Data C200 Graduate Project - Final Submission First Draft, Project Writeup

Sander August Heggland Schrader, Aksel Nordli Katralen

April 29, 2024

1 Abstract

This paper discusses how machine learning models can be used to help emergency services with quick information for decision-making. The paper focuses on two main tasks, Task A and B. Task A aims to create a model for classifying which type of natural disaster has occurred (either SoCal Fire or Midwest Flooding). Task B on the other hand aims to create a model which can classify the damage level after Hurricane Matthew. Both tasks use different models. Logistic regression models are used as a baseline, while CNN models are used as the most optimal models. The regression models are trained on features extracted from the images in the dataset, like RGB and HSV features. The results of the model were generally good. The regression models scored around 90-91% accuracy and around 54%. The CNN models were slightly better, coming in at 99% accuracy for task A.

2 Introduction

Emergency situations are often chaotic and hard to get an overview over. Since time is a factor in these kind of situations, can machine learning models help in getting an overview and information used for decision-making? Research papers like [2], explain how models can be built for extracting information for decision-making in natural disasters. In the paper they represent how a CNN model can be used for this purpose. The CNN model they use, is trained on a pre-trained ResNet 50 architecture. Other research papers have explored similar approaches like [1], which assesses damage on buildings after the tsunami in Japan. This paper also uses a CNN model to assess the damage level.

This project will explore two main areas, Task A and Task B. Task A is to help an agency to deploy the correct subdivision to the corresponding disaster. For example the firefighter to the SoCal Fire disaster. The task will explore how satellite images can be used for disaster-type classification between SoCal Fire and Midwest Flooding.

Task B will explore how to assess the level of damage after a disaster, more specifically after Hurricane Matthew. The damage levels used in the dataset, vary in the form of different damage labels. The labels use a scale from 0 to 3, where 0 is no damage and 3 is completely destroyed.

We are given a dataset that is a subset of the larger dataset xBD dataset [2]. The dataset contains satellite images of three different disasters; Hurricane Matthew, SoCal Fire, and Midwest Flooding. In addition to the images and the disaster type, there is also information about the damage level of the disaster for the specific area as well as information about the height, width, and size of the images. All the images are taken after a disaster has occurred.

The project will use different approaches for the models for both tasks. First, we will provide benchmark models using logistic regression. Then we will make CNN models and compare these against our regression models.

3 Description of data

Since the dataset contains three different disasters, where the disasters correspond to different tasks, our first job was separating the disaster into Task A and Task B. Task A contains all the images for disaster types SoCal Fire and Midwest Flooding, while Task B contains only the images for Hurricane Matthew. The distribution of the amount of data per disaster is shown in figure 1. SoCal Fire and Midwest Flooding are about the same amount, while Hurricane Matthew has a considerably higher amount of data.

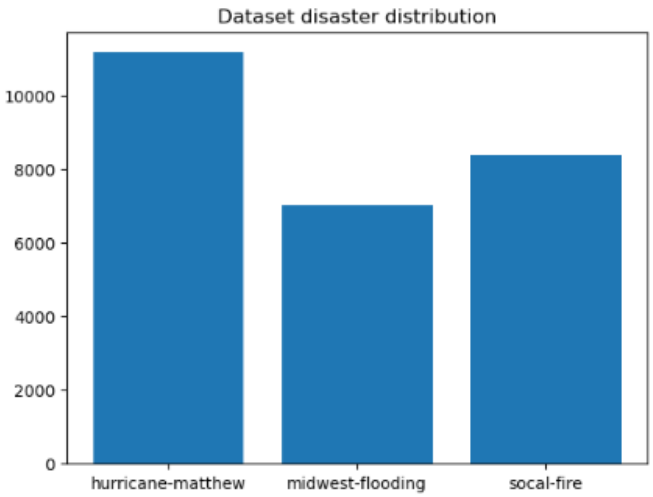

Figure 1: Distribution of disasters in the dataset.

The distribution of labels varies from disaster to disaster. SoCal Fire and Midwest Flooding have almost no data in labels 1,2,3, while Hurricane Matthew is relatively more evenly distributed. The distribution of Hurricane Matthew can be seen in figure 2. As shown in the figure, Hurricane Matthew has most of its data concentrated in the "Minor Damage" label.

