Mapping Disaster Risks from Aerial Imagery

Driven Data Competition

(revised December 2019)

Ákos Jakub, Gábor Pelesz and Zeynep Tasci

Abstract—Throughout the semester our team worked on an online deep learning competition started by DrivenData. The main goal was to use provided aerial imagery to classify the roof material of identified buildings in St. Lucia, Guatemala, and Colombia, providing efficient disaster risk mapping on the given islands. Though the task may seem easy to solve at first glance through this task we learned that the proper image preprocessing and data cleaning is just as important (in some case even more important) than a proper hyperparameter optimization

Index Terms—Deep Learning, CNN, Deep CNN, Swish Activation, Squared Image Augmentation, aerial imagery processing

- *Ákos Jakub - data engineering, augmentation, cloud training E-mail:jakubakos@gmail.com*
- *Gábor Pelesz - image extraction and evaluation, model optimization, team coordination E-mail: gaborpelesz@gmail.com*
- *Zeynep Tasci- architecture research, documentation E-mail: tascizey@gmail.com*

1 Introduction

IN areas like the Caribbean that face considerable risk from natural hazards like earthquakes, hurricanes, and floods, the forces of nature can have a devastating effect. This is especially true where houses and buildings are not up to modern construction standards, often in poor and informal settlements. While buildings can be retrofit to better prepare them for disaster, the traditional method for identifying high-risk buildings involves going door to door by foot, taking weeks if not months and costing millions of dollars.

This is where artificial intelligence can help, using the aerial drone imagery of buildings across the Caribbean, provided by WeRobotics. Roof material is one of the main risk factors for earthquakes and hurricanes and a predictor of other risk factors, like building material, that are not readily seen from the air. Our goal was to create a Convolutional Neural Network which can classify the rooftops of houses across affected islands in order to help the case of disaster prevention.

2 Previous results of the field

2.1 About aerial imagery

Land use classification has been a long-standing research problem in remote sensing, and has historically been applied to coarse resolution multi-spectral imagery (for example, LANDSAT has 30m x 30m ground sampling distance (GSD), Quickbird 2.2m GSD). More recently, high-resolution aerial imagery has become available with a GSD of 5-10 cm, so that objects such as cars and buildings are distinguishable. [1]

2.2 Similar models

Hundreds of researches was found on the internet that could have given us insight of aerial image classification; such as Jamie Sherra's writing about Dense Semantic Labeling of High Resolution Aerial Imagery or Michael Xie's work about Poverty Mapping with Aerial Imagery.

However these particular articles helped us to realize the complexity and the depth of the field, they did not provide us further information on how the network should be trained.

2.3 Best similar model

The closest similar model that we found was written by Ramón Alcarria about Geospatial Element Detection. This paper tackles the problem of object recognition in high-resolution aerial imagery and addresses the application of Deep Learning techniques to solve a challenge related to detecting the existence of geospatial elements (road network) in the available cartographic support. [2]

This model was basically a binary classification network for the given satellite image, our hope was that by modifying this network and applying to our data, it could provide a somewhat acceptable solution with the appropriate hyperparameter optimization. Based on this model we were able to develop a more advanced Deep Convolutional Neural Network which produced one of the best results compared to other networks.

3 Structure

In the next section we talk about the steps we took to enhance our dataset with a new augmentation technique as the dataset was heavily biased towards the healthy metal category.

It was a heavy concern among the team since the beginning of the project whether we should use transfer learning to complete the task or is it more beneficial to create our own CNN.

As a first attempt we tried the ResNet 50 v2 pre-trained architecture to classify the rooftops. Sadly the model was too advanced, therefore we tried to create our own CNNs, such as: Binar_v01, Caribbean DCNN. All of the models are explained in section 4.2.

4 Implementation

4.1 Data

4.1.1 Accessing data

By enrolling to the Driven Data competition 5 GeoTIFF images were given to work with. Geographic information systems use GeoTIFF and other formats to organize and store gridded raster datasets such as satellite imagery and terrain models. [3]

Every image represented a separated island with 3.5-4.5 cm resolution created by drone imagery. The coordinates and the labels for the rooftops were given.

Drone image of borde rural(Fig. 1) Example rooftops (Fig. 2)

To achieve easy accessibility we created our own image extractor algorithm. In the first step we located a

rooftop by its coordinates (which were given in the GeoJSON files linked to the main images). After the localization the rooftop was extracted from the TIFF image and saved as an individual PNG formatted image. With iterating through all of the rooftops the dataset were created. Since none of the other image processing libraries could handle images this big, we used Rasterio to

4.1.2 Cleaning the data

After the extraction of the rooftop images, there were two major problems with the freshly generated dataset.

