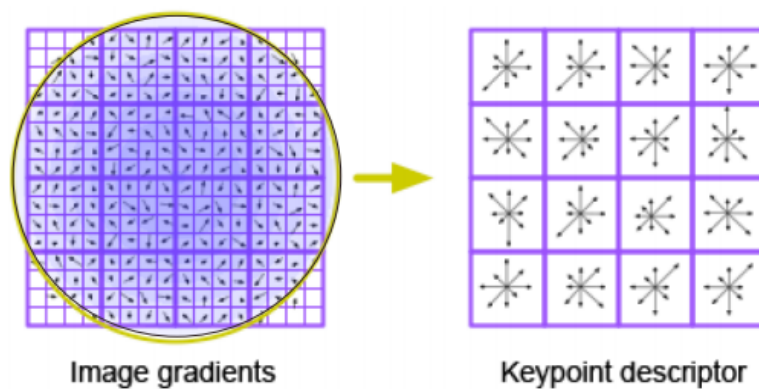


Figure 2.4:  $4 \times 4$  gradient window around keypoints is taken and histogram of gradients in 8 directions is computed. Image from: Jonas Hurrelmann.

feature vector ( $4 \times 4 \times 8$ ), is computed using gradient orientation and magnitudes as shown in Fig. 2.4. SIFT method can be used for object recognition task. Feature descriptor vector of the query image can be compared to the descriptors of the training database. The nearest neighbor in the training set, using minimum Euclidean distance, can be the best candidate match representing object category.

Exploiting gradients as robust feature descriptors has been used for individual object recognition tasks. Histograms of Oriented Gradients (HOG) [10] utilizes these features for human/non-human classification task. The fundamental idea is edge directions and the distribution of local intensity gradients can be used as object appearance and shape representation. HOG divides each image into blocks of cells, in which each cell contains groups of pixels. The size of cells and groups has been determined by evaluating performance of different settings. The feature descriptor is computed by accumulating the orientation of the gradients of cells and pixels. A linear SVM is then employed on extracted feature vectors in order to classify image into human/non-human classes.

## Deep Learning Methods

Variants of convolutional neural networks (ConvNets) are being used in different computer vision applications, including classification and regression. Construction of ConvNets follows conventional patterns, stacked convolutional layers followed by max-pooling and fully-connected layers. LeNet-5 [27] and ResNet [20] are examples of simplest and most complex network structures to-date which follow the same pattern.

Large-scale publicly available datasets (e.g. [28, 16]) and high-performance hardware devices such as GPUs, significantly contributed to the recent success of ConvNets. However, recent achievements are not just the outcome of larger datasets and powerful hardware. New ideas, improved algorithms, and advanced network structures play significant role in this new trend. Inspired by the great success of deep learning in classification and regression, our model relies on multiple ConvNets for extracting independent features related to damaged parts and predicting a continuous dependent value.

AlexNet [39], the winner of ImageNet Large Scale Visual Recognition Competition (ILSVRC) [34], was the first network that popularized ConvNets in computer vision. It consists of 5 convolutional layers (some of them followed by max-pooling layers), 3 fully-connected layers, a softmax output layer, and 60 million parameters. Dropout and data augmentation are used in order to reduce over-fitting in the fully-connected layers. The successful structure of AlexNet inspired other researchers to improve this architecture in order to achieve better accuracy. Zeiler et al. and Sermanet et al. [46, 15], best submissions to the ILSVRC-2013, made use of smaller receptive window size and smaller stride.

VGGNet [37] showed that the depth of the network (number of stacked convolutional and fully-connected layers) significantly impacts on the performance. VGG contains 16 convolutional and fully-connected layers, which uses small-size convolution filters ( $3 \times 3$ )

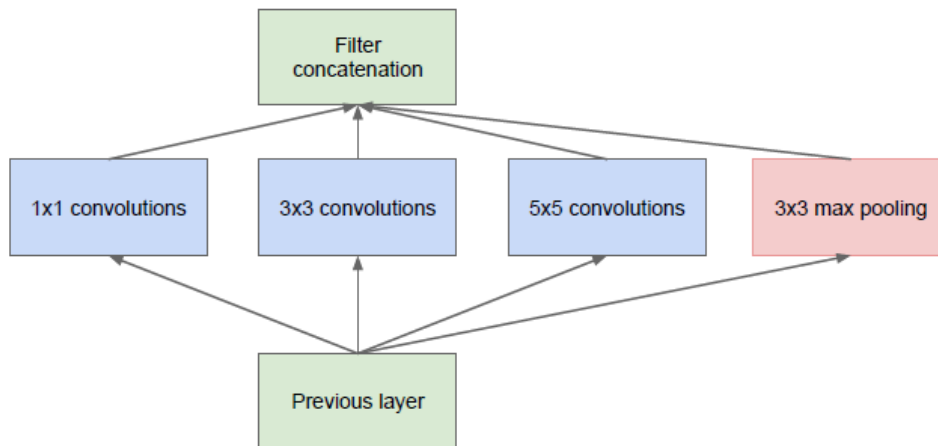


Figure 2.5: Inception module, naïve version [42] (Copyright © 2015, IEEE).

and  $2 \times 2$  pooling from the beginning to the end. VGG roughly has 140 million parameters which make it expensive to evaluate. Constructing deeper network structures results in having more complex and accurate models. However, enlarged networks are prone to overfitting and vanishing/exploding gradients, due to the great number of parameters, as well as they require more powerful hardware devices to be trained and evaluated.

To make constructing deeper networks feasible, GoogLeNet [42], the ILSVRC 2014 winner, developed a module called Inception, which dramatically reduced the number of parameters used in GoogLeNet (4 million, compare to AlexNet with 60 million and VGGNet with 140 million). The Inception architecture optimally approximates the sparse structure of a convolutional neural network by dense components (shown in Fig. 2.6). The intuition behind this structure is that, visual information should be processed at different scales which makes the next stage able to aggregate and abstract features simultaneously from various scales. The naïve version of Inception module, employs filter sizes  $1 \times 1$ ,  $3 \times 3$  and  $5 \times 5$  and concatenates their outputs into a single output vector as the input of the next stage (Fig. 2.5). This results in a large number of outputs which is extremely expensive to compute. The second structure of Inception module, reduces dimension, by using  $1 \times 1$  convolution filters prior to  $3 \times 3$  and  $5 \times 5$  filters as illustrated in Fig. 2.6. This architecture implies that even low dimensional filters represent a lot of information about an image, as well as makes it feasible for increasing both width and depth of the Inception networks and controlling computational complexity. GoogLeNet, the Inception architecture submitted to ILSVRC, contains 22 layers, using traditional convolutional components at lower layers and Inception modules at higher layers. Additionally, it uses average pooling before the classifier instead of a fully-connected layer, results in eliminating plenty of parameters and increasing accuracy.