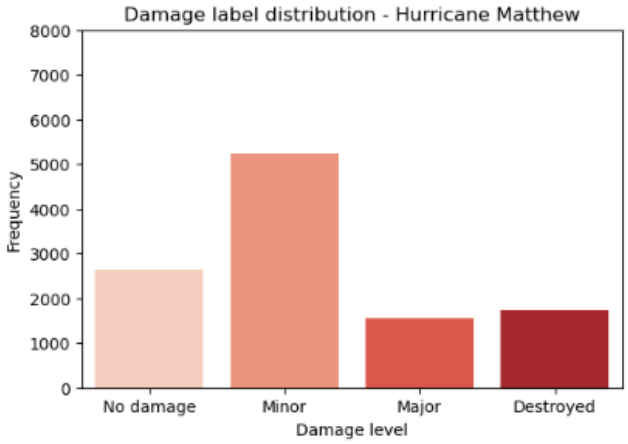

Figure 2: Distribution of labels for Hurricane Matthew.

3.1 Exploratory Data Analysis

Task A

For the EDA for task A, our main goal was to extract features from the images that clearly vary between the disasters SoCal Fire and Midwest Flooding. Our initial thought was to use color as a

feature, as there usually are very strong color differences between fires and floods. In addition, there are different landscapes for normal conditions in California and the Midwest.

The way we explored color, was through exploring each RGB channel separately for each disaster. We took the average of the color distribution of all images for both disasters in Task A. We then presented the data in a histogram with 8 separate bins which represent the distribution of the color channel. By doing this, we could check if there were generally speaking more of a color for a disaster. We could also check the amount in each bin for every histogram and compare them with each other. The histogram showing this color distribution for the disasters, is shown in figure 3.

