Term Project - Final Report

Damage Classification using Satellite Images
Bilkent University Fall 2019-2020
CS 551 - Pattern Recognition

Sinan Sonlu, 20801272 Irmak Türköz, 21400487

I. Introduction

After natural disasters, detecting the damage in an urgent manner is an important problem in automatic image classification. Most of the time, satellite images of the disaster area supply the most comprehensive data and are not disturbed by the disaster as much as the local surveillance systems. By comparing the satellite images, that are captured from the same area at different times, change (in this case the post-damage) could be automatically assessed, using pattern recognition methods.

In this term project, we implement and compare multiple approaches for classifying post damage level of buildings from satellite imagery, given pre and post-disaster satellite images and building locations. We experiment with different feature extraction methods and various Convolutional Neural Network (CNN) architectures. We used multiple libraries and machines for training different models. For CNNs, we mainly used Google Colab and Keras with TensorFlow backend. Although Google Colab offers free and fast GPU computing, daily time limits and difficulties in data transfer prevented us from taking full advantage of the service. For preprocessing of the data, feature extraction, and Support Vector Machine (SVM) models, we used Matlab on our computers. Due to limited memory on our systems, we used a smaller subset of the data in such models. We use the xView2 Challenge Dataset to train and test our models.

A. Problem Definition

Given pre and post satellite images as well as building locations, we try to classify the damage level of a building as one of four classes, ranging from no-damage to destroyed. Generally in each satellite image, there are multiple buildings with varying sizes. Instead of using the whole satellite image, we first extract samples based on building locations. More advanced networks with attention could potentially work with whole satellite images as well. Since most image classification algorithms operate on rectangular images we extract each building along with the surrounding rectangular area.

We experiment with different settings for constructing such building samples. We also tried to find the whole image's damage locations and classify the damage level in the whole area where the image is located.

We try classifying the building sample based on the post image only and also based on the absolute difference between post and pre images. We experiment with using raw RGB and grayscale images, as well as using different image feature extraction methods. Input to the CNN is a set of raw or refined square images that contain a building, categorized based on damage level. We experiment with different CNN architectures to achieve the best performance for the test set, which consists of 30% of the labeled data. The remaining 70% is used for training. However, in certain cases where we limit the sample size, we used 80% for training and 20% for testing, to have more training samples.

B. Dataset

The xView2 training dataset contains 2799 pairs of 1024 by 1024 RGB satellite images in PNG format. Image file names include the location and type of the disaster, as well as an indicator for "pre" or "post" disaster. There are also JSON files per image that include WKT polygon annotations for building locations and post-disaster building damage levels. Damage level for a building is categorized with one of the following labels: No Damage, Minor Damage, Major Damage, and Destroyed. There are also unclassified buildings which we ignore. We split this dataset for training and test purposes. They also supply a test dataset of 933 pairs of satellite images for building and damage level allocation, which we do not use for the project since they do not provide the true labels yet, as this is an ongoing competition. Also, there is one additional set of labelled images that is introduced later on, which we also do not use.

II. Implementation

In this part, we will explain our implementation of the project and report the results we get in each step. The steps we have followed are preprocessing, feature extraction and models to finalize the output.

A. Preprocessing

To read and preprocess our data for the models we have experimented with, we have followed two different techniques. In one technique we have implemented sample extraction for training neural network VGG16 which we train with buildings, in the second technique we have implemented a different method to execute training for the whole image.

1) Sample Extraction for VGG16: Each satellite image pair in the dataset contains an area with multiple buildings in it. Since we focus on damage assessment of individual buildings, we first extract separate samples from each satellite image per building. We use two different approaches for extracting building samples from the satellite image: based on the smallest bounding rectangle that includes the building polygon, and based on a fixed size square that has the same center point as the building polygon.

The sizes of the smallest bounding rectangles vary dramatically. In order to input these samples to the network, we resize each into a fixed size square, as a result, we obtain samples with varying detail levels. Since we do not perform rotation on the satellite images, based on the rotation of the original building polygon, samples would have different numbers of non-building pixels. In other words, the rotation of the building would determine how much of the surrounding area is included in each sample, as a result of using rectangular bounding boxes. The resized image resolution is decided based on the network.