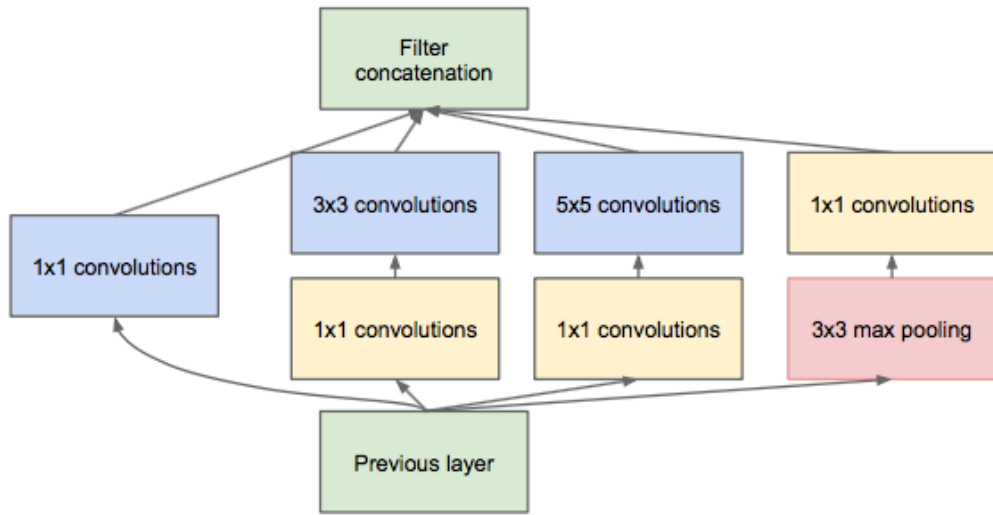


Figure 2.6: The approximation of a sparse structure with spatially repeated dense components, convolutional filters with different sizes, and using dimension reduction by  $1 \times 1$  convolutions to keep the computational complexity in bounds [42] (Copyright © 2015, IEEE).

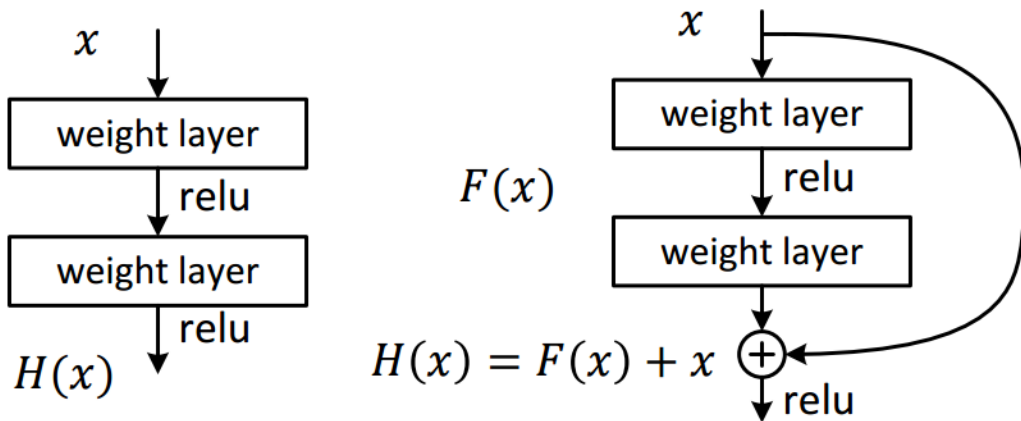


Figure 2.7: Residual learning. left: plain net, right: residual net [20] (Copyright © 2016, IEEE).

Residual networks [20] are state of the art in multiple computer vision tasks including image classification and segmentation. The ResNet structure takes advantage of considerably increased depth. The ILSVRC [34] 2015 winner contains 152 layers which is  $8\times$  deeper than VGGNet [37] and still having lower time complexity. Due to their experiments, simply stacking more layers and constructing deeper networks result in higher training and testing errors. The solution is instead of hoping stacked layers fit the desired mapping ( $H(x)$ ), let the stacked layers fit another mapping ( $F(x) = H(x) - x$ ) called residual mapping shown in Fig. 2.7. These shortcut connections add neither time complexity nor extra parameters.

Additionally, ResNet utilizes batch normalization [21]. Since the inputs of each layer rely on all preceding layers, as the network becomes deeper, small changes to the network hyperparameters such as learning rate amplify and make the network hard to train. Using batch normalization improves regularization, greatly accelerates training process, and makes the model less sensitive to initialization .

## 2.3 Semantic Segmentation

The goal of semantic segmentation is to classify each pixel of an image into predefined classes of a training set. Semantic segmentation involves both classification and segmentation at pixel-level, which makes it a challenging task. In this work, we consider utilizing semantic segmentation algorithms to split the input data of our proposed algorithm into objects of interest (advantageous in damage assessment) and background (irrelevant to damaged parts). In the following section, We review a few semantic segmentation datasets and models built upon them.

### 2.3.1 Semantic Segmentation Datasets

Many datasets have been collected and used for the purpose of semantic segmentation. These datasets cover various types of object categories and provide different level of details. In this work, we use ADE20K [47] scene understanding dataset, which is annotated with objects related to our task.

The PASCAL VOC dataset [13] contains 20 object classes for the tasks of classification, detection and segmentation. Images of the VOC 2007 were obtained through the Flickr photo-sharing website, by querying a set of keywords for each of classes, such as bicycle, bike, and cycle. The fact that these photos were taken for personal interests not computer vision research, makes the dataset "unbiased". After removing duplicate samples, annotators were asked to annotate random images from a set of 500,000 downloaded photos. The annotation contains the class of object, a bounding box around it, a viewpoint of an object (front, rear, left, right or unspecified) and truncation which means the object is occluded or it extends outside the image. Additionally, a subset of images was annotated pixel-wise for the segmentation competition.



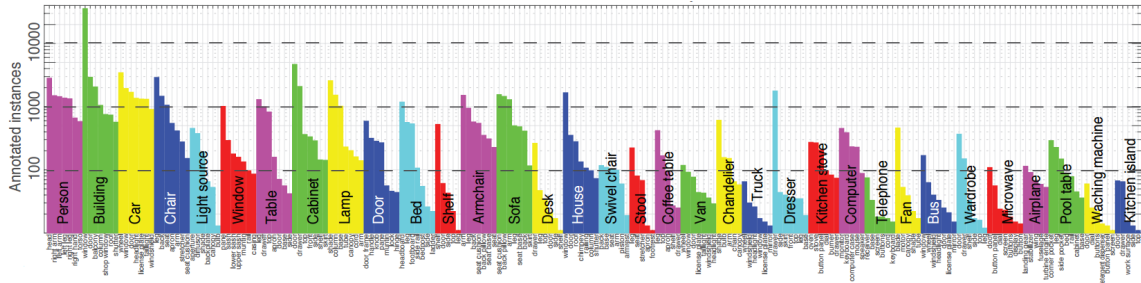


Figure 2.10: Frequency of object parts per each category [47] (arXiv preprint arXiv:1608.05442, 2016).

object categories to the dictionary as they appeared in new instances. They collected all the images from LabelMe [35], SUN datasets [32] and Places [41] in a way to cover all 900 scene categories defined in the SUN database, and a single expert worker annotated all the images. Instances are annotated densely with object segments with their names and object parts. More than 200 classes are annotated with object parts (Fig. 2.10). The broad range of covered objects as well as consistent and precise annotations provided by a single annotator, persuaded us to use the ADE20K dataset as the training set of a part of our developed algorithm.

### 2.3.2 Semantic Segmentation Methods

Analogous to object classification algorithms, traditional semantic segmentation methods relied on extracting hand-crafted features and applying linear classifiers, such as Support Vector Machines [43] or random forests [23]. However, the great success of convolutional neural networks changed the trend and transferred strength of ConvNets to the semantic segmentation task.

#### Interactive Segmentation Methods and Object Proposal Algorithms

There are interesting algorithms which perform semantic segmentation using the cut algorithm. These methods build a graph representing the image data and attempt to cut this graph in a way that objects of interest are separated from the background (e.g. [6, 25]). They either require a user interaction to segment image parts (e.g. [25]) or propose a pool of object candidates (e.g. [30]).

GrabCut [25] is an interactive foreground-background segmentation algorithm which optimizes Graph Cuts [6] for color images. GrabCut defines a new energy function which the minimum of it corresponds to a good segmentation and utilizes Gaussian Mixture Model for color images. The algorithm minimizes the energy function in an iterative process and requires user interaction to mark foreground and backgrounds.