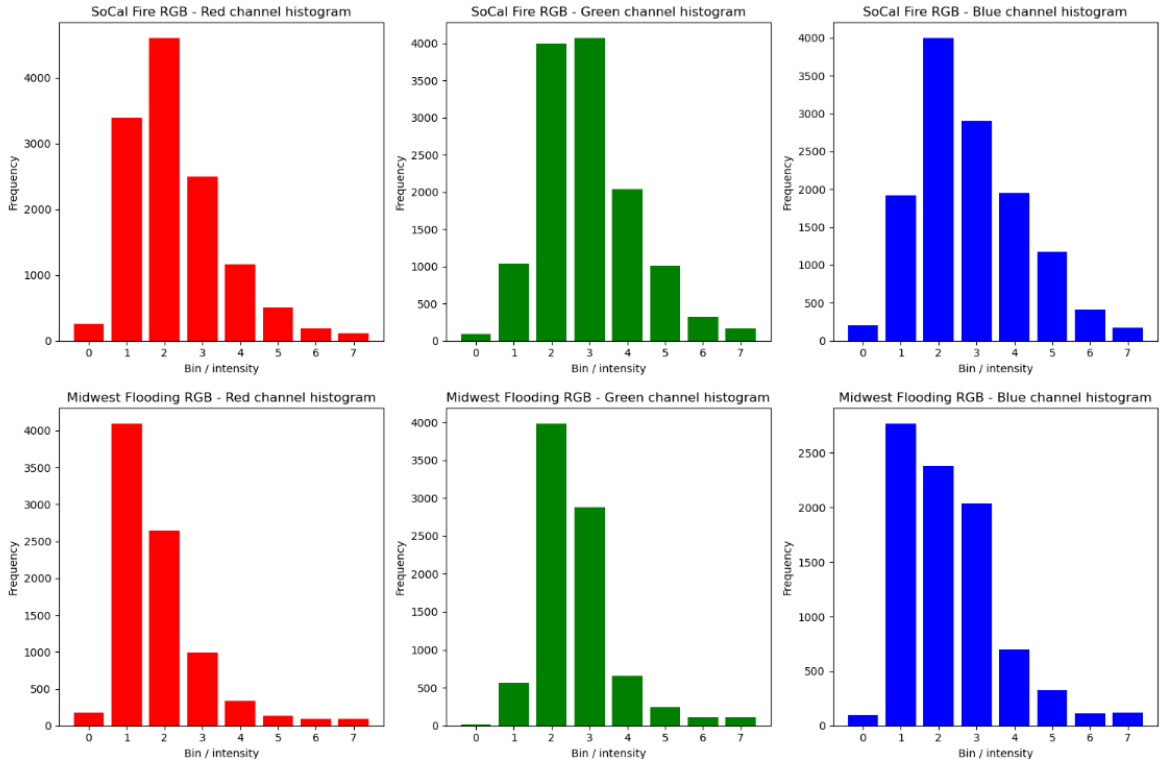

Figure 3: Distribution of color for RGB (red, green, and blue) channels per disaster.

As we can see from figure 3, there are clearly differences between the color distribution for the bins. As a result of this, we can then choose this as a feature for our model. This will give us 8 features per channel, which gives us a total of 24 features, just from these color distributions.

As we saw color distributions as a viable feature, we looked at how we could extract more information who resembled this. Therefore we tested out HSV (hue, saturation, value). We used the method as RGB, by ploting the distribution into a histogram with 8 bins. This is shown in figure **??**. Like RGB, the bin distribution varies by disasters, making it a suitable feature we can use. The addition of HSV as a feature brings in a new 24 feature in which the model can train on. This brings our total amount of features up to 48 features.

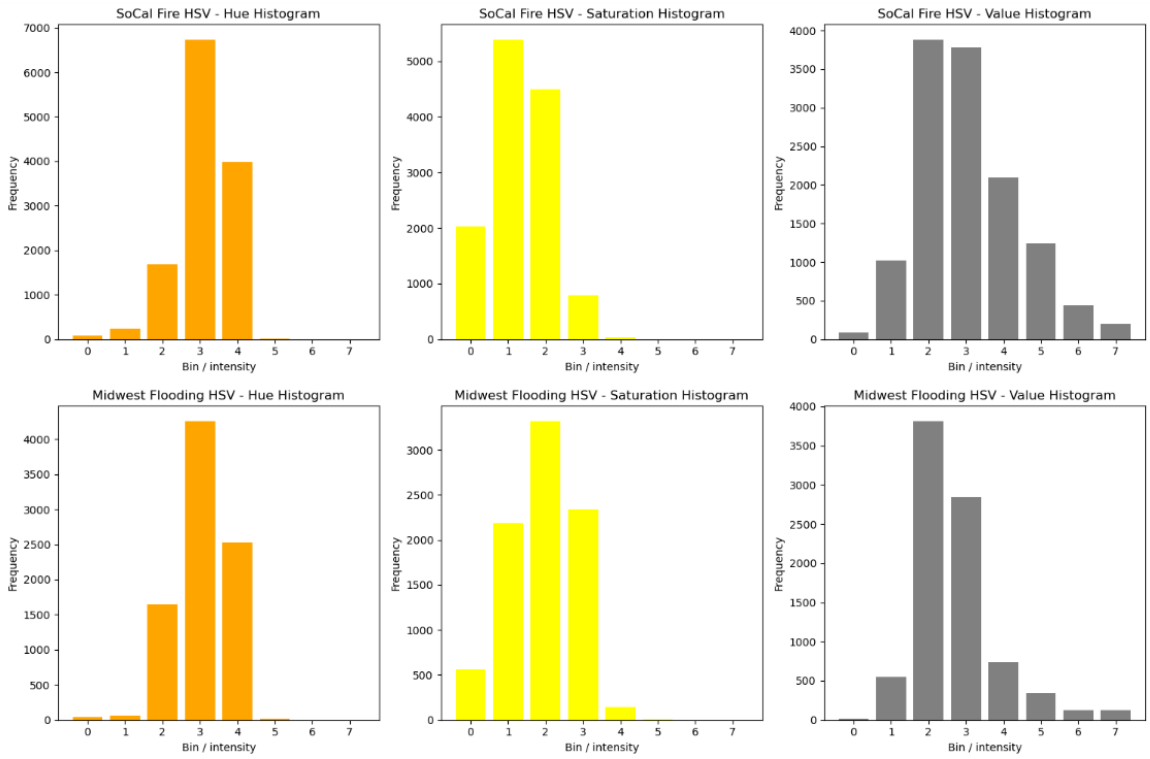

Figure 4: Distribution of color for HSV (hue, saturation, and value) channels per disaster.

