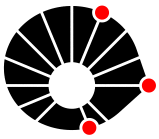
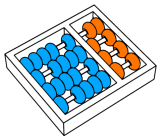




Event Repurposing Detection based on Image analysis

Gabriel Capiteli Bertocco *Anderson Rocha*

Relatório Técnico - IC-PFG-18-27
Projeto Final de Graduação
2018 - Dezembro

UNIVERSIDADE ESTADUAL DE CAMPINAS
INSTITUTO DE COMPUTAÇÃO

Event Repurposing Detection based on Image Analysis

Gabriel Bertocco* Anderson Rocha*

Abstract

Nowadays one of the greatest problems faced in social and electronics media is the attempt to change the purpose of images in order to increase or change the impact of some event. For example, a malicious person can use a picture depicting a specific event to illustrate another completely event, leading to misunderstanding and changing the public opinion about the event or related topics. This repurpose of the meaning of the original picture is called Event Repurposing. In this work, we propose two methods to detect if a image is being repurposed or not (binary classification problem) based on the full scene analysis (using the whole image) and on the analysis of the objects (cars, people, trucks, and so on) present in the scene. In order to model the binary classification, we extract features from the images using a Deep Convolutional Neural Network (DCNN) and train an One-Class SVM. To check the robustness of the proposed methods, we consider eight different events with different objects and landscapes to get variability in terms of scenario, context and purpose of the events.

1 Introduction

Nowadays almost everyone around the world has access to information available on the Internet. People now have the power to generate content about any subject, such as politics and sports, and spread it on social medias. Recently, we are experiencing an era where people use this power in order to create fake news and to impact public opinion on a particular subject. One way to try to induce some idea to the public is to change the purpose of an image, i.e., reuse an image to achieve a new interpretation of the event in which the image is inserted on, as shown in Figure 1. In this example, which occurred during the 2018 Brazilian elections, images of the Carnival 2017 were used to illustrate the manifestation NotHim (EleNão, in Potuguese) against the president candidate Jair Bolsonaro. Both events happened in the same place in, in São Paulo. The name of this process is Event Repurposing based on Images. In this context, the goal of the present work is to detect if an image or set of images were repurposed to another event based on the analysis of images truly taken on the original event using Machine Learning and Pattern Analysis techniques. Our method is divided in two steps: detection of objects of interest and training of classifiers based on features extracted from those objects (Object of Interest Analysis) and features extracted from whole images of the event (Full Image Analysis). The inference is done by a weighted combination of the outputs of the classifiers trained for each object of interest or, in the case of the analysis of the full image, by a unique answer given by a classifier trained with features from the whole image. The rationale is to compare the performance of the classifiers when considering only components (objects of interest)

*Instituto de Computação, Universidade Estadual de Campinas, 13081-970 Campinas, SP

of the event with the performance of the classifiers trained with the whole scene understanding. We conduct experiments on images from eight different events released by by DARPA [1].

This report is divided in the following sections: in Section 2, we explain the methodology for each step and the full pipeline of the two methods (one based on objects of interest and another based on full images). In Section 3, we describe the dataset and the events considered in the experiments, and in Section 4, we present the experiment protocol and the results for each method and for each of the eight events. Finally, Section 5 states the conclusions and future work.

(a) Manifestation NotHim (EleNao, in Portuguese) in São Paulo, Brazil, 2018, against the president candidate Jair Bolsonaro.

(b) Carnival in São Paulo, Brazil, 2017.

Figure 1: Image (b) was repurposed to the event depicted in (a).

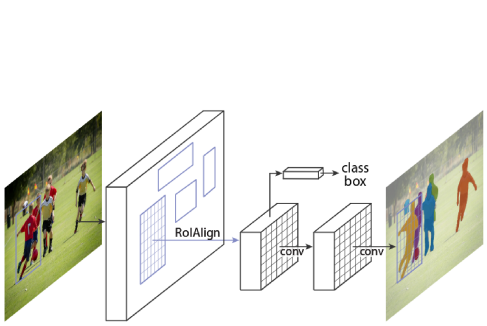
2 Methodology

We explore the problem of Event Repurposing Detection in two fronts: one considering the analysis of the full images in the training and inference stages, and one considering the analysis of objects of interest in the images (cars, people, trucks, airplanes, backpacks, bicycles, and another possible objects present in the scenes). We will first explain the feature extraction step used in the both methods and then the full pipeline of each one.

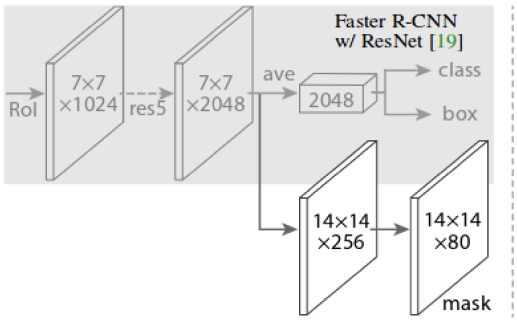
2.1 Feature Extraction

In the last years, Deep Learning has grown a lot due to its capability to model and solve several problems in computer vision, medical images [2], economics [3], audio [4], text analysis [5] and another great fields involving a lot of data. This capability comes from the high complexity of the features produced by Deep Learning techniques, allowing the computation of high non-linear models to divide the hyper-space in which the data is represented, providing high accuracies in many of those cited problems. Since a deep model and, more specifically in this work, a Deep Convolutional Neural Network (DCNN), produces representative features, we use it to extract features from the whole images and from the objects of interest which we expect to have a good representation about what is being seen by the DCNN in the input image. To do so, we choose Mask R-CNN [6] proposed by Facebook AI Research, which was trained to classify, detect (find a bounding box), and segment up to 91 objects of interest - cars, people, bicycles, trucks, backpacks, chairs, laptops and another important objects to an event analysis. Mask R-CNN (Figure 2) has three outputs and three