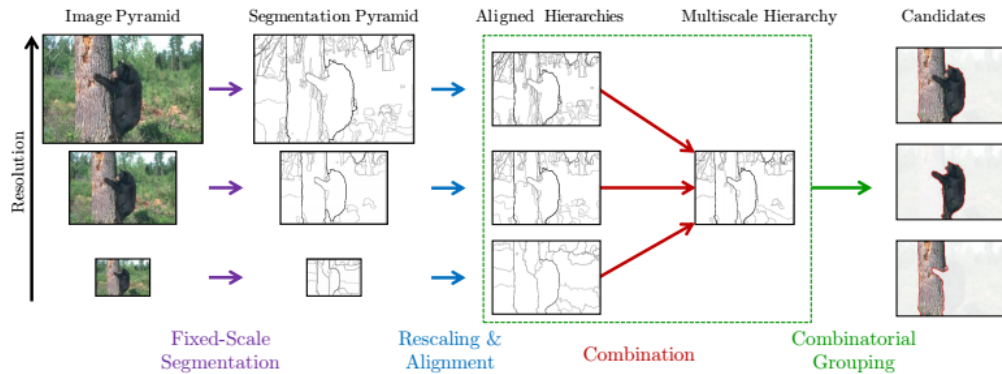


Figure 2.11: MCG performs hierarchical segmentation on different image resolutions and produces a ranked list of object candidates by combining these hierarchies [30] (Copyright © 2014, IEEE).

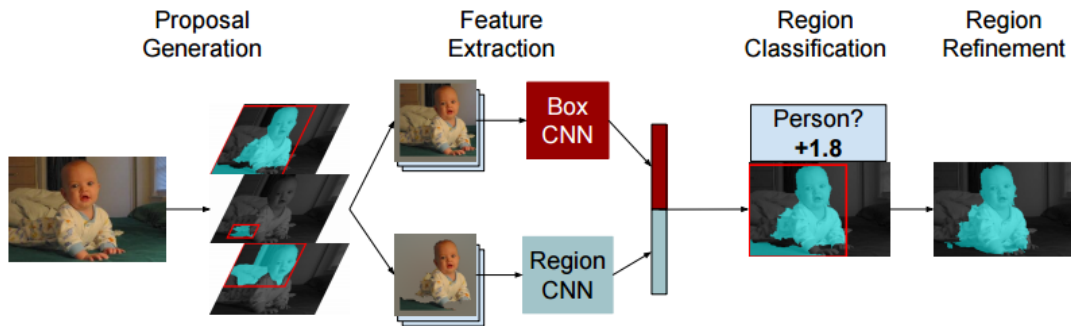


Figure 2.12: Overview of Simultaneous Segmentation and Detection algorithm [17] (Copyright © 2014, Springer International Publishing Switzerland).

Multiscale Combinatorial Grouping (MCG) [30] is an object proposal algorithm which generates a pool of ranked object candidates. It applies fast normalized cuts algorithm on different image resolutions in order to acquire contour maps of different scale images, and aggregates multiscale information to propose object candidates (Fig. 2.11).

### Deep Learning Methods for Semantic Segmentation

The first generation of ConvNet-based semantic segmentation approaches relies on classical semantic segmentation algorithms as part of their pipeline. For instance, Simultaneous Detection and Segmentation [17] applies MCG [30] to generate region candidates per image. Afterward, they use R-CNN [11] to extract features of both the bounding box and foreground regions of the proposed candidates. At the final step, a trained SVM classifies each region based on the extracted features. The pipeline is depicted in Fig. 2.12.

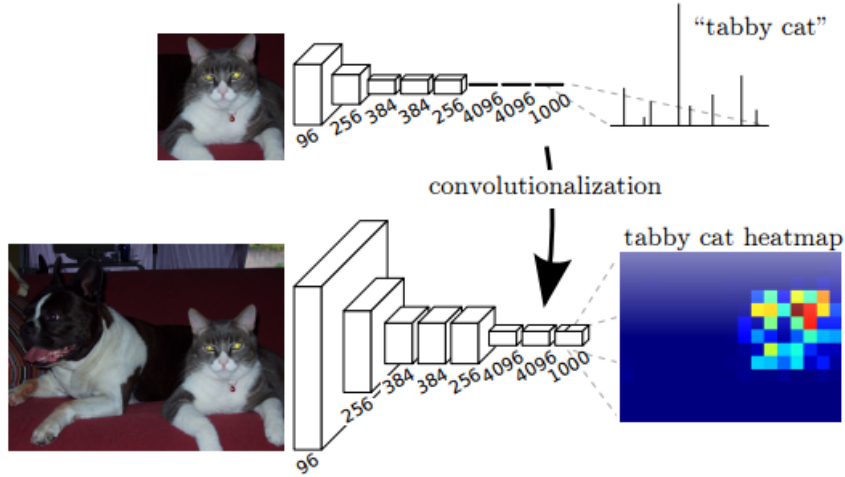


Figure 2.13: Transforming fully connected layers into convolutional layers enables a classification net to output a heatmap [36] (Copyright © 2015, IEEE).

Recently, deep ConvNets have been used directly to predict pixel-level classification. For this purpose, the fully-connected layers which were used to output probability scores are replaced by convolution layers (Fig 2.13), allows ConvNets to preserve spatial dimensions. Fully Convolutional Networks for Semantic Segmentation (FCN) [36] upsamples intermediate extracted features from convolutional layers to preserve dimensions. Using ground truth at pixel-level, made the loss function of the FCN able to sum over all spatial dimensions of the last convolutional layer and compute gradients, which made the feed-forward, backpropagation, and learning process straightforward.

DeepLab [8] and DilatedNet [45] are another examples of using deep ConvNets for dense prediction. They both use *atrous convolution* or *dilated convolution* in their fully convolutional network structures. The DilatedNet [45] replaces *pool4* and *pool5* of the VGG-16 [37] with dilated convolution. The DeepLab [8] utilizes *atrous convolution* with up-sampled filters, and fully-connected conditional random fields [26] in order to accurately segment along object boundaries. Fig. 2.14 illustrates the approach of DeepLab[8].

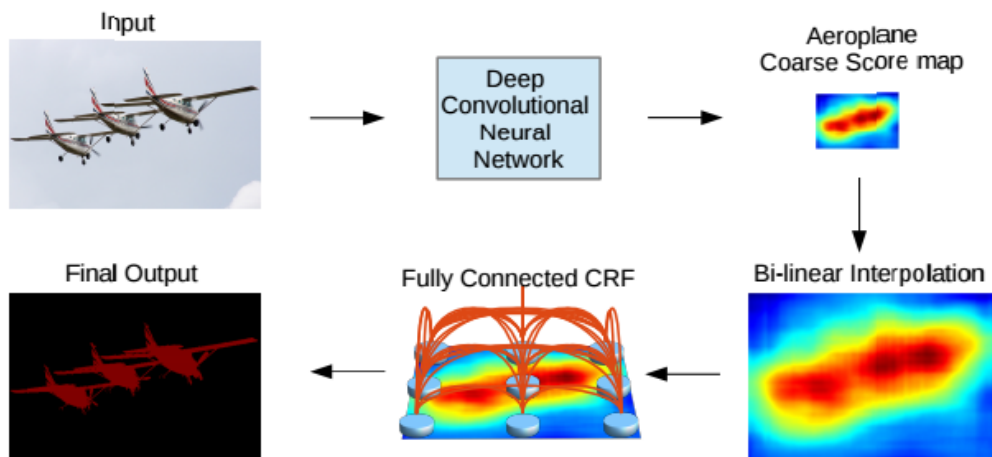


Figure 2.14: A fully convolutional network structure is employed to extract features, a bilinear interpolation stage up-samples the feature map to the image resolution. A fully-connected CRF refines the segmentation [8] (arXiv preprint arXiv:1606.00915, 2016).