Task B

In the EDA for Task B, we continued looking at using RGB and HSV as features. First we divided the data from Hurricane Matthew based on the labels. Then we used our previous histograms to plot the distribution for each channel in RGB and HSV for every label. The plots, as shown in figures 5 and 6.

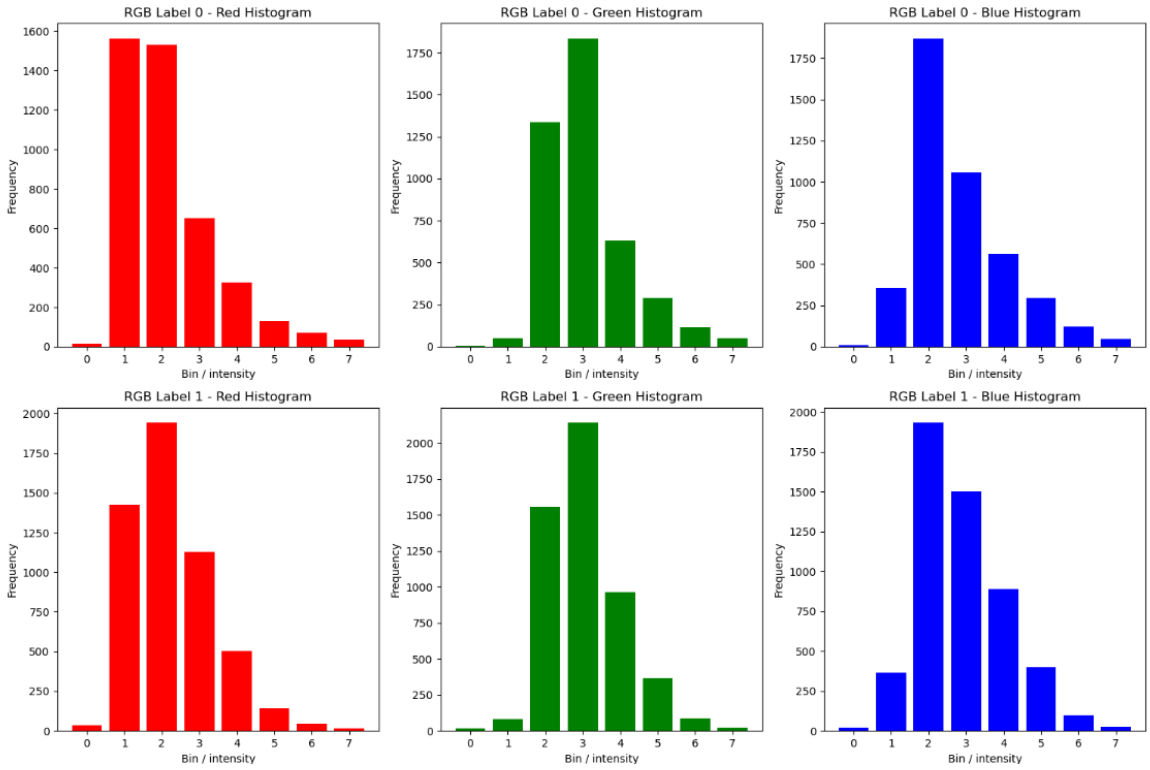

Figure 5: Distribution of RGB channels per label in Hurricane Matthew.

Another area we looked at was contrast. Our theory was that we would see some difference in contrast comparing labels because scattered debris would most likely increase the variation in intensity (contrast). Therefore, we have a plot showing the difference between labels. This can be seen in figure 7. The figure shows how the contrast also is divided into 8 bins.