respective loss functions: one for object classification, one for object detection, and one for segmentation. The ground truth fed to train each objective is: a probability distribution (softmax over all the possible 91 possibilities of objects) for the classification loss (L_{class}); set of four real numbers, one for each corner of the bounding box for the detection (L_{reg}); and a binary mask representing the region occupied by the object in the image for the segmentation loss (L_{seg}). The final loss is the sum of those loss functions equally weighted: $L_{final} = L_{class} + L_{reg} + L_{seg}$. The illustration of the outputs from each branch of Mask R-CNN is shown in the Figure 3. Therefore, Mask R-CNN puts together two main advantages for this work: it is able to detect as many objects as possible in only one forwarding of the image in the network and, because of the high confidences reported for each output, the network generates very representative features for each object detected, as well as for the whole image. The layer from which we extract features is the last one before the ramification of the net to each one of three branches.



(a) Mask R-CNN pipeline.



(b) Last layers of Mask R-CNN before the division in three branches.

Figure 2: Mask R-CNN.

(a) Original image.

(b) Image with the three outputs of the Mask R-CNN: the bounding box, the mask and the predicted class.

Figure 3: Example of the three outputs of Mask R-CNN.

2.2 Full Image Analysis

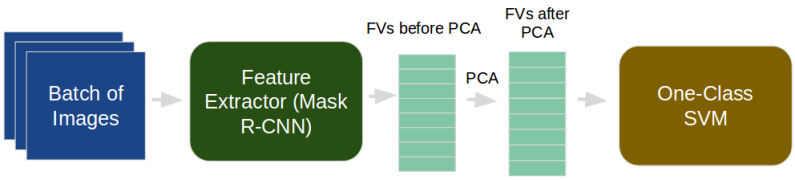


Figure 4: Pipeline for the Full Image analysis.

Regarding the full Image Analysis, we forward an image through the Mask R-CNN and get the output of the last layer before the ramification. When the network receives an entire image of an event, we hypothesize that the extracted feature vector contains a full representation of all elements depicted in the image and holds correlations and associations between them. By extracting feature vectors for several images of an event, we can train a external classifier with such feature vectors to detect patterns of that event. The pipeline for this method is shown in Figure 4. After extracting the feature vectors we can apply PCA to perform dimensionality reduction or keep the dimension of the feature vectors. The two cases were tested to check the robustness of the methods.

2.3 Object of Interest Analysis

The second method, in contrast to previous one that analyzes full images, is based on the analysis of each component of the image separately. The first step is to detect the components by forwarding an input image through the network and considering three outputs: the mask, the bounding box, and the accuracy for each predicted component. The components can be cars, people, trucks, backpacks, bicycles and another objects that might be useful to represent the event. The second step is to crop each component using its predicted bounding box and, using the predicted mask, paint all background pixels in black. This will give us images of each component separately, with black background (Figures 5 and 6). Then, we forward these new component images through the Mask R-CNN and take the feature vectors from the same layer as in the previous method. By considering the feature vectors extracted from the new component images, where the only representative contents are the components themselves, we expect to have good representations for each component separately, also considering that the DCNN has high accuracy in classification, detection and segmentation of the considered classes of objects.

(a) Original Image.

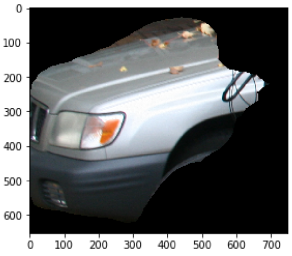
(b) Detected component (person with the predicted mask and bounding box).

(c) New image, with the detected person only and black background.

Figure 5: Example of creating a new component image (person).

(a) Original Image.

(b) Detected components (People, car, bicycle and backpacks), with the predicted masks and bounding boxes.



(c) New image, with one detected component (car) only and black background.

Figure 6: Example of creating a new component image (car).

Once we do this for all training images, we will have feature vectors for all components in all images among the 91 possibilities given by Mask R-CNN. Then, for each set of feature vector belonging to a specific component, we will train a classifier in order to be able to analyze the event based only on a specific component. Doing this for all components, we will have a set of classifiers for each one in order to be able to analyze the event based on the objects of interest detected during the training phase.

During the inference phase, we consider all trained classifiers for the considered event. So we performed a weighted combination of the answers given by the classifiers, based on the frequency of the components (objects of interest in the training data), in the following way: Let M be the total number of feature vectors in the training phase and n_i the number of feature vectors detected for component i, with $i \in \{1, 2..., 91\}$. Let C be the number of classifiers which is equal to the number of non-zero elements of M. Weight w_j for the output of classifier C_j is given by $w_j = 1 - n_j/M$. As each classifier outputs a number $p \in \{-1, +1\}$ in which -1 means that the classifier predicts that the query image does not belong to event and $+1$ that query image does belong to the event, the

final answer combining all the classifier outputs is giver by:

$$Final = \sum_{j=1}^{C} w_j p_j$$

If $Final < 0$ the query image is assigned as not belonging to the event, otherwise the query image is assigned as belonging to the event. Note that the weight of a given component j is $1 - n_j/M$, so the greater is n_j the lower will be the weight, that is, the more frequent is the component j, the lower will be its importance in the decision. Since a component is very frequent in the event it might be very frequent in order events (people, for example, which appear as object of interest in all events), so it might not be helpful to classify if a image is or is not belonging to the event, since it might not be a particularity of the event. This pipeline is showed in the Figure 7