Extracting samples using a fixed size square keeps the detail level of each building the same, however, this time ratio of building and non-building pixels in a sample varies dramatically. If the building size is small, the fixed square would contain many pixels from the surrounding area. For disasters such as floods, including many pixels from the surrounding area could be useful; on the other hand, if the damage is limited to the building only, then the surrounding pixels could confuse the network. Additionally, due to the change in weather and lighting conditions, most satellite image pairs contain base differences, and when we include more surrounding pixels, it also increases the noise. We determine the fixed resolution based on VGG16 input size, which is 224 by 224.

Sample extraction produces two different sample sets: Fixed Size (FS) and Varying Size (VS), each with 159,794 pre and post sample pairs. Since there is class imbalance, we use different class weights calculated using the compute_class_weight function of SKLearn. After various experiments, we realized that the influence of each disaster type is different, and trying to learn the damage assessment for different kinds of disasters using the same model gives poor results. As a result, we categorize the samples based on the disaster type as well and train separate models for each type of disaster. Additionally, with VS, we still need to resize images to a fixed size for the network to be able to train on them. This caused smaller images to have big patches of same pixels, which negatively influenced the accuracy. Therefore we have chosen to continue with FS samples only. Table I shows the count of building samples per damage type and damage level.

2) Sample Extraction for Classification of Images: The second technique we have implemented for sample extraction is to assign the whole image's area to a damage level. To do this, we require information about the areas' damage assessment. However, since the only information available in our dataset is damage assessment for the buildings; we needed a different approach. First, we check the post damage images' JSON file and find the buildings in every image. Afterwards, we calculate a mean damage assessment for the area of the image by taking these buildings damage assessment levels.

Disaster	No-D.	Minor-D.	Major-D.	Dest.	Total
Earthquake	32066	110	18	2	32196
Fire	17923	161	114	4800	22998
Flooding	24240	2944	9602	530	37316
Tsunami	25455	1	571	4966	30993
Volcano	639	9	9	25	682
Wind	17103	11755	3847	2904	35609
Total	117426	14980	14161	13227	159794

TABLE I
NUMBER OF BUILDING SAMPLES PER DISASTER AND DAMAGE LEVEL. DAMAGE LEVEL ABBREVIATIONS ARE FROM LEFT: NO-DAMAGE, MINOR-DAMAGE, MAJOR-DAMAGE, AND DESTROYED.

Since for this purpose, we always require supervised training, instead of using the test data and validating the information we get; we have decided to use only the already assessed images and split them into test and train data. This, caused our dataset to decrease in size. The final step after we labeled every image with respect to their average damage assessments, to equalize the sizes of the classes for every damage level. The dataset has unequal numbers of data for average damage level assessments, and the minimum data we had was for the 'destroyed' label. To prevent our system to generate bias information about the classes due to the unequal sizes, we have extracted equal sizes for each of the classes which is exactly 259 images which also caused our dataset size to be shrink. One possible solution was to divide the images that belong to the least size class and classify them with these image patches. However, since our data was already normalized and we had the advantage of the preprocessing; we have not implemented a partitioning to the least sized class data.

B. Formatting Sample Pairs as Input

We would like to classify the change in pre to post building samples, which is a non-trivial function. We follow three different approaches to form an input out of extracted samples: using the post sample only (Post), using the absolute difference between pre and post samples (Diff) and using the concatenation of pre and post samples (Concat). With the first approach, we assume the damage is internal to the post image and could be determined solely based on it. In this case, input to the network is post building images only. For the second approach, we calculate the absolute distance between the post and pre images for each building, assuming the change that damage causes would be based on the difference in pixel intensities. In this case, input to the network is the difference image per building, calculated by the absolute difference between corresponding pre and post building images. Finally, we input the concatenation of pre and post building images to the network in order to learn the difference operation in a data-driven manner. Download links for each sample subset is given as a Google Drive folder, links are written in a text file that comes with the project code.