The last area we explored in our EDA, was the use of LBP (Local Binary Pattern). The theory behind this was that LBP is usually very good at detecting edge patterns. These patterns would vary from label to label, especially between no damage and destroyed buildings. We used uniform LBP to reduce complexity and increase robustness against noise. Figure 8 shows the relation between the different labels. As we can see from the figure, labels 0, 1, and 2 have similar distributions, although not the same values, while label 3 has a different distribution. All in all, this gives us a total of 66 features.

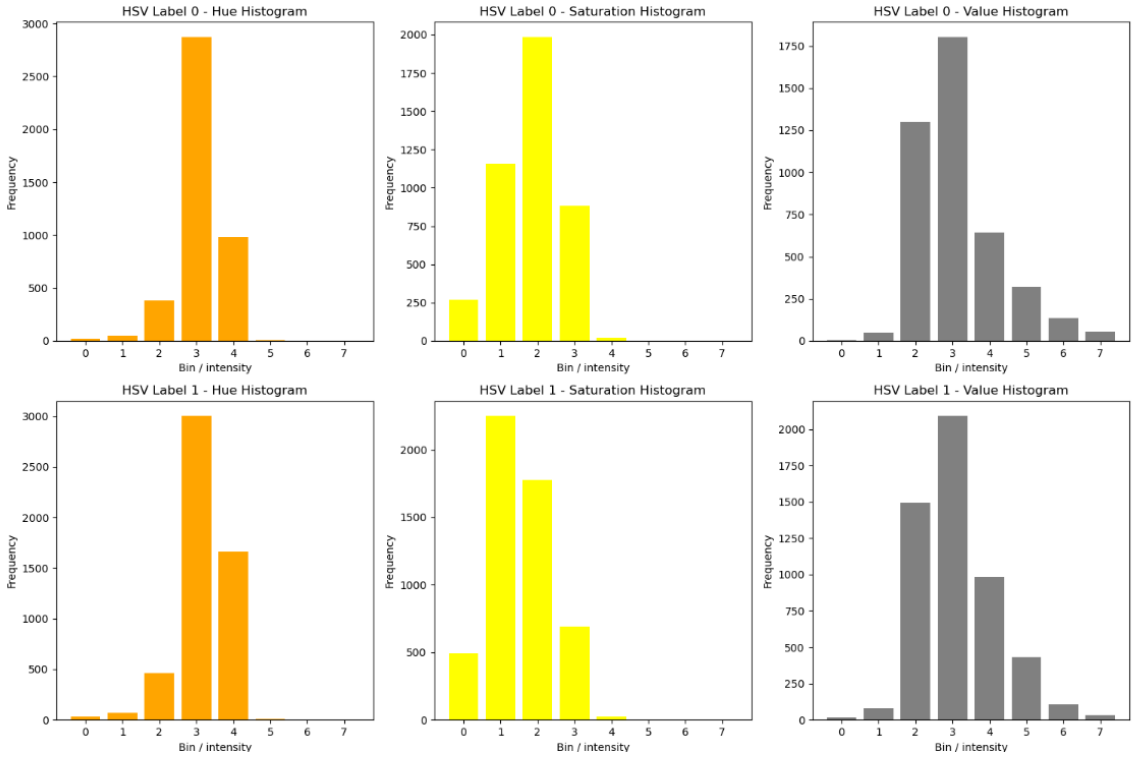

Figure 6: Distribution of HSV channels per label in Hurricane Matthew.

4 Methodology

During the project, we decided to explore two different modeling approaches. One of them was the required logistic regression model. The second modeling type explored was Convolution Neural Network (CNN) as these have over time proven to be extremely efficient and accurate on image classification tasks.

4.1 Modeling

4.1.1 Feature Engineering

Feature engineering is an important step in regression models. Especially with logistic regression. Our first attempts included getting the RGB and HSV (hue saturation value) distributions. The distributions were then split into 8 bins each creating a total of 48 features that we could use in our regression model.

4.1.2 Logistic Regression

Our first model was a logistic regression model train of features generated during the feature engineering step. Before training the dataset was sampled and shuffled. This was done in order to reduce bias in the dataset. Then the features were extracted from the new dataframe we made. The target values, the disaster types, were encoded from SoCal Fire/Midwest Flooding to 0/1. The feature vectors were scaled using a StandardScaler, and the data was split into a training and validation set. Then we train the logistic regression model, and evaluate the model through calculating the mean accuracy with cross validation.