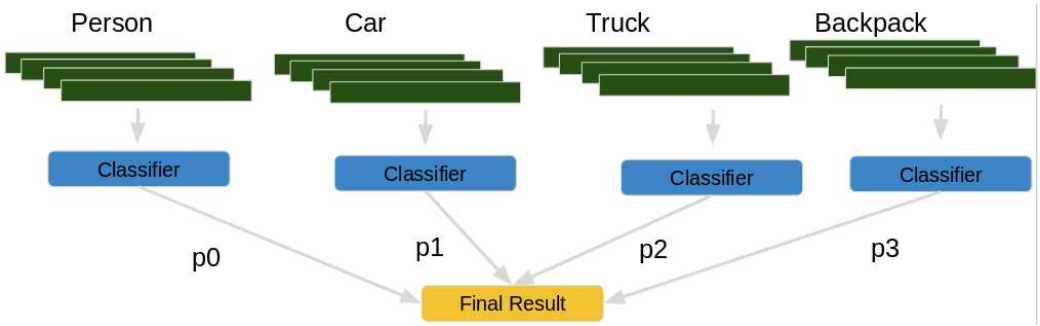


Figure 7: Classification by weight combining the outputs from each classifier. In this figure we show to illustrate only the feature vectors and classification for 4 components detected in the training: Person, Car, Truck and Backpack.

2.4 One-Class SVM

The goal of this project is to detect if an image is being repurposed or not based on deep feature analysis. For an event, we have to extract features of images depicting this event using a DCNN and then train a classifier using them in order to create a hyper-plane in the hyper-space to distinguish features of the event from other features. To do this, we apply a novelty detection technique whose main goal is to detect if a new feature vector is an outlier based on the pattern analysis of the training feature vectors. If the new feature vector is an inlier then it is assigned as belonging to the event, otherwise the new feature vector is an outlier and it is assigned as not belonging to the event. The novelty detection algorithm used here is the One-Class SVM [7] which create a hyper-sphere around the feature vectors belonging to the event and any new feature vector inside the hyper-sphere is an inlier and outside the hyper-sphere is an outlier, as shown in the Figure 8.

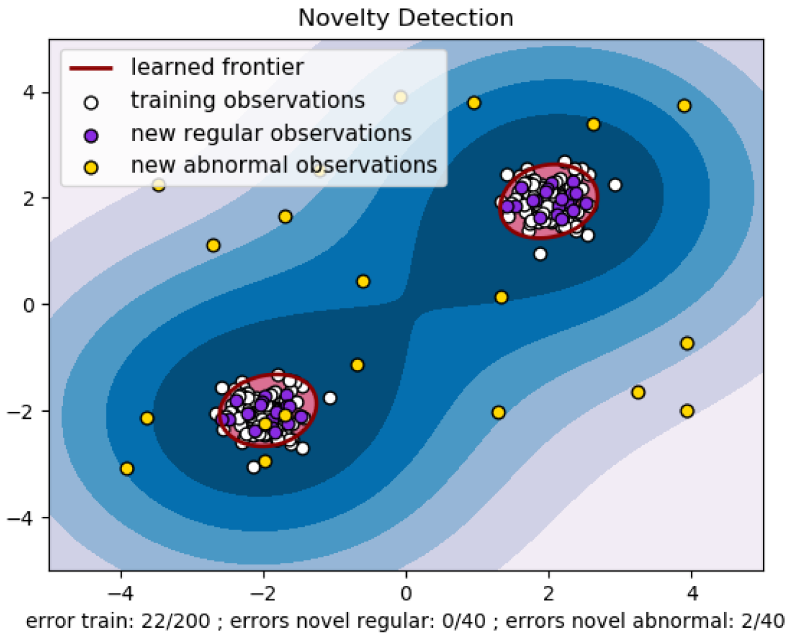


Figure 8: Example of the One-Class SVM hyper-plane division (projected in the 2D space).

To train the One-Class SVM, we need to tune the parameter ν which controls the probability of finding a regular feature vector that corresponds to the event outside the hyper-sphere, i.e., a parameter which controls the radius of the hyper-sphere. In this work, we train the two models with $\nu = 0.05$, $\nu = 0.1$, $\nu = 0.15$.