C. Feature Extraction

Even though we had already extracted building available in our dataset, we wanted to experiment with our own feature extraction techniques. We experiment with training the network using the input samples directly as well as using features extracted from the input samples. We have not applied feature extraction as an input of the neural network system however we have experimented with other classification models. For feature extraction, we have experimented with Local Binary Patterns, and Maximally Stable External Regions (MSER). The inputs to these feature extraction methods are gathered by sample extraction for classification of images and they are prepossessed as pre disaster minus post disaster.

1) Local Binary Patterns: We have experimented with local binary patterns on the absolute differences between post and pre images. We have used average building damage assessment to decide the damage level in training labels and extracted LBP with eight number of neighbors. Each central pixel is compared with its eight neighbors; the neighbors having smaller value than that of the central pixel will have the bit 0, and the other neighbors having value equal to or greater than that of the central pixel will have the bit 1 [2]. We have used LBP to experiment if it can detect the most important texture changes on the images.

Overall, our experiments with these features proved that LBP features does not work well as a solution to our problem. This might be due to the fact that we have different locations and different types of disaster that makes the LBP features meaningless. Moreover, to test the information we get with the LBP features, we have drawn a confusion matrix (Fig 1).

The results are not very promising as we have low accuracy and we detect most of the images as no damage. Another cause of this could be he average building damage label does not perfectly define the damage in the area where the image is taken. We have used fitted discriminant analysis model based on the LBP input variables to drawn this confusion matrix.

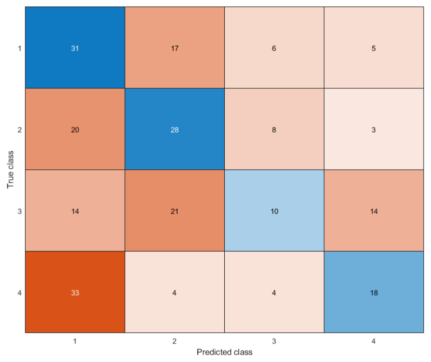

Fig. 1. Confusion matrix for Local Binary Patterns

2) Maximally Stable Extremal Regions (MSER): In computer vision, maximally stable extremal regions (MSER) are used as a method of blob detection that are aimed at detecting regions in a digital image which differ in properties, such as brightness or color, compared to surrounding regions[3]. These region analyzing aspect of MSER features have potential to detect buildings under different scenarios, not compromising with accuracy with shape based post-processing to improve building detection[4]. Thus, we have decided to experiment with MSER regions. MSER regions have potential to detect the paddings of the images as buildings because these regions appears as a big homogeneous object against the other objects on the absolute difference between two objects. To prevent this, we have used a normalization so that the only a meaningful size of the pixels are taken into the account. After we have detected MSER features for every image, we have visualized the edges of the MSER regions (Fig 2) as well as comparing four of the classes (Fig 3). These regions by themselves are not meaningful as we have tried to implement a direct classification algorithm based on the region areas.

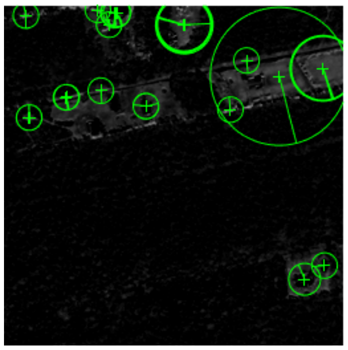

Fig. 2. Example of MSER feature regions edges.