## Chapter 3

# Proposed Approach

Our goal in this work is to automatically assess damages to buildings caused by natural disasters including earthquake and hurricane. Several attempts have been made to classify damaged areas and detect changes by employing both pre-event and post-event aerial images and extracting hand-crafted features as the input to linear classifiers such as SVM (e.g. [19, 5]). In this work, however, we extract rich features from post-event images, as the only input source to our algorithm, and output a continuous value as a factor measuring damage severity rather than classifying damages into predefined categories.

Inspired by the recent advances in deep learning, we leverage multiple convolutional neural networks (ConvNets) to propose a novel deep hierarchical model performing building damage assessment. Our model is made up of 3 feature streams, each represent attributes of the image data which are significant in damage assessment. Each pipeline comprises one or two ConvNets to extract rich features from the input data. Color image feature stream employs a deep structure (VGG [37]) to extract features from the raw input data (color image). Color mask feature stream applies a segmentation deep structure (DilatedNet [45]) to semantically segment raw input data into objects of interest and then utilizes a deep model (VGG [37]) to obtain features out of segmented images. The last pipeline, binary mask feature stream, attains binary masks of the objects of interest in image data using the same segmentation method (DilatedNet [45]) and utilizes a different deep structure (LeNet [27]) to analyze them. Having the extracted features, we could follow two directions. Using them to rank the instances (e.g. [9]) of damaged buildings based on the severity of damage, or summarize them into a single continuous value (regression) representing the level of destruction. In this work we perform regression using the extracted features. At last, the regressor exploits extracted features by 3 streams to assess the level of destruction. An overview of the proposed model is illustrated in Fig. 3.1. In the following sections of this chapter, we describe the behavior of each stream, their inputs, and corresponding outputs in detail. Moreover, we build several models based on these pipelines and evaluate their performance in chapter 4.

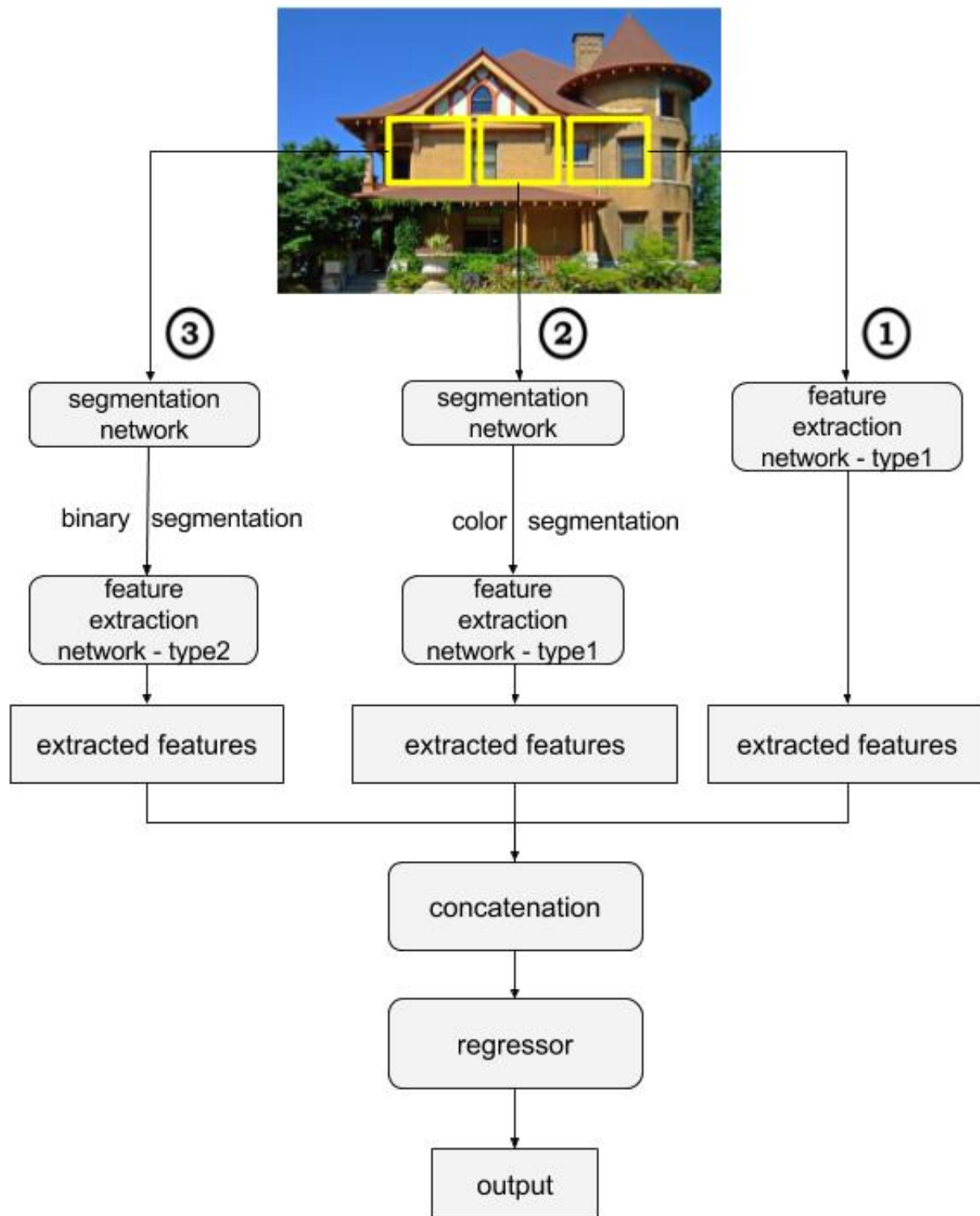


Figure 3.1: Overview of our proposed model. (1): Color image feature stream (section 3.1). A deep structure directly analyzes the input image data. (2): Color mask feature stream (section 3.2). A deep structure analyzes color masks of the image data. (3): Binary mask feature stream (section 3.3). A Different deep structure is employed on the binary masks of the input data. The regressor utilizes extracted features to estimate the level of destruction.

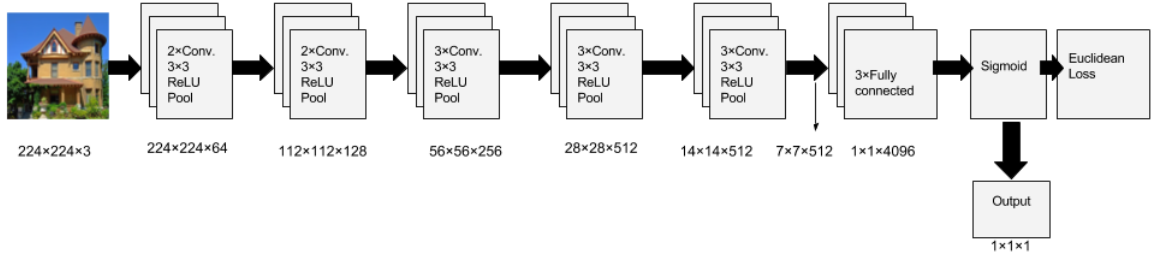


Figure 3.2: VGG [37] network structure with some modifications. The network contains 13 convolutional layers followed by 3 fully-connected layers. A Sigmoid and a Euclidean loss layer are appended as the regressor.