3 Dataset

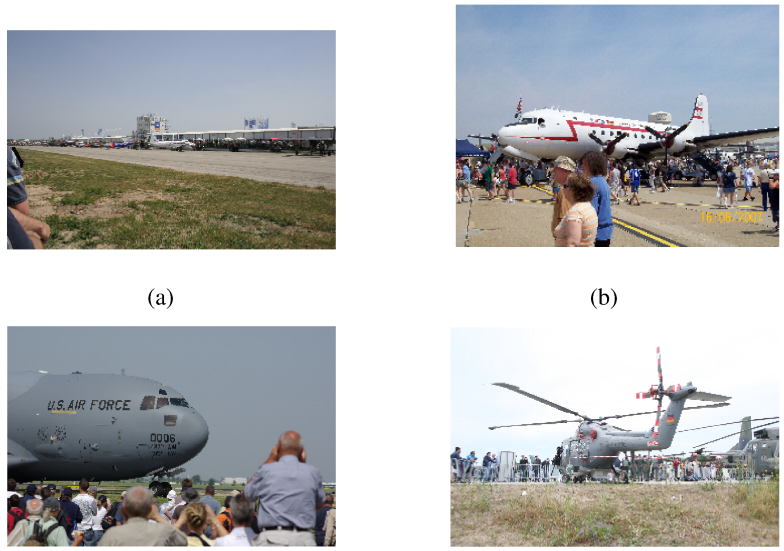
In this section, we explain the dataset used in the experiments of the proposed methods. The dataset was originally released by DARPA [1] to an Event Repurposing competition. There are eight events with 200 images each one. The events are: Austin Marathon (Figure 9), Boston Marathon (Figure 12), Berlin Air Show (Figure 10), Hurricane Matthew (Figure 13), Hurricane Sandy (Figure 14), Occupation of Portland (Figure 15), Occupation of Baltimore (Figure 11), and the event of Oshkosh Air Show happened in 2011 (Figure 16). The images have high resolution and dimension (from 700 x 1500 up to 2000 x 3700), so we can explore all the details with the Mask R-CNN. There are indoor and outdoor images, selfies of the people, photos in landscape and portrait orientations, pictures of food, cars, streets and buildings; images with different illumination conditions, and from different points of view. Some images are in gray scale. Another images have more hints about the depicted event than others that do not seem to belong to the event. We call this last set of images as hard images (Figure 17). We train over 160 images with high resolution and high dimensions and validate over 40 images belongs to the same event and all of the 200 images present in each of other

events. We used all of the images that do not correspond to the event as negative images in in the validation, since to train One-Class SVM is necessary only feature vector of the event of interest.

(a) (b)

(c) (d)

Figure 9: Example of images from the Austin Marathon.



(a) (b)

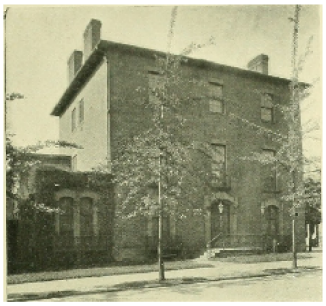
(c) (d)

Figure 10: Example of images from the Berlin Air Show.

(a)

(b)

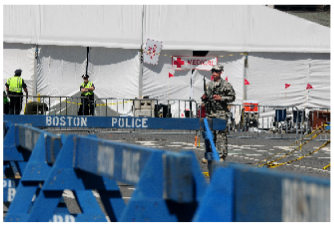
(c)



(d)

Figure 11: Example of images from the Occupy Baltimore.

(a)

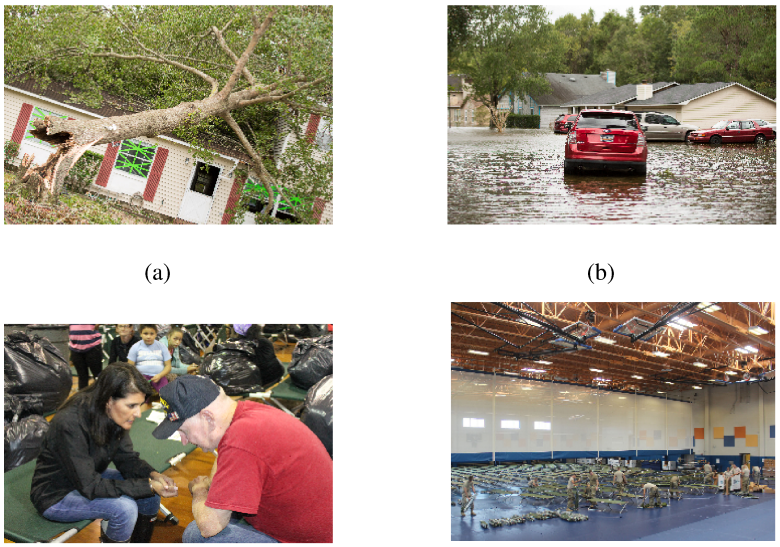


(b)

(c)

(d)

Figure 12: Example of images from the Boston Marathon.



(a) (b)

(c) (d)

Figure 13: Example of images from the Hurricane Matthew.

(a) (b)

(c) (d)

Figure 14: Example of images from the Hurricane Sandy.

(a) (b)

(c) (d)

Figure 15: Example of images from Occupy Portland event.

(a) (b)

(c) (d)

Figure 16: Example of images from the Oshkosh Air Show, 2011.

(a) (b) (c) (d)

Figure 17: Example of hard images that do not have explicity hints about the depicted event.

4 Experiments and Results

As we consider eight events, we applied the Full Image Analysis and the Objects of Interest Analysis pipelines to each one of the events. For each event, we split all of the 200 images in training (80%) and validation (20%) sets, extracted all the features for the two pipelines, using the Keras [8] implementation of Mask R-CNN, and then we trained the classifiers using scikit-learn library [9]. In the case of the Full Image Analysis, we have only one classifier, as we have only one feature vector by image. In the Objects of Interest Analysis, we have a variable number of classifiers as it depends on the number of components detected in the training set. The feature vectors have 81000 dimensions, so it is represented by $v \in \Re^n$ with $n = 81000$, which is a number much higher than the number of available images and higher than the number of feature vector for each component. So, to see the impact of the dimensionality, we applied the Principal Component Analysis [10] (PCA) in order to project the vector from 81000 dimensions to 200 dimensions. This number was chosen based on the number of images for each event. The decomposition inside PCA is directed by the lower dimension, and it was not possible to reduce to dimensions higher than 200, unless we had more images to train.