1. The labeling and the form of some images were confusing and misleading

concrete cement (Fig. 3)

healthy metal (Fig. 4)

For example: among the mainly grey colored cement rooftops were outstandingly rust colored, or metallish images as shown in Fig. 3; in healthy metal category were also misplaced images such as shown in Fig. 4.
Seemingly cement based rooftops could be found in the healthy metal folder and vica versa in almost all of the 5 categories. Some images were faulty to the point where their belongings cannot be interpreted even by human eyes; **manual data cleaning were needed over 14.000 images. Based on the experience with the training dataset sadly we can assume similar defects in the testing dataset.**

2. Big difference between the number of images in the classes

A more solvable problem was that the number of images in the healthy metal category were much bigger than the sum of the 3 smallest group. There are several problems hidden in spatial data acquired from different sources with heterogenous representations. In order to maintain a proper training session it is inevitable to have almost equal amount of training data for each category. Otherwise the data with the lesser amount of images will ruin the quality of the neural network. [6]

This problem in particular is not unknown to Data Science, and various algorithms were created in order to solve it: such as Synthetic Minority Over-sampling Technique (SMOT), or an easier solution, like Random Over-sampling. Since these techniques seemed promising, we decided to create our own data augmentation algorithm based on the mentioned solutions. [5]

4.1.2 Squared Image Augmentation (SIA)

To enhance the quality of our dataset we developed an augmentation technique, which was a mixture of undersampling on the majority classes, and squared oversampling on the minority classes. The main goal was to equalize the number of images in every category. The ideal number was calculated by this heuristic:

Let's name s_k the number of images in category k, and S the array of these numbers for every category.

$$S = (s_0,\ s_1,\ s_2,\ s_3,\ \ldots,\ s_k)$$

The ideal number of images called I. A fare approximation would be to take the average of the image numbers, but by this approach a dataset with high difference (f.e: S = [50, 10.000, 130]) would suffer an extreme loss in the biggest datasets, which would lead to loss of crucial information for the training. Therefore the category with the highest image count is selected and multiplied with a 0.15 weight and added to the average.

$$I = \frac{\sum_{i=0}^{k} s_i}{k} + (max(S) * 0.15)$$

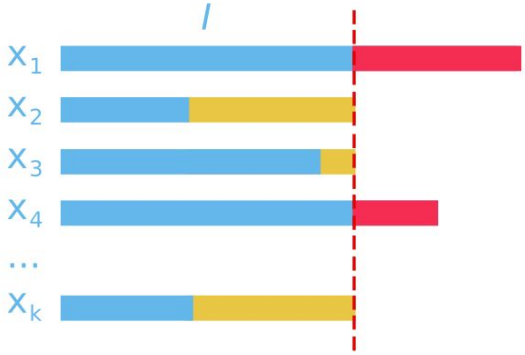
$$L_k = |s_k - I|$$

In the category where is more image than needed, keep the first I amount of image and delete the rest.
(Fig. 5: x_1 and x_4)

$$s_k > I \Rightarrow s_k \leftarrow s_k - L_k$$

In categories where is less image than needed, we need to generate the lacking amount.
(Fig. 5: x_2, x_3 and x_k)

$$s_k \leq I \Rightarrow s_k \leftarrow s_k + L_k$$



(Fig. 5)

However the way of generation is not specified, we used the *ImageDataGenerator* class (*Keras*) to create new images. (Obviously the most plausible solution for the image creation would be done by GANs. It may be a good idea

for further development.)

It is important to note that applying SIA technique is only useful on the training dataset! The generation process, on images in the validation dataset can cause lower validation accuracy.

Through the development we worked with different datasets to earn good accuracy for different models. To avoid misunderstandings four main datasets were created:

- **original:** uncleaned extracted images
- **cleaned**: The manually cleaned dataset with 80% training and 20% validation split
- **augm_v2**: SIA algorithm ran on the training data of the vanilla dataset
- **augm_v3:** The manually cleaned dataset with 50-50% for training and validation and SIA algorithm applied on the training set

4.2 Training and hyperparameter optimization

Many different models were created to complete the task. The best networks and their performance are detailed below. All networks were trained on Google Cloud Platform with an NVIDIA K80.

Despite having a strong GPU it is unavoidable to state that the Google Cloud AI Platform is still under development. Kernel freeze and model saving problems were woefully common under the development which was one of the cases we could not try every implementation ideas through the semester. The possible solutions that we were not able to create are detailed in the conclusion section.