### 3.1 Color Image Feature Stream

Color image feature stream is the first pipeline of our proposed model. This pipeline is designed to analyze raw input data as opposed to two other pipelines which require a pre-processing step on the input data. Given an input image data  $X(\text{Width} \times \text{Height} \times \text{Channels})$ , which holds the raw pixel values of the image, several convolutional layers compute a dot product between their weights ( $W$ ) and a small region of the input they are connected to (known as *receptive field*) and apply their bias offset ( $b$ ) as shown in Equation 3.1. Afterward, elementwise activation function ( $ReLU$ ), shown in Equation 3.2, and a downsampling operation (known as *Pooling*) will be applied to the output of neurons. The network structure and dimensions of each volume is illustrated in Fig. 3.2.

$$z_j = f\left(\sum_i w_i x_i + b\right) \quad (3.1)$$

$$f(x) = \max(0, x) \quad (3.2)$$

Extracted features by convolutional layers are then processed by fully-connected layers, which hold neurons that are connected to all the neurons in the previous volume. The result is features of color images which then will be used by the regressor.

### 3.2 Color Mask Feature Stream

The second pipeline of our model is color mask feature stream. Several visual factors, including camera position and angle, could affect the performance of vision-based systems by adding extra objects to the image, such as sky, trees, and etc. These factors either slow down the learning process or have negative effects on the accuracy of vision-based algorithms. To address this issue in our work, we consider utilizing semantic segmentation



Figure 3.3: The first row: raw images which are used by the color image feature stream pipeline. Second row: corresponding color masked images as the input to the color mask pipeline. Instances of greenery and the sky are omitted. Hence, the model is able to focus on relevant parts.

algorithms to focus on the relevant regions in the input data rather than processing the whole image.

For this purpose, a deep structure (DilatedNet [45]) has been used to segment images into objects of interest as foreground and the rest as background. Given an input image, DilatedNet collects all the instances of a pre-defined object set, which we consider as relative, through an image and outputs those as foreground. A couple of output instances are shown in Fig. 3.3.

Similar to section 3.1, several convolutional layers followed by an activation function (*ReLU*) and a pooling layer traverse color masks and extract features. These features are then used by fully-connected layers to provide the regressor (section 3.4) with features of color masks. The feature extraction process is described in section 4.2.

### 3.3 Binary Mask Feature Stream

Inspired by LeNet [27], which recognizes handwritten digits using a simple convolutional neural network, we consider learning the shape of intact and destroyed buildings. The MNIST dataset [27], which has been used to train LeNet, contains black and white two-

dimensional vectors representing handwritten digits. Using MNIST, the LeNet successfully learned to distinguish between digits.

The third pipeline of our algorithm, binary mask feature stream, is designed to learn the shape of damaged building parts. To this end, DilatedNet [45] is used to produce binary masks of our pre-defined object set which we expect to be relevant to damage assessment. The process is similar to section 3.2, however, the foreground (objects of interest) is marked as white, and the background is marked as black. In order to extract features correspond to the shape of objects, LeNet [27] network structure is then employed on the binary masks. Given an input data  $X(Width \times Height \times 1$ , since the input is binary the dimension of channels is reduced to 1 as opposed to 3 in RGB images), two convolutional layers (Equation 3.1), each followed by a pooling layer which downsamples by a factor of 2, and a fully-connected layer using an activation function ( $ReLU$  in Equation 3.2), and another fully-connected layer, traverse the binary input and extract features to be used by the regressor (section 3.4).

### 3.4 Regression

Aforementioned pipelines provide hierarchical rich features in the feed-forward process. The goal is to learn to conclude a single continuous value out of these features. To this end, features obtained through the described streams are concatenated into a single feature vector. This vector is processed by a fully-connected layer, to infer a single value. A Sigmoid function (Equation 3.3) is then used to map the output to a range between 0 and 1, which we expect to be the level of destruction.

$$S(x) = \frac{1}{1 + e^{-x}} \quad (3.3)$$

Having the output, makes the model able to utilize a loss function in conjunction with an optimization method in the error back-propagation process to update the weights in a way that the loss is minimized.

In section 2.2 we mentioned several deep learning based classification methods. However, there is a difference between training a deep structure for classification and training it for regression. The loss function, which controls the learning process, in a regression network has to penalize outputs based on their distance to the ground truth. For example, for an unobserved data point equal to 2, a network which outputs 3 is performing better than a network which outputs 4, however, in classification task both outputs are considered equally wrong. To address this issue in our work, we employ Euclidean distance (Equation 3.5) as the loss function and stochastic gradient descent (SGD) as the optimization method. Assuming  $X_n = (x_1, \dots, x_D)$ ,  $Y_n = (y_1, \dots, y_D)$ , and  $Z_n = (z_1, \dots, z_{D'})$  as the extracted features of the  $n_{th}$  sample through the color image, color mask, and binary mask pipelines, we can formulate the process as the following optimization problem:



$$p_n = \phi(X_n \frown Y_n \frown Z_n) \quad (3.4)$$

$$\operatorname{argmin}_w \frac{1}{2N} \sum_{i=1}^N \|t_n - S(p_n)\|_2^2 \quad (3.5)$$

Where  $\frown$  stands for the concatenation,  $\phi$  is the applied function of the fully-connected layer to extracted features,  $N$  is the total number of training samples,  $t_n$  is the ground truth,  $S(p_n)$  denotes applying Sigmoid function (Equation 3.3) to the predicted label for the sample  $n$ , and  $w$  stands for all the learnable weights in convolutional and fully-connected layers.

During several iterations of training, our model has learned to perform the building damage assessment task for an unobserved image data. Experimental results and training procedures are provided in section 4.2.

# Chapter 4

## Experiments

### 4.1 Dataset

We collected a novel dataset of ground-level post-disaster images of buildings affected by natural disasters. All the images are obtained through the internet using three sources, Virtual Disaster Viewer [3], Open Images Dataset [2] and Google image search engine. We collected and refined images and labels through the different stages. There are 200 images in the final training set and 50 images in the final testing set. The collected images had different sizes. Due to memory and space considerations we downsampled all the instances to  $224 \times 224$  to be used by color image and color mask pipelines and  $28 \times 28$  to be used by binary mask pipeline. Moreover, by downsampling, we were able to take advantage of pretrained state of the art models which use the same image size.

#### 4.1.1 Data Collection

Virtual Disaster Viewer [3] is a web application utilizing Microsoft’s Virtual Earth, and containing images of distressed areas undergoing a calamity. It covers earthquakes of different parts of the world including 2015 Nepal and 2010 Chile. Some of the images are paired with meta-information, such as category and damage level. We acquired a part of our training and testing set from the Virtual Disaster Viewer using a Python script. However, due to the inconsistency of the labels, we have gone through the images and labels and have made modifications. An example of an inconsistency is illustrated in Fig. 4.1.

Open Images [2] is a large-scale Dataset consisting of about 9 million images annotated with labels covering 6000 real-life categories. On average, each image of the dataset is annotated with 8 labels. We used a Python script to search through the images of the Open Images Dataset. We defined a set including all existing labels of this dataset which are related to our task. Earthquake, building, historic house, outhouse, manor house, house, and skyscraper are considered as relevant labels. We went through the instances of each word in the set, and collected the related samples.



Figure 4.1: An example of an inconsistency in Virtual Disaster Viewer labels. The left is labeled as total destruction, however, the right one, which looks more damaged, is labeled as moderate damage. We manually corrected all of such examples.