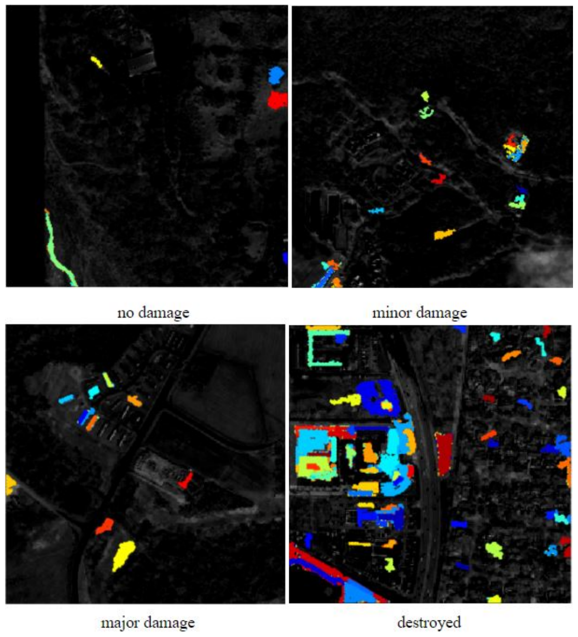

Fig. 3. MSER features for levels of damage

D. Models

1) Fine-tuning VGG16: We experiment with training different classification models. We fine-tune VGG16 and also train different CNNs from scratch. For fine-tuning we selected VGG16 since it has a relatively simple architecture that makes it easy to understand. Since for some disaster types, certain damage levels contain too many building samples that do not fit into the memory, we use a maximum of 4000 samples per damage level when training models. We use different class weights to compensate for the class imbalance. We use batch size 32, and train for 100 epochs and keep the best model according to the validation accuracy.

For fine-tuning VGG16, we freeze all the original layers and only train the three additional dense layers (128 units with Relu, 64 units with Relu, and 4 units with Softmax because of 4 classes we have). Table II shows the best validation accuracy per model, using different sample formats for different disaster types (Results with a '*' or '**' symbol indicate experiments where we experience memory issues and need to set a smaller sample threshold of 3000(*) and 2250(**). In the model Concat, since we need to concatenate post and pre images, more memory is required. Batching images from the disk were not feasible on Google Colab, therefore we preferred all samples to be on memory during each experiment, resulting in limited samples per class.)

Disaster	**Post Only**	**Difference**	**Concatenation**
Earthquake	0.972	0.975	0.975
Fire	0.918	0.888	0.929
Flooding	0.744	0.836	*0.823
Tsunami	0.917	0.93	0.916
Volcano	0.985	0.992	0.970
Wind	0.568	0.676	**0.648

TABLE II
TEST ACCURACY WITH DIFFERENT TRAINING SAMPLE APPROACHES USING VGG16 TO FINE-TUNE.

2) Training extended VGG16 from scratch: We also try training the same extended VGG16 model from scratch, with data augmentations to add some variation to the training set with image operations such as rotation and shift. Since VGG16 is pretrained on general images, unfreezing all the layers for training helps to focus on building images. Additionally, in the training set, some buildings only appear in certain angles, and by adding random rotations in each batch with data augmentation, we improve the generalization of the model. Since data augmentation requires more memory, we limit the sample size of each class to 1500. We also calculate class weights based on the training set, using compute_class_weight function of SKLearn Tools, to reduce the influence of imbalance, different from the earlier experiments. Table III shows the results.

	Post Only	**Difference**
Earthquake	0.969	0.898
Fire	0.902	0.909
Flooding	0.862	0.796
Tsunami	0.930	0.892
Volcano	0.963	0.956
Wind	0.720	0.685

TABLE III
TEST ACCURACY WITH DIFFERENT TRAINING SAMPLE APPROACHES TRAINING VGG16 FROM SCRATCH WITH DATA AUGMENTATION.

As training takes a long time, we did not have a chance to do an exhaustive search in the parameter space. We also trained each model once, instead of taking the average performance of multiple experiments, so the reported accuracy could have a 3-5% difference in repeated experiments.

Using the wind subset, we experiment with different starting learning rates of the Adadelta optimizer, the plots in Fig 4, 5 and 6 show the test and train accuracy changes in each epoch. Using a learning rate of 1 yields the plot in Fig 4, which fluctuates a lot. In general, such large jumps could cause the model to miss an optimal solution, although it has the potential to achieve higher accuracy faster.