4.2.1 ResNet-50-Binary

The first attempt was a transfer learning solution with the ResNet-50 architecture. [7]

Naturally came the idea that such a large and advanced convolutional network, specialized for image classification and object recognition would easily solve the problem, but as we later found out this was not the case.

To test the limits of the network at first we gave it a simple task. Binary classification on the biggest two categories (healthy_metal, irregular_metal) over 10.000 samples.

```
Layer (type)                 Output Shape              Param #
=================================================================
resnet50 (Model)             (None, 2048)              23587712
_________________________________________________________________
dense_9 (Dense)              (None, 256)               524544
_________________________________________________________________
dense_10 (Dense)             (None, 128)               32896
_________________________________________________________________
dense_11 (Dense)             (None, 1)                 129
=================================================================
Total params: 24,145,281
Trainable params: 557,569
Non-trainable params: 23,587,712
_________________________________________________________________
Found 10097 images belonging to 2 classes.
Found 2526 images belonging to 2 classes.
```

ResNet-50 with binary classifier on top (Fig. 6)

The validation accuracy stopped at the first epoch and did not change throughout the whole training, although the training was time and resource consuming.
The feature extractors of the ResNet with weights pre-trained on the ImageNet dataset were too different to efficiently use in our case. The extracted images had high variance in size parameters. The average height was about 233 px, with 80.7 standard deviation, and the average width was 236.2 px with 80.6 standard deviation. Next intuition was to create a much simpler lightweight network.

4.2.2 Binary_v01

With the same binary classification dataset a second handmade model was created. 3 convolutional layer with 2x2 pixel Max Pooling window, and ReLU activation on every layer.