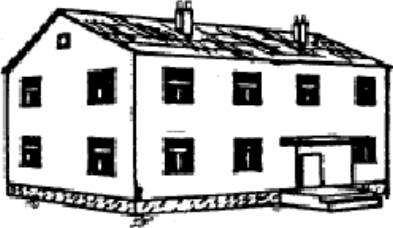




Moreover, we leveraged the Google image search engine in order to enlarge our dataset. We queried keywords such as earthquake, damaged buildings, damaged houses, and etc. as well as their translations into Persian to acquire more images.

#### 4.1.2 Data Annotation

As described in section 3.1 our goal is to compute a continuous value representing the severity of the damage. In order to train the proposed algorithm to accomplish aforementioned task, a dataset consisting of building images paired with damage estimation was required. To this end, we utilized damage classification scheme described in the damage assessment report of Haiti earthquake [1] to annotate each image in the training set with a value between 0 and 1 at a step size of 0.25. Greater values imply more destruction. All the images are annotated by myself.

The damage assessment report [1], classifies damage to buildings of Haiti, caused by earthquake 2010, based on visual attributes of walls, floors, and roofs. Inspired by their approach, we manually classified images of the dataset into five categories. No damage, slight damage, moderate damage, heavy damage, and total destruction classes are described with 0, 0.25, 0.5, 0.75, and 1 respectively. The labeling scheme is described in Table 4.1. Having described labels, made our algorithm able to learn visual features which are significant in the damage assessment process. Section 4.2 describes the learning procedure and evaluation in detail.

Table 4.1: Classification of damage to buildings.

Figure	Description	Label
	<p>No damage</p>	<p>0</p>
	<p><b>Slight damage</b> Negligible damage such as cracks in walls</p>	<p>0.25</p>
	<p><b>Moderate damage</b> Fall of pieces of walls and roofs</p>	<p>0.5</p>
	<p><b>Heavy damage</b> Serious failure of walls Partial structural failure of roofs and floors</p>	<p>0.75</p>
	<p><b>Total destruction</b> Total or near total collapse</p>	<p>1</p>

## 4.2 Training Procedures and Results

In this section, we provide details of training procedures and conducted evaluation experiments. We trained two different network structures (VGG [37] and LeNet [27]) in several varying settings to extract features from three different types of inputs (color image, color mask, and binary mask). Caffe framework [22] and stochastic gradient descent as the optimization method have been used to train deep networks.

### 4.2.1 Data Preparation

As the first step of training, we converted the dataset images and labels to HDF5 data models to be used by Caffe as the input source. Each image in the data model is paired with its label. These labels are used in the backward propagation of errors (backpropagation) to compute the gradient of Euclidean loss function with respect to all the weights. Stochastic Gradient Descent (SGD) uses the computed gradient to update the weights and minimize the Euclidean loss. In the evaluation stage, these labels are used to compute Euclidean distance which represents the performance of the algorithm.

The first type of the input source is color images of the dataset. We developed a Python script to read images and labels one by one and make the HDF5 database. This database is then used by Caffe to feed to the VGG network [37].

The second input type is the color mask of images. DilatedNet [45] which has been trained on ADE20K dataset [47] is able to segment 900 scene categories. In our case, we were interested in a small part of these classes. We collected a set of 29 out of 900 categories, as scenes which are related to our work. wall, building, floor, ceiling, road, windowpane, sidewalk, earth, door, house, field, rock, column, skyscraper, path, stairs, runway, screendoor, stairway, bridge, countertop, hovel, awning, booth, pole, land, banister, escalator, and tent are the objects which we considered as related. We wrote a Matlab code which takes images as input, feeds them to the DilatedNet and based on pixel-wise classification output, preserves the color of all pixels which are an instance of the defined set and makes the rest black. Sample outputs are shown in Fig. 3.3, which the foreground is colored and the background is black. We used the Python script to convert these color masks to HDF5 data model. VGG [37] network structure is then used to extract features from segmented images.

The third input type is the binary mask of the dataset images. Inspired by MNIST dataset [27], we created binary masks of our objects of interest using DilatedNet. The procedure is similar to creating colored masks, however, the Matlab code colors foreground as white and background as black. LeNet network structure [27] which was pretrained on MNIST is then employed to analyze binary images.

### 4.2.2 Evaluation

We conducted several experiments in order to evaluate the performance of our proposed algorithm. As illustrated in Fig. 3.1 the proposed method consisting of three main pipelines, color image feature stream, color mask feature stream, and binary mask feature stream. The color image feature stream utilizes VGG network and described color image as the input. The color mask feature stream employs color masked images as well as VGG network. The binary mask feature stream makes use of binary masked images and LeNet [27] network structure. Each pipeline is designed to perform a specific visual analysis. Various models are built based on the three streams. First, we explain extensive experiments conducted for the evaluation in Table 4.2, and provide analysis of each experiment in detail. Afterward, we introduce our best-performing model and few test samples in conjunction with their estimated damage.

The first model is our baseline, which is pretrained VGG [37] deep network on ImageNet [28] dataset. We replaced the last fully-connected layer of the original structure with a new fully-connected layer followed by a Sigmoid layer and Euclidean loss function. We initialized network weights with pretrained weights, and randomly initialized weights of the appended fully-connected layer. In the training time, all the layers up to the last fully-connected layer are frozen. Stochastic Gradient Descent (SGD) as the optimization method, updates the weights of the last fully-connected layer to minimize the loss function. The network has been trained for 10,000 iterations with weights learning rate equal to 0.001 and bias term learning rate equal to 0.002. The model is evaluated on our color image testing set. The average Euclidean distance between ground truths and network outputs is 0.38, which is the highest distance based on Table 4.2.

As for the next experiment, we modified the training procedure of the baseline to obtain an improved model. We consider both models as the baseline of our work. There is a training strategy in deep learning called fine-tuning. In this strategy, the model is initialized with pretrained weights on large-scale datasets such as ImageNet, and the optimization method updates pretrained weights very slightly based on the given dataset. Fine-tuned VGG utilizes the network structure of the baseline and fine-tuning strategy. Since our dataset is small but very different from the ImageNet, we considered training all the layers of the network with a small learning-rate, and training the last fully-connected layer with a higher learning-rate. The network has been trained for 10,000 iterations with base learning-rate equal to 0.0001, last fully-connected layer’s learning-rate equal to 0.001 and its bias term learning-rate equal to 0.002. This model achieved better test result (0.20) on the color image testing set compared to the naïve baseline (0.38).

Moreover, fine-tuned VGG is tested on color mask testing set. The result shows that its performance decreases when testing on color mask data (performance decreased from 0.20 to 0.28), however, the performance is better than the baseline. This behavior occurs due to

Table 4.2: Evaluation of proposed models. All the VGG networks are pretrained on ImageNet [28] and LeNets are pretrained on MNIST. The dimension of color image and color mask features is 4096 and the dimension of binary mask features is 500.

Model components			Model description	Euclidean distance
Color image feature stream	Color mask feature stream	Binary mask feature stream		
✓	—	—	VGG	0.38
✓	—	—	fine-tuned VGG on color image training set	0.20
—	✓	—	fine-tuned VGG on color image training set	0.28
—	✓	—	fine-tuned VGG on color mask training set	0.18
—	—	✓	fine-tuned LeNet on binary mask training set	0.33
✓	✓	—	fine-tuned VGG on color image training set	0.20
✓	✓	—	fine-tuned VGG on color image and color mask training sets	<b>0.17</b>
✓	—	✓	fine-tuned VGG on color image training set and fine-tuned LeNet on binary mask training set	0.20
—	✓	✓	fine-tuned VGG on color mask training set and fine-tuned LeNet on binary mask	0.19
✓	✓	✓	fine-tuned VGG on color image training set and LeNet	0.21
✓	✓	✓	fine-tuned VGG on color image training set and LeNet, reduced dimension version	0.21
✓	✓	✓	fine-tuned VGG on color image and color mask training sets, and fine-tuned LeNet on binary mask	<b>0.18</b>

fine-tuning the network on color image dataset and using learned weights to test on color mask dataset.