4.1.3 Convolution Neural Network

Convolution neural are deep learning algorithms that use convolution layers. The network uses kernels to extract features from the input channels. The method has proven to be especially accurate and efficient for image classification. This was one of the reasons we decided to explore and use Convolutional

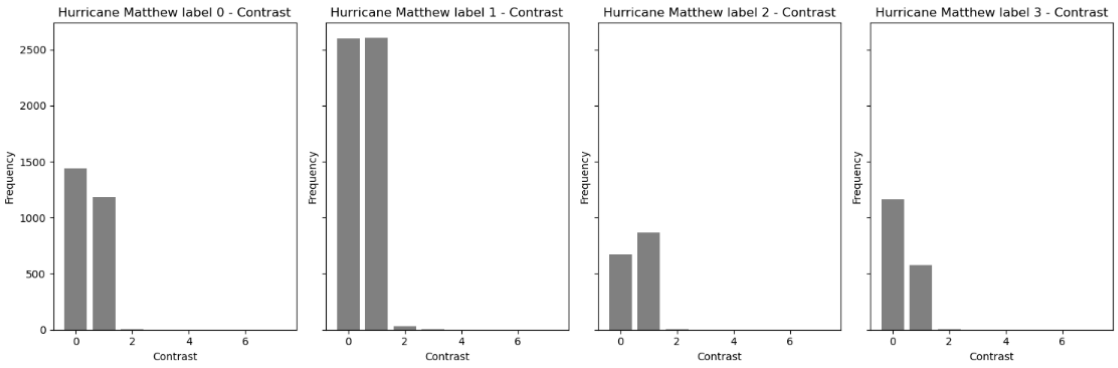

Figure 7: Contrast per label for Hurricane Matthew.

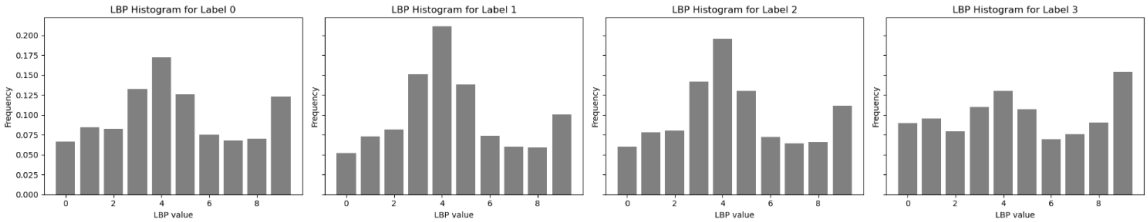

Figure 8: LBP per label for Hurricane Matthew.

Neural Networks for this classification task.

Convolution Neural Networks require a fixed data format. Due to this every image has to be resized to the same size. This was completed by padding the satellite photos to the same aspect ratios before resizing them. This ensures that the images keep as much detail and avoid smearing the images by changing the aspect ratios.

Custom Model

Our first trials utilizing Convolution neural networks were used with a custom model consisting of three convolution layers with max pooling. The convolution models used a kernel size of 3x3 and the number of filters was increased from 32-128 in increments of n*32. The model were then flattened before the data was put into two dense layers of size 256 and then 124 before the binary classification output was calculated. The model's architecture can be seen in Figure 9