As we are using a SVM, we tested two kernels: linear and RBF kernels. We also changed the ν parameter as explained in Section 2. Therefore we have a total of 12 combinations for each pipeline for all events. The results are shown below in Tables 1 to 6. For all tables, the first column refers to the location of each event, meaning that the related classifiers were trained over the images of the specific event. The main diagonal contains the rate of acceptance — images classified correctly as belonging to the event (True Positive Rate). In other positions, we have the rejection rates — images classified incorrectly as the event in the column (True Negative Rate). For instance, considering the first line in Table 1, the classifiers were trained for the Austin Marathon (with images from this event) using SVM with linear kernel, allowing 5% of the features of the event to be misclassified ($\nu = 0.05$), without PCA, and using the Full Image Analysis pipeline. We got 87.5% of images from this event correctly classified, 12.5% of the Berlin Air Show correctly rejected, 42.2% of the Boston Marathon correctly rejected, 47.8% of the Hurricane Matthew correctly rejected and so on. Below we presented only the best results for each value of ν for each pipeline. We present a plot (Figure 18) with the mean True Positive Rate (sum the main diagonal up and divide by the number of events) for each value of ν.

Results for Linear kernel and without PCA using Full Image Analysis pipeline								
	Austin	Berlin	Boston	Matthew	Sandy	Baltimore	Portland	Oshkosh
Austin	0.875	0.125	0.422	0.478	0.665	0.643	0.490	0.275
Berlin	0.795	0.700	0.765	0.743	0.805	0.755	0.869	0.265
Boston	0.320	0.185	0.818	0.570	0.795	0.617	0.475	0.325
Matthew	0.195	0.055	0.256	0.565	0.430	0.342	0.328	0.030
Sandy	0.245	0.080	0.325	0.443	1.000	0.235	0.232	0.055
Baltimore	0.495	0.145	0.394	0.543	0.555	0.846	0.369	0.130
Portland	0.315	0.090	0.336	0.317	0.510	0.383	0.769	0.085
Oshkosh	0.685	0.180	0.715	0.704	0.735	0.755	0.747	0.825

Table 1: $\nu = 0.05$

Results for Linear kernel and without PCA using Full Image Analysis pipeline								
	Austin	Berlin	Boston	Matthew	Sandy	Baltimore	Portland	Oshkosh
Austin	0.875	0.125	0.422	0.478	0.665	0.643	0.490	0.275
Berlin	0.850	0.700	0.765	0.787	0.895	0.908	0.894	0.365
Boston	0.320	0.205	0.782	0.657	0.855	0.633	0.480	0.365
Matthew	0.205	0.055	0.390	0.565	0.465	0.378	0.333	0.035
Sandy	0.245	0.085	0.329	0.443	0.950	0.235	0.253	0.055
Baltimore	0.510	0.145	0.444	0.548	0.585	0.718	0.389	0.165
Portland	0.310	0.090	0.336	0.317	0.510	0.383	0.769	0.085
Oshkosh	0.680	0.180	0.733	0.722	0.735	0.699	0.773	0.825

Table 2: $\nu = 0.10$

Results for Linear kernel and without PCA using Full Image Analysis pipeline								
	Austin	Berlin	Boston	Matthew	Sandy	Baltimore	Portland	Oshkosh
Austin	0.850	0.125	0.462	0.478	0.665	0.668	0.520	0.280
Berlin	0.850	0.675	0.773	0.787	0.910	0.923	0.914	0.365
Boston	0.365	0.215	0.782	0.787	0.860	0.633	0.500	0.365
Matthew	0.210	0.080	0.397	0.761	0.470	0.408	0.379	0.070
Sandy	0.245	0.085	0.329	0.457	0.925	0.240	0.263	0.055
Baltimore	0.585	0.165	0.448	0.557	0.610	0.692	0.384	0.185
Portland	0.310	0.090	0.404	0.504	0.505	0.383	0.744	0.120
Oshkosh	0.750	0.190	0.758	0.739	0.740	0.811	0.843	0.700

Table 3: $\nu = 0.15$

Results for Linear kernel and with PCA using Object of Interest Analysis pipeline								
	Austin	Berlin	Boston	Matthew	Sandy	Baltimore	Portland	Oshkosh
Austin	0.725	0.785	0.315	0.735	0.605	0.455	0.515	0.900
Berlin	0.245	0.850	0.265	0.655	0.545	0.385	0.405	0.270
Boston	0.255	0.690	0.650	0.680	0.560	0.385	0.385	0.860
Matthew	0.145	0.490	0.115	0.550	0.365	0.310	0.280	0.705
Sandy	0.170	0.385	0.160	0.435	0.375	0.370	0.320	0.550
Baltimore	0.230	0.680	0.220	0.655	0.565	0.325	0.410	0.855
Portland	0.200	0.660	0.215	0.655	0.525	0.390	0.700	0.860
Oshkosh	0.110	0.070	0.135	0.500	0.385	0.350	0.260	0.875