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_4 (Conv2D)            (None, 298, 298, 32)      896
_________________________________________________________________
activation_6 (Activation)    (None, 298, 298, 32)      0
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 149, 149, 32)      0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 147, 147, 32)      9248
_________________________________________________________________
activation_7 (Activation)    (None, 147, 147, 32)      0
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 73, 73, 32)        0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 71, 71, 64)        18496
_________________________________________________________________
activation_8 (Activation)    (None, 71, 71, 64)        0
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 35, 35, 64)        0
_________________________________________________________________
flatten_2 (Flatten)          (None, 78400)             0
_________________________________________________________________
dense_3 (Dense)              (None, 64)                5017664
_________________________________________________________________
activation_9 (Activation)    (None, 64)                0
_________________________________________________________________
dropout_2 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_4 (Dense)              (None, 2)                 130
_________________________________________________________________
activation_10 (Activation)   (None, 2)                 0
=================================================================
Total params: 5,046,434
Trainable params: 5,046,434
Non-trainable params: 0
```

Binary_v01 architecture (Fig. 7)

There were more trainable parameters than in the previous model of ours and network performed quite well. The validation loss had huge volatility, but the validation accuracy stagnated between 75 and 80 percent.

At this point we stated that with the right amount of layers and proper hyperparameter optimization, the task is solvable.

4.2.3 Caribbean DCNN

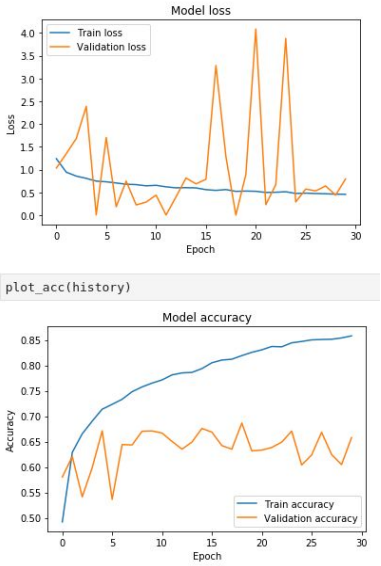
Deep convolutional neural networks have recently been substantially improving upon the state of the art in image classification and other recognition tasks.[6] In our case, the images are passed through five convolutional layers, where we use filters with a 3×3 receptive field. Spatial pooling is carried out by five max-pooling layers, which follow every convolutional layer. Max-pooling is performed over a 2×2 pixel window. Next, a flatten operation on the tensor reshapes it to have a 1-d share that is equal to the number of elements contained in the tensor as in [2].

Finally, the stack of convolutional layers is followed by three Fully-Connected (FC) layers: the first has 256 channels while the second has only 64. The last one performs the 5-way classification and contains 1 channel.

Currently, the most successful and widely-used activation function is the Rectified Linear Unit (ReLU). Although various alternatives to ReLU have been proposed, none have managed to replace it due to inconsistent gains.

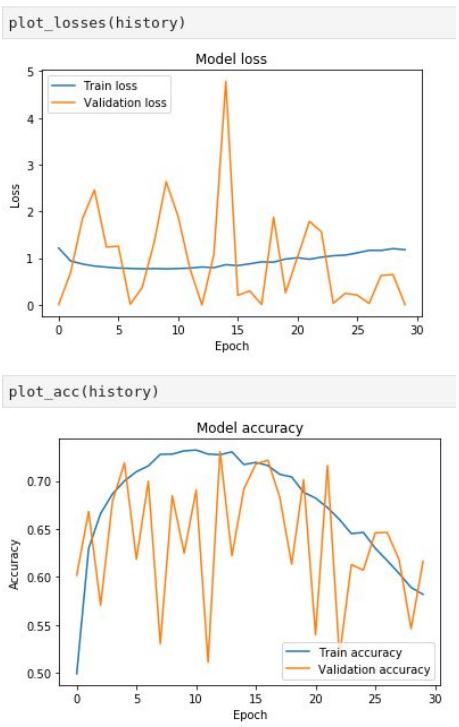
Unlike in the Binary_v01 and ResNet-50-Binary models where we used ReLU, the Swish activation was introduced after every layer (except the last one; softmax). Swish is simply f(x) = x·sigmoid(x). Swish tends to work better than ReLU on deeper models across a number of challenging data sets. For example, simply replacing ReLUs with Swish units improves top-1 classification accuracy on ImageNet by 0.9% for Mobile NASNetA and 0.6% for Inception-ResNet-v2. [8]

The new model didn't perform as good as expected with a fare 0.63 validation accuracy and an everlasting volatility of the validation loss:



Caribbean DCNN performance (Fig. 8)

To lessen the volatility of the validation loss, and improve the quality of the whole network, a 0.4 Dropout layer were added between the two middle convolutional layer. The early stopping were also removed just in order to check whether the network is properly learning (of course the best model was restored at the end of the training).



Caribbean DCNN (with dropout) performance (Fig. 9)

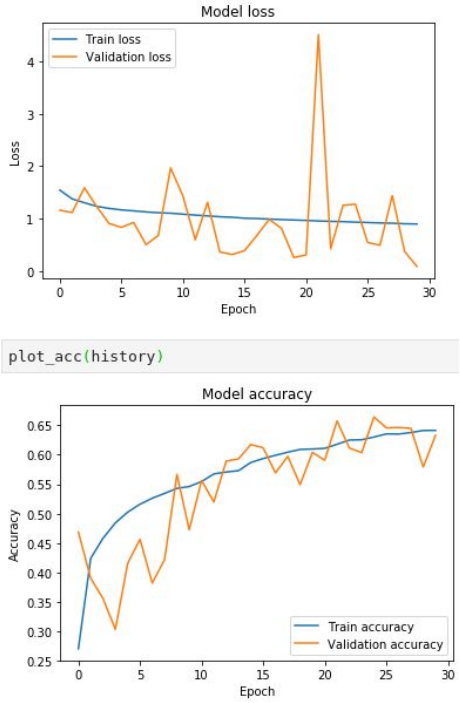
Despite making more hyperparameter changes and adding dropout layers the results didn't seem to change. A huge fluctuation was observable between the losses and also between the accuracies, which could be caused

by two major factors:

1. **Inaccurate hyperparameters**: Too big learning rate, which leads to extreme jumps in the optimization algorithm

2. **Mislabeled or low quality data**: Portion of examples is classified randomly, which produces fluctuations, as the number of correct random guesses always fluctuate