We attempt to fine-tune the VGG network on color mask training set, which is images acquired by applying DilatedNet on color image training set. We followed two different fine-tuning strategies. First, we fine-tuned the network by training the last fully-connected layer with higher learning-rate than other layers. In the second strategy, we trained the network in two phases. In the first phase, we trained the VGG in the same way as the first strategy. In the second phase, we replaced the last fully-connected layer with another fully-connected layer which has lower learning-rate than the original one. The second strategy outperforms the first one. This model achieved 0.18 on average on the testing set, which is provided in Table 4.2. Both phases have been trained for 10,000 iterations. The base learning rate of the first phase is 0.0001 and the last fully-connected layer’s learning rate for weights is 0.001 and for the bias term is 0.002. In the second phase, we divided both learning rates of the last layer by a factor of 2, 0.0005 for weights and 0.001 for the bias term.

In the next experiment, we modified LeNet [27] network structure for regression task. The last fully-connected layer is replaced by another fully-connected layer followed by a Sigmoid and a Euclidean loss layer. This model has been fine-tuned on binary mask training set for 10,000 iterations with base learning-rate of 0.0001, and 0.001 and 0.002 for weights and bias term learning-rates of the last layer. Since this model has a simpler network structure compared to VGG-based networks, as we expected the testing result is not as good as VGG-based models, however, it is better than the baseline.

Thus far, we evaluated various settings of single pipelines of our algorithm, illustrated in Fig. 3.1. Afterward, we considered combining these pipelines into more powerful models. Due to the complexity of constructed models by combining pipelines, we followed different training strategies to address time and memory concerns.

Our first model of this type is a combination of color image and color mask feature streams. In this model, we used pretrained VGG on ImageNet and fine-tuned it on the color image training set. In order to make our model able to be trained on a single GPU, we extracted features of color image and color mask training sets using the fine-tuned VGG on color images and saved them into separate HDF5 data models. Afterward, a single fully-connected layer followed by a Sigmoid and a Euclidean loss layer is trained, given the concatenation of extracted features. We trained this model for 10,000 iterations, with weights learning-rate (lr) equal to 0.001 and bias term lr equal to 0.002. In order to evaluate this model, we saved extracted features of color image and color mask testing sets into separate HDF5 data models and acquired test outputs using trained single fully-connected layer. This model acquired 0.20 on the testing time, which is slightly better than a single fine-tuned VGG.

In the next experiment, we followed the same training strategy, however, we utilized fine-tuned VGG on color mask training set to extract features of color mask data rather



than fine-tuned VGG on color images. This model achieved the best performance on our testing set, which is about 0.2 better than the baseline in a range between 0 and 1. This achievement is due to extracted rich features from both the images and their corresponding color masks.

In the next two experiments, we took advantage of both VGG and LeNet network structures. We extracted features of color image, color mask, and binary mask data sets. Combined color image feature stream with binary mask feature stream in one model, and color mask feature stream and binary mask feature stream in the other one. Additionally, VGG-based models output a feature vector of size 4069, and LeNet-based models output a feature vector of size 500, therefore, we matched the dimension of VGG features to LeNet features to prevent domination of larger features. Since the fine-tuned VGG on color mask feature stream performs better than the fine-tuned VGG on color image feature stream, their combinations with LeNet follow the same trend as well.

In order to practically evaluate whether larger features dominate the smaller ones, two experiments are conducted. In one of them color image feature stream of size 4096, color mask feature stream of size 4096 and binary mask feature stream of size 500 are used. In the other one, dimension of color image and color mask features are reduced to 500. Results (shown in Table 4.2) imply that the effect of dimension reduction is not significant, however, a slight improvement is observed.

In the last experiment, we combined all the three feature streams together. Color image, color mask, and binary mask features are extracted using fine-tuned VGG on color images and color masks and fine-tuned LeNet on binary masks respectively. Inspired by dimension-reduction experiment, the dimension of color image and color mask features are reduced to 500. Training features are extracted separately and used together to train a fully-connected layer followed by a Sigmoid layer as our regressor. We applied the learned weights to our testing set to evaluate the performance. The result shows that taking advantage of rich features extracted from fine-tuned VGG on color image and color mask training sets and fine-tuned LeNet on binary mask training set, makes this model as the second best-performing model on our dataset. However, if we used a stronger network structure rather than LeNet, performance of this model could be better.

### 4.2.3 Best-Performing Models

In the section 4.2.2 we introduced and described several deep learning models performing damage assessment task in detail. Table 4.2 comprises the comparison of proposed methods based on their average Euclidean distance on the testing set. In this section, we explain more experiments conducted on two best-performing models in order to improve the performance.

Having results of proposed methods, we considered training models with different settings. To this end, we chose the two models achieved best results, combination of color image feature stream and color mask feature stream as the best-performer as well as their combina-

Table 4.3: Training settings. “lr” stands for learning-rate and “fc” stands for fully-connected layer. “base lr” applies to all the layers, except the last fc layer. Settings are ranked relatively based on their lr.

<b>Setting</b> \ <b>lr</b>	<b>base lr</b>	<b>last fc weights learning rate</b>	<b>last fc bias term lr</b>
<b>high lr</b>	0.0001	0.001	0.002
<b>medium high lr</b>	0.0001	0.0005	0.001
<b>medium lr</b>	0.0001	0.0002	0.0005
<b>low lr</b>	0.0001	0.0001	0.0002











tion with binary mask feature stream as the second-best performing model, to be evaluated further. As a brief summary of the training procedure of these two models, we employed modified VGG (pretrained on ImageNet) and modified LeNet (pretrained on MNIST), the last fully-connected layer of the original structure is replaced by a fully-connected layer followed by a Sigmoid layer, as the feature extractor units. We fine-tuned VGG on both color image and color mask training sets and fine-tuned LeNet on binary mask training set. Networks are trained for 10,000 iterations having 0.0001 as the base learning-rate which applies to all the layers, and 0.001 and 0.002 as the weights learning-rate and bias learning-rate of the added fully-connected layers. Using learned models, we extracted training and testing features from input data.

In the next step, we trained the regressor (fully-connected layers followed by a Sigmoid layer) using extracted features. Different training strategies are applied and each strategy is evaluated based on its performance on the testing set. Training settings are described in Table 4.3 and results are shown in Table 4.4. Additionally, pairs of test images in conjunction with their estimated damage in a range between 0 and 1 of our best-performing model (color image feature stream + color mask feature stream trained by medium lr in Table 4.4), are illustrated in Table 4.5.

Table 4.4: Results of further experiments conducted on our two best-performing models, pretrained VGG on ImageNet and fine-tuned on color image and color mask training sets as the best-performing model and its combination with pretrained LeNet on MNIST and fine-tuned on binary mask training set as the second best-performing model. We trained them using different learning rates described in Table 4.3.

Model \ Setting	high lr	medium high lr	medium lr	low lr
	color image + color mask feature streams	0.172	0.169	<b>0.166</b>
color image + color mask + binary mask feature streams	0.177	0.176	0.175	<b>0.173</b>

Table 4.5: Examples of our best-performing model, color image + color mask feature streams trained by medium lr in Table 4.4. Left two columns are inputs to the algorithm, the third column is ground-truth, and the last column is the estimated damage.