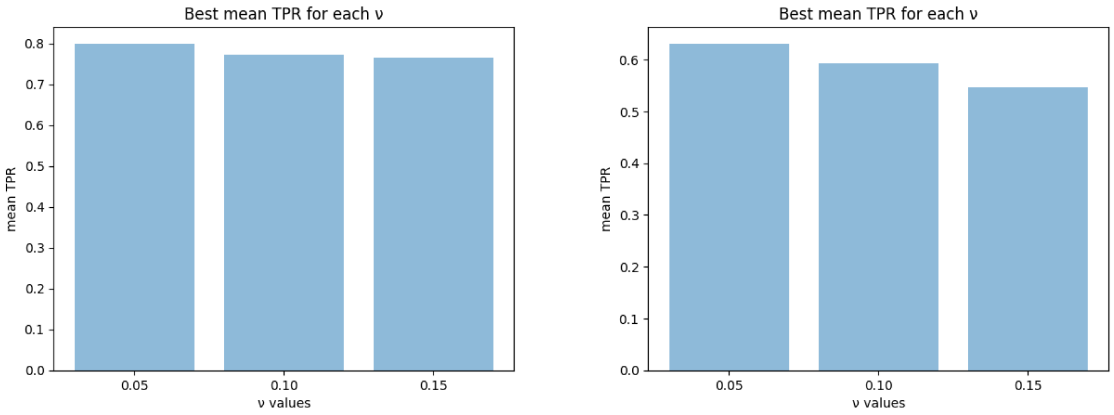
Table 4: $\nu = 0.05$

Results for Linear kernel and without PCA using Object of Interest Analysis								
	Austin	Berlin	Boston	Matthew	Sandy	Baltimore	Portland	Oshkosh
Austin	0.700	0.800	0.365	0.740	0.625	0.460	0.540	0.915
Berlin	0.255	0.825	0.285	0.675	0.580	0.405	0.420	0.305
Boston	0.265	0.715	0.625	0.710	0.575	0.395	0.405	0.870
Matthew	0.165	0.535	0.125	0.500	0.375	0.335	0.305	0.730
Sandy	0.185	0.440	0.205	0.455	0.300	0.375	0.355	0.570
Baltimore	0.245	0.690	0.250	0.665	0.575	0.275	0.410	0.860
Portland	0.225	0.760	0.290	0.665	0.540	0.405	0.650	0.875
Oshkosh	0.110	0.095	0.150	0.520	0.395	0.350	0.270	0.875

Table 5: $\nu = 0.10$

Results for Linear kernel and with PCA using Object of Interest Analysis								
	Austin	Berlin	Boston	Matthew	Sandy	Baltimore	Portland	Oshkosh
Austin	0.625	0.870	0.420	0.775	0.640	0.480	0.555	0.925
Berlin	0.290	0.700	0.310	0.705	0.605	0.425	0.465	0.375
Boston	0.270	0.730	0.600	0.710	0.580	0.395	0.410	0.875
Matthew	0.175	0.570	0.145	0.500	0.380	0.340	0.325	0.735
Sandy	0.200	0.495	0.250	0.475	0.300	0.380	0.385	0.595
Baltimore	0.280	0.735	0.335	0.680	0.615	0.275	0.445	0.875
Portland	0.265	0.790	0.375	0.680	0.570	0.410	0.600	0.880
Oshkosh	0.130	0.115	0.160	0.535	0.405	0.360	0.285	0.775

Table 6: $\nu = 0.15$



(a) Full Image Analysis pipeline

(b) Object of Interest pipeline

Figure 18: Best mean TPR for each value of ν for each pipeline.

Analyzing the results, we can see that using the liner kernel without PCA outperforms other combinations. This happens mainly because of the sparsity of the feature vectors in the vector space.

In the Full Image Analysis, we have only 200 feature vectors with 81000 dimensions which resulted a very sparse distribution of data in the vector space. When we applied the RBF kernel, all the feature vectors are projected to a greater dimension, and then the training is performed. Since we have a sparser data in the greater dimension, the feature vectors are more distant among themselves, and then the rejection rate with RBF kernel is higher than with linear kernel. In the appendix we see all the results with RBF kernel that shows this behaviour, in which we have all features vectors from other events being fully rejected and some feature vectors of the depicted event being accepted, in order words, high TNR and low TPR. The same explanation is applied to the Object of Interest pipeline.

The effect of PCA is interesting: when the liner kernel is applied with and without PCA the results tend to be very similar which show us the difference of dimensions did not influence neither the training neither the test using One-Class SVM with linear kernel. In the RBF kernel, the dimension has impact over the results. Since we are reducing the dimension (with PCA) and then expanding (with RBF kernel), the feature vectors are affected in a way that the rejections increases a lot even to depicted event (TPR decreases) as to the other event (TNR increases).

In terms of ν value, we have the expected conclusion: as ν increases (the radius of the hypersphere decreases) the number of rejected examples also increases, since we allow that more examples belonging to the depicted event be misclassified. We can check this result looking to the main diagonal in which the values decrease (TPR decreases) and in the values of the other events (TNR increases). This conclusion can be observed in the two pipelines.

In the plot, we see the values of mean TPR to each of the six tables above. Full Image Analysis pipeline reported a better result to all three values of ν than the Object of Interest Analysis pipeline, so we conclude that features from a whole understanding of the scene give us better performance than features extracted from objects.

There are some cases that Object of Interest Analysis pipeline outperforms a lot the Full Image

Analysis pipeline. For instance, when we see the line of Austin and column of Oshkosh for $\nu = 0.15$ for the two pipelines, they reported 28% for Full Image Analysis pipeline and 92.5% for Object of Interest Analysis pipeline, this great gap between them is explained because the objects of interest in the both events have significant different frequencies. For example, Oshkosh has a lot of airplanes while Austin does not have airplanes since one refers to a marathon and another one to the Air Show. That is, different events have different objects and those objects depend on the context, scenario and purpose of the event which helps a lot the Object of Interest analysis pipeline. The same great difference between results we see in the same line of Austin and column of Berlin which is also a Air Show.