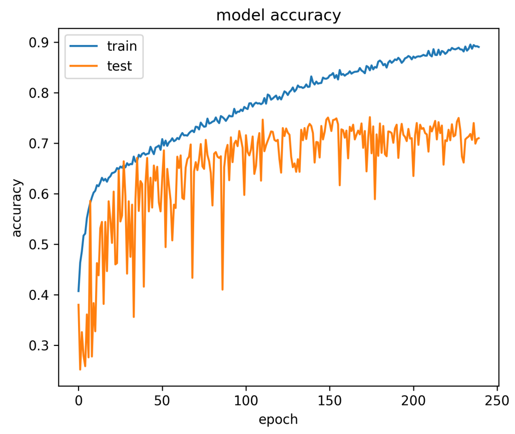

Fig. 4. Training and test accuracy plot for training the extended VGG16 model from scratch, using post only wind subset, with learning rate of 1 of Adadelta optimizer.

The learning rate of 0.0001 produces the plot in Fig 5, which was not able to achieve 0.50 training accuracy after 190 epochs, so we decided to abandon it since we require the models to train faster. For other experiments, we decided to use the learning rate of 0.01, which yields the plot in Fig 6, since it gives acceptable results within 200 epochs.

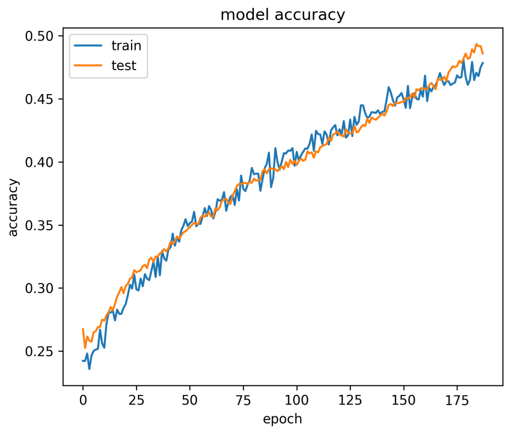

Fig. 5. Training and test accuracy plot for training the extended VGG16 model from scratch, using post only wind subset, with learning rate of 0.0001 of Adadelta optimizer.

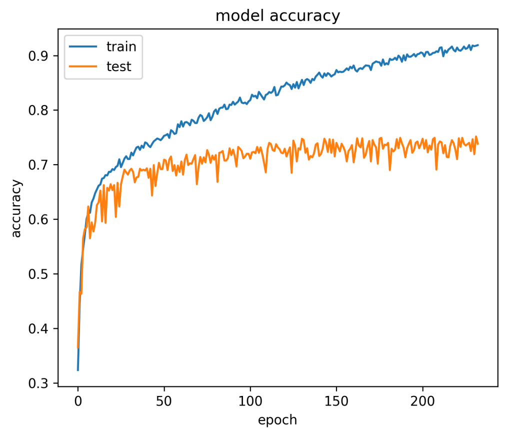

Fig. 6. Training and test accuracy plot for training the extended VGG16 model from scratch, using post only wind subset, with learning rate of 0.01 of Adadelta optimizer.

In general, training the model from scratch with data augmentation has positive influence on the post-only approach. Since we were not able to use data augmentation with concatenation, we did not train that model in this part. Without augmentation, over-fitting was a problem since we have a small set of training data as opposed to many parameters of the complex model.

For flooding, the confusion matrix is given in Table IV. For flooding, since not many samples for destroyed buildings exist, therefore lower precision and recall test scores for the destroyed class is understandable. Metrics for other classes with this experiment are given in Table V.

		Predicted			
		0	**1**	**2**	**3**
True	**0**	297	0	4	4
	1	5	240	33	11
	2	3	29	252	15
	3	2	10	25	76

TABLE IV
TEST CONFUSION MATRIX FOR FLOODING POST-ONLY VGG16 FROM SCRATCH.

	Precision	**Recall**	**F1**
0 (No-D.)	0.97	0.97	0.97
1 (Minor-D.)	0.86	0.83	0.85
2 (Major-D.)	0.80	0.84	0.82
3 (Dest.)	0.72	0.67	0.69
Accuracy:			0.86
Macro Avg:	0.84	0.83	0.83
Weighted Avg:	0.86	0.86	0.86

TABLE V
METRICS FOR FLOODING POST-ONLY VGG16 FROM SCRATCH.