Color image	Color mask	Label	Output
		0.0	0.03
		0.25	0.30
		0.5	0.54
		0.75	0.74
		1	0.97

## Chapter 5

# Conclusion and Future Work

In this work, we presented a novel hierarchical model performing building damage assessment in a continuous fashion using post-disaster images of damaged buildings as the input. The model contains three different pipelines and each pipeline is made up of one or two different ConvNets. These pipelines are designed to extract features from the raw input data, objects that we consider as relevant to damage assessment, and the shape of intact and damaged building parts. Additionally, we collected a dataset of post-disaster images to train and evaluate our model. We built several models using combinations of three pipelines and evaluated their performance. The results show that the model taking advantage of features of raw input data and objects of interest performs better than other models.

### 5.1 Limitations and Future Directions

In our proposed methods, we leveraged the power of deep structured architectures to deal with the automatic damages assessment problem. To our knowledge, there is no publicly available dataset of ground-level images of areas undergoing calamity containing consistent annotations. Therefore, we collected a small dataset to train and evaluate our models. However, training deep networks using small datasets is prone to the over-fitting issue, in which the method performs well on the training set and poorly on the testing set. Having larger datasets could result in developing more complex and robust models.

Moreover, expert people in the tectonic community could provide more annotations per image, such as different level of damage for different parts of the building structure rather than having a single label for the whole structure. Additionally, collected data by insurance companies, such as an estimation of damage in a specific currency, could be added to the dataset.

In this work, we assess damages to buildings by a continuous value between 0 and 1. However, having aforementioned annotations in the dataset and using the same model

structure could lead to estimating the total loss in a specific currency. This application is useful for insurance companies and expedites post-disaster attempts in responding to catastrophes.

Yet another interesting direction is to apply active learning in conjunction with region-based convolutional neural networks (R-CNN) in order to detect damaged areas of different structures, such as road, steel, and bridge. Utilizing active learning in automatic damage assessment is a significant field of research since it reduces human annotation costs substantially while having comparable performance to supervised methods.

# Bibliography

- [1] *Building Damage Assessment Report*. Retrieved April, 2016, from [http://www.eqclearinghouse.org/co/20100112-haiti/wp-content/uploads/2010/02/PDNA\\_damage\\_assessment\\_report\\_v03-1.pdf](http://www.eqclearinghouse.org/co/20100112-haiti/wp-content/uploads/2010/02/PDNA_damage_assessment_report_v03-1.pdf).
- [2] *Introducing the Open Images Dataset*. Retrieved September, 2016, from <https://research.googleblog.com/2016/09/introducing-open-images-dataset.html>.
- [3] *Virtual Disaster Viewer*. Retrieved April, 2016, from [http://vdv.mceer.buffalo.edu/vdv/select\\_event.php](http://vdv.mceer.buffalo.edu/vdv/select_event.php).
- [4] Susan A Bartels and Michael J VanRooyen. Medical complications associated with earthquakes. *The Lancet*, 379(9817):748–757, 2012.
- [5] Jim Thomas, Ahsan Kareem, Kevin Bowyer. Towards a robust, automated hurricane damage assessment from high-resolution images. In *International Conference on Web Engineering (ICWE)*, 2011.
- [6] Yuri Y. Boykov and Marie-Pierre Jolly. Interactive graph cuts for optimal boundary & region segmentation of objects in n-d images. In *International Conference on Computer Vision (ICCV)*, 2001.
- [7] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.
- [8] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *arXiv preprint arXiv:1606.00915*, 2016.
- [9] Koby Crammer and Yoram Singer. Pranking with ranking. In *Advances in Neural Information Processing Systems 14*, pages 641–647. MIT Press, 2001.
- [10] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition (CVPR)*, 2005.
- [11] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Computer Vision and Pattern Recognition (CVPR)*, 2014.
- [12] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollar, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision (ECCV)*, 2005.

- [13] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.
- [14] Li Fei-Fei, Rob Fergus, and Pietro Perona. One-shot learning of object categories. *IEEE transactions on pattern analysis and machine intelligence*, 28(4):594–611, 2006.
- [15] Pierre Sermanet, David Eigen, Xiang Zhang, Michael Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2014.
- [16] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Computer Vision and Pattern Recognition (CVPR)*, 2012.
- [17] B. Hariharan, P. Arbelaez, R. Girshick, and J. Malik. Simultaneous detection and segmentation. In *European Conference on Computer Vision (ECCV)*, 2014.
- [18] Gregory Griffin, Alex Holub, and Pietro Perona. Caltech-256 object category dataset. 2007.
- [19] Lionel Gueguen and Raffay Hamid. Large-scale damage detection using satellite imagery. In *Computer Vision and Pattern Recognition (CVPR)*, 2015.
- [20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Computer Vision and Pattern Recognition (CVPR)*, 2016.
- [21] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning (ICML)*, 2015.
- [22] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 675–678. ACM, 2014.
- [23] J. Shotton, M. Johnson, and R. Cipolla. Semantic texton forests for image categorization and segmentation. In *Computer Vision and Pattern Recognition (CVPR)*, 2008.
- [24] Adam Kilgarriff and Christiane Fellbaum. Wordnet: An electronic lexical database, 2000.
- [25] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. “Grabcut”: Interactive foreground extraction using iterated graph cuts. In *Special Interest Group on Computer GRAPHics and Interactive Techniques (SIGGRAPH)*, 2004.
- [26] P. Krahenbuhl and V. Koltun. Efficient inference in fully connected crfs with gaussian edge potentials. In *Neural Information Processing Systems (NIPS)*, 2011.
- [27] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.



- [28] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition (CVPR)*, 2009.
- [29] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.
- [30] P. Arbelaez, J. Pont-Tuset, J. T. Barron, F. Marques, and J. Malik. Multiscale combinatorial grouping. In *Computer Vision and Pattern Recognition (CVPR)*, 2014.
- [31] Pascal Monasse and Frederic Guichard. Fast computation of a contrast-invariant image representation. *IEEE Transactions on Image Processing*, 9(5):860–872, 2000.
- [32] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *Computer Vision and Pattern Recognition (CVPR)*, 2010.
- [33] J. Shotton, J. Winn, C. Rother, and A. Criminis. Textonboost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. In *European Conference on Computer Vision (ECCV)*, 2006.
- [34] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.
- [35] Bryan C Russell, Antonio Torralba, Kevin P Murphy, and William T Freeman. Labelme: a database and web-based tool for image annotation. *International journal of computer vision*, 77(1-3):157–173, 2008.
- [36] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Computer Vision and Pattern Recognition (CVPR)*, 2015.
- [37] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations (ICLR)*, 2015.
- [38] R. Sitton. *Spelling Sourcebook*. Egger Publishing, 1996.
- [39] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Neural Information Processing Systems (NIPS)*, 2012.
- [40] Antonio Torralba, Rob Fergus, and William T Freeman. 80 million tiny images: A large data set for nonparametric object and scene recognition. *IEEE transactions on pattern analysis and machine intelligence*, 30(11):1958–1970, 2008.
- [41] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva. Learning deep features for scene recognition using places database. In *Neural Information Processing Systems (NIPS)*, 2014.

- [42] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Computer Vision and Pattern Recognition (CVPR)*, 2015.
- [43] B. Fulkerson, A. Vedaldi, and S. Soatto. Class segmentation and object localization with superpixel neighborhoods. In *International Conference on Computer Vision (ICCV)*, 2009.
- [44] Luis Von Ahn and Laura Dabbish. Labeling images with a computer game. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 319–326. ACM, 2004.
- [45] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. In *International Conference on Learning Representations (ICLR)*, 2016.
- [46] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European Conference on Computer Vision*, pages 818–833. Springer, 2014.
- [47] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Semantic understanding of scenes through the ade20k dataset. *arXiv preprint arXiv:1608.05442*, 2016.