Even in the Full Image Analysis we have events that reports high TPR and high TNR (Berlin line in table 1) and events with low TPR and low TNR (Portland line in Table 2), which show us that research is still needs to improve the results to the two pipelines and, as next step, combine the results from two pipelines in a single one to take advantage of the full scene understanding and of the objects of interest understanding to improve the results.

5 Conclusion and Future Work

In this work, we presented two pipelines to detect repurpose of images, that is, attempt to use the original meaning of the event to another purpose. One pipeline is based on the Full Image Analysis and the other one in the Object of Interest analysis and the results show us that, in general, the Full Image Analysis helps more than the Object of Interest Analysis since the first one is considering the whole scene instead of see the objects individually. However, events with different frequencies of components may be better distinguishable using Object of Interest Analysis since we can have different objects between events. Even with some good results to some events, research is still need to improve them for the two pipelines and take advantage from the combination of both of them in order to have a complementary result analyzing the whole scene and objects of it. So the next steps are explore the combination of the pipelines, explore another classifiers besides one-Class SVM, use another DCNNs to describe the scene and detect stuffs that Mask-R CNN does not detect and them combine the features to have richer descriptions about the scenes of the event. We also aim to methods that involve graphs to take advantage of the relations established by this data structure between its nodes. Finally, we aim to get more data to have a better representation and variability of the events and then better robustness of the descriptors and classifiers.

References

[1] "Defense advanced research projects agency," https://www.darpa.mil, Accessed: 2018-06-26.

[2] Dinggang Shen, Guorong Wu, and Heung-Il Suk, "Deep learning in medical image analysis," *Annual review of biomedical engineering*, vol. 19, pp. 221–248, 2017.

[3] John H Holland and John H Miller, "Artificial adaptive agents in economic theory," *The American economic review*, vol. 81, no. 2, pp. 365–370, 1991.

[4] Honglak Lee, Peter Pham, Yan Largman, and Andrew Y Ng, "Unsupervised feature learning for audio classification using convolutional deep belief networks," in *Advances in neural information processing systems*, 2009, pp. 1096–1104.

[5] Xiang Zhang, Junbo Zhao, and Yann LeCun, "Character-level convolutional networks for text classification," in *Advances in neural information processing systems*, 2015, pp. 649–657.

[6] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick, "Mask r-cnn," in *Computer Vision (ICCV), 2017 IEEE International Conference on*. IEEE, 2017, pp. 2980–2988.

[7] Bernhard Schölkopf, John C Platt, John Shawe-Taylor, Alex J Smola, and Robert C Williamson, "Estimating the support of a high-dimensional distribution," *Neural computation*, vol. 13, no. 7, pp. 1443–1471, 2001.

[8] François Chollet et al., "Keras," https://keras.io, 2015.

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

[10] N. Halko, P.-G. Martinsson, and J. A. Tropp, "Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions," *arXiv preprint arXiv:0909.4061*, 2009.

A Tables of Results to other configurations of kernel and PCA

Below we have the other results that we had during the test of many combinations of kernel, use of PCA, pipeline method and ν. There are lines in some tables that are full filled with 0, and -1 in the main diagonal. This happened only when we tried to apply PCA and failed since the number of vectors available were lower than 200 (the target number to be reduced to). So we simply unconsidered those results. All of the explanation behind the values were done in section 4.

Results for Linear kernel and with PCA using Full Image Analysis pipeline								
	Austin	Berlin	Boston	Matthew	Sandy	Baltimore	Portland	Oshkosh
Austin	0.875	0.125	0.422	0.478	0.665	0.643	0.490	0.275
Berlin	0.795	0.950	0.765	0.743	0.805	0.755	0.869	0.265
Boston	0.000	0.000	-1.000	0.000	0.000	0.000	0.000	0.000
Matthew	0.195	0.055	0.256	0.783	0.430	0.342	0.328	0.030
Sandy	0.245	0.080	0.325	0.443	0.850	0.235	0.232	0.055
Baltimore	0.495	0.145	0.394	0.543	0.555	0.641	0.369	0.130
Portland	0.315	0.090	0.336	0.317	0.510	0.383	0.821	0.085
Oshkosh	0.685	0.180	0.715	0.704	0.735	0.755	0.747	0.725

Table 7: $\nu = 0.05$

Results for RBF kernel and without PCA using Full Image Analysis pipeline								
	Austin	Berlin	Boston	Matthew	Sandy	Baltimore	Portland	Oshkosh
Austin	0.275	1.000	1.000	1.000	1.000	1.000	1.000	1.000
Berlin	1.000	0.225	1.000	1.000	1.000	1.000	1.000	1.000
Boston	1.000	1.000	0.600	1.000	1.000	1.000	1.000	1.000
Matthew	1.000	1.000	1.000	0.000	1.000	1.000	1.000	1.000
Sandy	1.000	1.000	1.000	1.000	0.150	1.000	1.000	1.000
Baltimore	1.000	1.000	1.000	1.000	1.000	0.205	1.000	1.000
Portland	1.000	1.000	1.000	1.000	1.000	1.000	0.513	1.000
Oshkosh	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.500