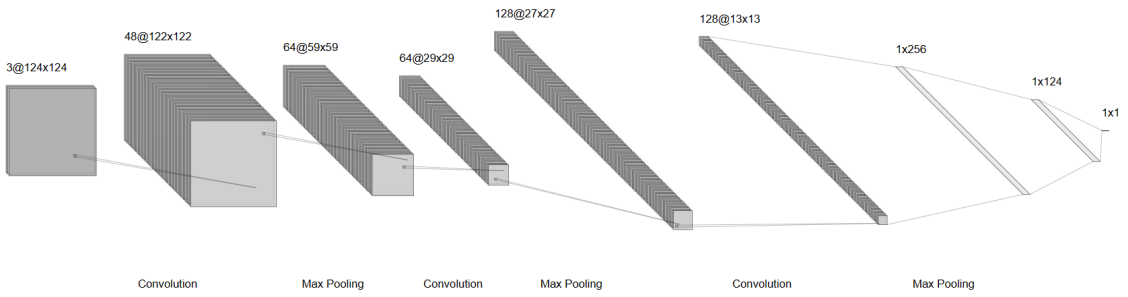

Figure 9: Image showing the architecture of the custom CNN model created

Transfer Learning

Transfer learning is a powerful method used in machine learning. By using a pre-trained and then slightly updating the weights we are able to utilize the filters already created by the model for other machine learning tasks. This method is efficient and cost-effective. We decided on using the RESNET 50 which contains 48 convolution layers and outputs 1000 classes. The model is trained on 14 million images[3].

5 Summary of results

The group used multiple models to solve both Task A and Task B. The Logistic Regression models served as a baseline for our results and CNN models were used to get better results. This is because CNN's are generally larger models fit for image classification tasks.

5.1 Task A

5.1.1 Logistic Regression

Our logistic regression models were generally able to get good results for our classification task. For Task A our regression model was able to get an accuracy of 91% on our validation set and 90% on the provided test set. This was done with a total of 48 features created from our EDA analysis. To validate the model and protect against overfitting we decided to use a train test split by taking 20% of our data to validate our model during training. The results for Task A can be seen in table 1.

Fold Number	Accuracy
1	0.7857
2	0.9286
3	0.9196
4	0.9643
5	0.9554
Mean	**0.9107**

Table 1: Cross-validation accuracies for each fold and mean accuracy for the regression model.

5.1.2 Convolution Neural Networks

Custom Model

The first convolution neural network used was our custom model. Through hyperparameter testing, we found that training the model with around 50 epochs seemed to be a good fit. The model was quickly able to get a good accuracy but since the task required an extremely high accuracy we had to train it for more epochs as the model had to find the best parameter weights for our problem. Plotting the training accuracy and our validation accuracy we could see that the model was not overfitting but found that the model had trouble getting higher accuracy. There could be multiple reasons for this, such as having the learning rate to high or the model being to small such that it could not extract enough information from our dataset. The resulting model had a test accuracy of 98.6% which was not enough to reach the threshold of 99.27%.

Resnet 50

	damage_label	recall	accuracy	count
0	0	99.0947%	99.0947%	1215
1	1	100.0000%	100.0000%	11
2	2	100.0000%	100.0000%	6
3	3	99.4565%	99.4565%	184

Figure 10: SoCal recall and accuracy by damage level

	damage_label	recall	accuracy	count
0	0	99.7749%	99.7749%	1333
1	1	100.0000%	100.0000%	25
2	2	100.0000%	100.0000%	16
3	3	100.0000%	100.0000%	12

Figure 11: Midwest recall and accuracy by damage level

Figure 12: Image showing the custom Model's accuracy, and recall grouped by damage label

As the team had experience with transfer learning from other classes and saw that this method was able to get a high accuracy in other problems we decided to try it out on this problem. Early results

showed that the model was able to quickly adapt to our problem showing an initial high accuracy on our training set. The validation accuracy quickly followed although taking a little bit of time. The ResNet 50 model is a lot larger than our custom model and were able to get an accuracy of 99.5% on our training set. A little over our custom model. This proved to be valuable as the model was able to get an accuracy of approximately 99.3% on the kept-out testing data.

	damage_label	recall	accuracy	count
0	0	99.8342%	99.8342%	1206
1	1	100.0000%	100.0000%	15
2	2	100.0000%	100.0000%	3
3	3	100.0000%	100.0000%	168

Figure 13: SoCal recall and accuracy by damage level

	damage_label	recall	accuracy	count
0	0	99.1098%	99.1098%	1348
1	1	96.4286%	96.4286%	28
2	2	100.0000%	100.0000%	21
3	3	100.0000%	100.0000%	13