For disaster type wind, all damage levels have at least 2500 samples, which makes the performance on this disaster pretty important. On the other side, detection of wind damage could be relatively harder since wind could carry building parts to areas that are not included in a building sample. With other disasters types, damaged building parts are generally still inside the building sample. Confusion matrix for post-only wind results are given in Table VI, and corresponding metrics are given in Table VII.

		Predicted			
		0	**1**	**2**	**3**
True	**0**	220	45	21	11
	1	48	259	13	8
	2	35	18	187	58
	3	24	5	54	194

TABLE VI
TEST CONFUSION MATRIX FOR WIND POST-ONLY VGG16 FROM SCRATCH.

	Precision	**Recall**	**F1**
0 (No-D.)	0.67	0.74	0.71
1 (Minor-D.)	0.79	0.79	0.79
2 (Major-D.)	0.68	0.63	0.65
3 (Dest.)	0.72	0.70	0.71
Accuracy:			0.72
Macro Avg:	0.72	0.71	0.71
Weighted Avg:	0.72	0.72	0.72

TABLE VII
METRICS FOR WIND POST-ONLY VGG16 FROM SCRATCH

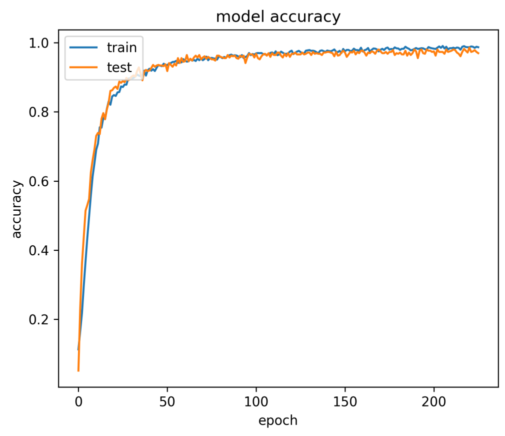

Fig. 7. Accuracy plot for training VGG16 for disaster type classification.

3) VGG16 for Disaster Type Classification: Additionally, we trained the extended VGG16 model with data augmentation for determining the disaster type based on post satellite images. Whole satellite images are resized to 224 by 224, and we used a 20% split for test. Resulting total accuracy is 0.985, and the confusion matrix is given in Table VIII. Since satellite images are coming from specific disasters that took place in specific regions, we cannot directly conclude that the models learns to classify the disaster type, it could be learning based on the features of the region rather than the artifacts of the disaster. Other metrics from this experiment is given in Table IX, and the accuracy plot is visible in Fig 7. The results suggest that we could first classify the disaster type based on the post whole image, and then use the specific model that is trained for classifying that disaster type.

	Predicted						
		Eqk.	Fire	Flood	Tsu.	Vol.	Wind
True	Eqk.	29	0	0	0	0	1
	Fire	0	205	0	1	0	0
	Flood	0	0	50	0	0	0
	Tsu.	0	2	0	22	0	0
	Vol.	0	0	0	0	2	1
	Wind	0	3	0	0	0	244

TABLE VIII
TEST CONFUSION MATRIX FOR DETERMINING THE DISASTER TYPE BASED ON THE POST SATELLITE IMAGE, USING THE EXTENDED VGG16 MODEL WITH DATA AUGMENTATION AND CLASS WEIGHT CALCULATION. DISASTER TYPE ABBREVIATIONS FROM LEFT: EARTHQUAKE, FIRE, FLOODING, TSUNAMI, VOLCANO AND WIND.