The first assumption can be easily tested by increasing the number the epochs and choosing a small learning rate to the optimization algorithm: $\alpha = 0.00001$, although as we stated before hyperparameter changes were not increasing the performance of the model.



Caribbean DCNN (low LR) performance (Fig. 10)

Although the new model's validation accuracy became much more stable, the loss fluctuation stood the same.

It was now clear that further improvements to the data itself was needed.

4.3 Test

For testing we used the competitions site where they provided a submission interface. To submit our results, the rooftops from the testset were extracted, evaluated and the probabilities were collected to a CSV file (in the submission format).

The metric used for this competition was logarithmic loss and the goal was to minimize it.

Network	val_loss	val_acc	submission
ResNet-50-Multy	1.18218	0.61660	1.6506
Binary_v01	0.52201	0.81703	---
Caribbean DCNN	0.00051	0.73260	32.140
Caribbean DCNN v2	0.21201	0.65201	19.176

Out of curiosity we examined the test images and sadly constated that these images faced the same problems as the training set did. Since we could not test our results locally we were not able to manually clear the testset. □

5 Conclusion

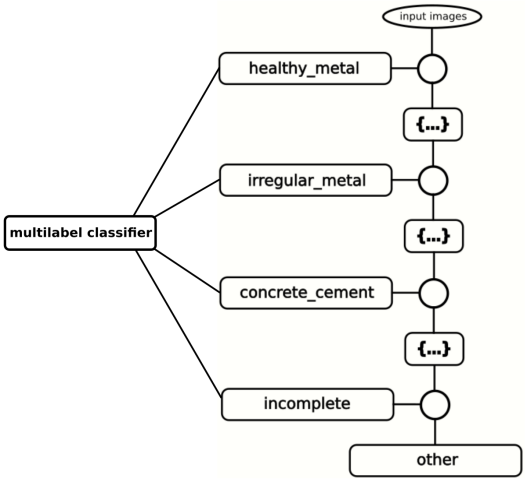
Almost every model result strengthened our belief that the used images were either labeled automatically or falsely which led to major validation inconsistency.

There might be a heuristical structure for solving this particular problem, however we could neither find nor create a network with truly satisfying results with the given dataset.

5.1 Further possible solutions

I. The main idea of SIA algorithm might be good, but by using ImageDataGenerator class we get only modified copies of the original ones, not new ones with similar features. By training **Generative Adversarial Networks** for each category in order to generate the needed amount of samples might eliminate the problem

II. **Creating binary classification models ordered in 5 layers** from the biggest image set on top to the smallest on the bottom.



*(**circle** - Binary CNN; **rectangle** - class; **{...}** - rest of the classes)*

With this approach we could reduce the probability of false classification since all of the CNNs could be trained separately, with specified hyperparameters. Assuming that there are learnable structure in the first 4 class, every previously unseen rooftop with new material would end up in the other section.

If more than one binary model is activated, then a multilabel classifier decides from the activated ones which class will dominate.

III. There is also a chance that we used wrong hyperparameters, to get the wanted results, but by simply variating the values we could not achieve success. An AutoML solution might come in handy. **AutoKeras** is an open source software library for automated machine learning (AutoML). AutoKeras provides functions to automatically search for architecture and hyperparameters of deep learning models [9].

IV. In natural images, information is conveyed at different frequencies where higher frequencies are usually encoded with fine details and lower frequencies are usually encoded with global structures. Similarly, the output feature maps of a convolution layer can also be seen as a mixture of information at different frequencies. To enhance our network we considered implementing **Octave Convolution Layers** instead of the classic Convolutions [10].

References

[1] Landsat Science Lab of NASA (https://landsat.gsfc.nasa.gov/)

[2] A Deep Convolutional Neural Network to Detect the Existence of Geospatial Elements in High-Resolution Aerial Imagery (https://www.mdpi.com/2504-3900/19/1/17/pdf)

[3] Spatial Data Cleaning (https://s3.amazonaws.com/academia.edu.documents/31130745/79e4150c6f1905948f.pdf?response-content-disposition=inline%3B%20filename%3DAn_analysis_of_four_missing_data_treatme.pdf&X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIWOWYYGZ2Y53UL3A%2F20191213%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20191213T211408Z&X-Amz-Expires=3600&X-Amz-SignedHeaders=host&X-Amz-Signature=26c2a07a5ef0299206736ce37b0a55e31f1b562cb75a77db39ea9c0fed33a59f#page=96)

[4] Rasterio: access to geospatial raster data (https://rasterio.readthedocs.io/en/stable/)

[5] Synthetic Minority Over-sampling Technique (SMOT) (https://arxiv.org/pdf/1106.1813.pdf)

[6] Some Improvements on Deep Convolutional Neural Network Based Image Classification (https://arxiv.org/pdf/1312.5402.pdf)

[7] ImageNet/ResNet-50 Training in 224 Seconds (https://pdfs.semanticscholar.org/70d1/bed62b275188727e26bd6095441005550068.pdf)

[8] Swish: A self gated activation function (https://pdfs.semanticscholar.org/4f57/f486adea0bf95c252620a4e8af39232ef8bc.pdf)

[9] AutoKeras Official Documentation (https://autokeras.com/)

[10] Octave Convolution (https://arxiv.org/abs/1904.05049)