Figure 14: Midwest recall and accuracy by damage level

Figure 15: Image showing Fitted ResNet50's accuracy, and recall grouped by damage label

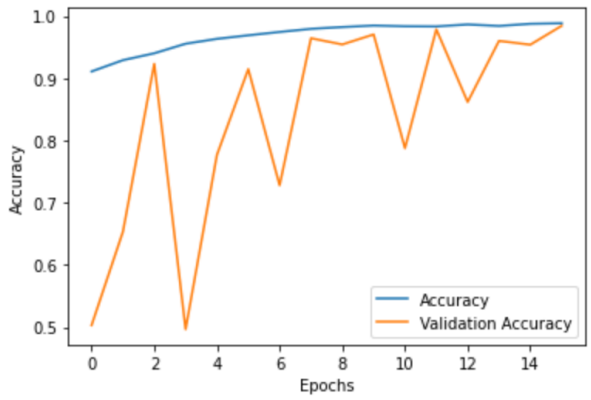

Figure 16: Image showing the Training vs Validation accuracy per epoch

The results in Figure 15 show the model does not seem biased toward any damage labels. However, the model seems better at predicting the social fire locations than the midwest flooding. Figure 16 shows part of the training of the ResNet 50 architecture where we see that the is more prone to overfitting.

5.2 Task B

5.3 Logistic Regression

Logistic regression was able to show good results on this classification task classifying the different damage labels found in Hurricane Matthew. The model had 66 features and had a mean F1 score of 54% on both the validation set and the test set. The F1 score of each fold is presented in table 2. The results for each fold vary from 53% to 56%, which may indicate that the model is sensitive to specific splits.

Fold Number	F1 Score
1	0.5499
2	0.5452
3	0.5344
4	0.5636
5	0.5514
Mean	**0.5489**

Table 2: F1 scores for each fold and the mean F1 score of the model.

6 Discussion

The results from Task A show that image classification to predict which disaster an image is taken from is a feasible task with high accuracy. The model shows low biases towards different damage labels and this is important since we might be more interested in predicting damaged buildings as these are the buildings that need attendance during a disaster. Although the model showed good results there is a small amount of data from the different damage labels in the social fire and midwest flooding and this might therefore be misleading and is something that we should be aware of.

The three different approaches also show that there are multiple ways to accurately classify images. The different models were all able to predict with high accuracy. It was surprising that the ResNet 50 model performed the best, but this could be to the architectural differences between the two CNN models as the custom CNN model is trained on smaller images vs the larger model. Therefore we might be able to see better results with larger models. But this shows that tuning a pre-trained model is an approach that can be useful for image classification tasks.

Task B also showed have damage level classification is achievable. However, the results for the regression model showed have the results still can be improved. This is clear due to the results not meeting the last threshold for the test set. To achieve better results a CNN model would be the best way to go. A CNN model would be able to catch minor differences and make better features than we currently have for damage-level classification.

7 Conclusion

In conclusion, this paper has shown that machine learning models can be used to classify both disasters and damage levels. Through use of models like logistic regression and CNN, our models have performed very well. The logistic regression model for Task A scored around 90-91% in accuracy and for Task B 54% in F1 score. The CNN model for task A scored an amazingly 99%, outperforming the regression model by 9%. These results underline the importance and effectiveness of machine learning in emergency response situations and understanding satellite images.

References

[1] Aito Fujita et al. "Damage Detection from Aerial Images via Convolutional Neural Networks". In: *15th IAPR International Conference on Machine Vision Applications (MVA)*. Nagoya University. Nagoya, Japan, May 2017. URL: https://www.mva-org.jp/Proceedings/2017USB/papers/01-02.pdf.

[2] Ritwik Gupta et al. "xBD: A Dataset for Assessing Building Damage from Satellite Imagery". In: (2019). URL: https://arxiv.org/pdf/1911.09296.

[3] Nitish Kundu. "Exploring ResNet50: An In-Depth Look at the Model Architecture and Code Implementation". In: (Jan. 2023). URL: https://medium.com/@nitishkundu1993/exploring-resnet50-an-in-depth-look-at-the-model-architecture-and-code-implementation-d8d8fa67e46f.