	Precision	Recall	F1
Earthquake	1.00	0.97	0.98
Fire	0.98	1.00	0.99
Flooding	1.00	1.00	1.00
Tsunami	0.96	0.92	0.94
Volcano	1.00	0.67	0.80
Wind	0.99	0.99	0.99
Accuracy:			0.99
Macro Avg:	0.99	0.92	0.95
Weighted Avg:	0.99	0.99	0.99

TABLE IX
METRICS FOR DETERMINING THE DISASTER TYPE BASED ON THE POST SATELLITE IMAGE, USING THE EXTENDED VGG16 MODEL WITH DATA AUGMENTATION AND CLASS WEIGHT CALCULATION.

4) Support Vector Machine (SVM): To experiment feature extraction described in section A.2, we have also tried SVM. Because this part of the project was done in Matlab which does not provide multi-class SVM thus, we have used SVM classifier done by Er.Abbas Manthiri BE, and published on MatWorks [5]. However, the LBP patterns remained meaningless to describe the data as the loss was over 0.5 for any number of neighbors and accuracy was 0.25 which is equal to the random labelling accuracy rate. This concluded that support vector machine does not work with the LBP features. The feature data we have extracted was not linearly divisible.

We have also experimented with MSER features and SVM. The technique that we applied to construct feature extraction to classify was to count the number of regions that change appear, compute the summation of the region pixels that change appear, and concatenation of both summation and the count. However, SVM still did not worked as the loss was over 0.5 for every extraction we applied. We have also tried to compute a binary image of absolute difference before computing the MSER features; however it even caused higher loss and less accuracy. Therefore, we do not represent any results for the SVM classifiers and just mention it did not work for the solution of this problem.The cause of the problem can be either LBP texture descriptors are not suitable for the data or the Support Vector Machine is not suitable for the extracted features.

5) Discriminant Analysis: The other classification we have tried to classify our data with respect to features extracted manually was discriminant analysis. The confusion matrix we get for discriminant analysis is given in Fig 1, and it is the best classifier that we found out of the experimented classifiers (k Nearest Neighbors, Naive Bayes and Support Vector Machines.)

On the other hand, for MSER features; the discriminant analysis did not worked as good as LBP features. The best discriminant calculated with MSER features, given as in the Fig 8. The cause of poor classification could be caused by the over fitting because we extract nearly 50-70 features for every image and there are 200 images in the train data.

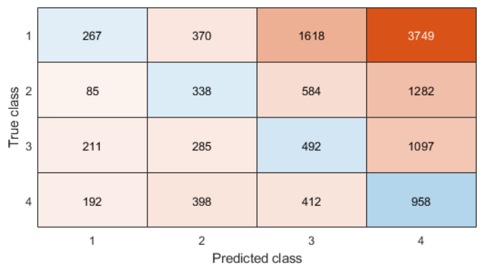

Fig. 8. Classification of MSER features with Discriminant Analysis

E. Conclusion

In this project, using multiple approaches we tried building damage classification based on satellite imagery. The dataset we use belongs to an ongoing challenge, and therefore we do not have a direct comparison to other approaches that use the same dataset. However, there are articles such as [1] that focus on a similar problem with various earthquake datasets using CNNs. Our earthquake damage assessment accuracy is similar to their results, however, since we do not know their class distribution, it is hard to compare. Moreover, our classification attempts with non state-of-the-art techniques did not work very well compared to neural network. We have also tried our own feature extraction techniques however the already-given buildings were seem to work more accurately thus we have not trained our neural network model with our own feature extractor algorithms.

REFERENCES

[1] Xu, Joseph Z., et al. ”Building Damage Detection in Satellite Imagery Using Convolutional Neural Networks.” arXiv preprint arXiv:1910.06444 (2019).

[2] Dornaika F., Ruichek, Y. et al. ”A Comparative Study of Image Segmentation Algorithms and Descriptors for Building Detection.” sciencedirect.com, (2017).

[3] P. Forss´en, D. Lowe, S-H Wang et al. ”Region detectors.” micc.unifi.it. (2011)

[4] A. Lipika, ”Building detection in VHR Satellite Imagery by using Fast ICA and MSER.” International Institute of Information Technology Hyderabad - 500 032, INDIA(2017).

[5] Abbas Manthiri S., ”Multi Class SVM” , MATLAB Central File Exchange.(2017).