Table 8: $\nu = 0.05$

Results for RBF kernel and with PCA using Full Image Analysis pipeline								
	Austin	Berlin	Boston	Matthew	Sandy	Baltimore	Portland	Oshkosh
Austin	0.325	1.000	1.000	1.000	1.000	1.000	1.000	1.000
Berlin	1.000	0.350	1.000	1.000	1.000	1.000	1.000	1.000
Boston	0.000	0.000	-1.000	0.000	0.000	0.000	0.000	0.000
Matthew	1.000	1.000	1.000	0.457	1.000	1.000	1.000	1.000
Sandy	1.000	1.000	1.000	1.000	0.900	1.000	1.000	1.000
Baltimore	1.000	1.000	1.000	1.000	1.000	0.333	1.000	1.000
Portland	1.000	1.000	1.000	1.000	1.000	1.000	0.590	1.000
Oshkosh	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.800

Table 9: $\nu = 0.05$

Results for Linear kernel and with PCA using Full Image Analysis pipeline								
	Austin	Berlin	Boston	Matthew	Sandy	Baltimore	Portland	Oshkosh
Austin	0.875	0.125	0.422	0.478	0.665	0.643	0.490	0.275
Berlin	0.850	0.725	0.765	0.787	0.895	0.908	0.894	0.365
Boston	0.000	0.000	-1.000	0.000	0.000	0.000	0.000	0.000
Matthew	0.205	0.055	0.390	0.783	0.465	0.378	0.333	0.035
Sandy	0.245	0.085	0.329	0.443	0.850	0.235	0.253	0.055
Baltimore	0.510	0.145	0.444	0.548	0.585	0.641	0.389	0.165
Portland	0.310	0.090	0.336	0.317	0.510	0.383	0.744	0.085
Oshkosh	0.680	0.180	0.733	0.722	0.735	0.699	0.773	0.725

Table 10: $\nu = 0.10$

Results for RBF kernel and without PCA using Full Image Analysis pipeline								
	Austin	Berlin	Boston	Matthew	Sandy	Baltimore	Portland	Oshkosh
Austin	0.125	1.000	1.000	1.000	1.000	1.000	1.000	1.000
Berlin	1.000	0.250	1.000	1.000	1.000	1.000	1.000	1.000
Boston	1.000	1.000	0.418	1.000	1.000	1.000	1.000	1.000
Matthew	1.000	1.000	1.000	0.000	1.000	1.000	1.000	1.000
Sandy	1.000	1.000	1.000	1.000	0.475	1.000	1.000	1.000
Baltimore	1.000	1.000	1.000	1.000	1.000	0.513	1.000	1.000
Portland	1.000	1.000	1.000	1.000	1.000	1.000	0.641	1.000
Oshkosh	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.725

Table 11: $\nu = 0.10$

Results for RBF kernel and with PCA using Full Image Analysis pipeline								
	Austin	Berlin	Boston	Matthew	Sandy	Baltimore	Portland	Oshkosh
Austin	0.350	1.000	1.000	1.000	1.000	1.000	1.000	1.000
Berlin	1.000	0.425	1.000	1.000	1.000	1.000	1.000	1.000
Boston	0.000	0.000	-1.000	0.000	0.000	0.000	0.000	0.000
Matthew	1.000	1.000	1.000	0.457	1.000	1.000	1.000	1.000
Sandy	1.000	1.000	1.000	1.000	0.725	1.000	1.000	1.000
Baltimore	1.000	1.000	1.000	1.000	1.000	0.385	1.000	1.000
Portland	1.000	1.000	1.000	1.000	1.000	1.000	0.615	1.000
Oshkosh	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.150

Table 12: $\nu = 0.10$

Results for Linear kernel and with PCA using Full Image Analysis pipeline								
	Austin	Berlin	Boston	Matthew	Sandy	Baltimore	Portland	Oshkosh
Austin	0.875	0.125	0.462	0.478	0.665	0.668	0.520	0.280
Berlin	0.850	0.625	0.773	0.787	0.910	0.923	0.914	0.365
Boston	0.000	0.000	-1.000	0.000	0.000	0.000	0.000	0.000
Matthew	0.210	0.080	0.397	0.783	0.470	0.408	0.379	0.070
Sandy	0.245	0.085	0.329	0.457	0.875	0.240	0.263	0.055
Baltimore	0.585	0.165	0.448	0.557	0.610	0.744	0.384	0.185
Portland	0.310	0.090	0.404	0.504	0.505	0.383	0.692	0.120
Oshkosh	0.750	0.190	0.758	0.739	0.740	0.811	0.843	0.575

Table 13: $\nu = 0.15$

Results for RBF kernel and without PCA using Full Image Analysis pipeline								
	Austin	Berlin	Boston	Matthew	Sandy	Baltimore	Portland	Oshkosh
Austin	0.325	1.000	1.000	1.000	1.000	1.000	1.000	1.000
Berlin	1.000	0.225	1.000	1.000	1.000	1.000	1.000	1.000
Boston	1.000	1.000	0.545	1.000	1.000	1.000	1.000	1.000
Matthew	1.000	1.000	1.000	0.304	1.000	1.000	1.000	1.000
Sandy	1.000	1.000	1.000	1.000	0.400	1.000	1.000	1.000
Baltimore	1.000	1.000	1.000	1.000	1.000	0.385	1.000	1.000
Portland	1.000	1.000	1.000	1.000	1.000	1.000	0.410	1.000
Oshkosh	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.500

Table 14: $\nu = 0.15$

Results for RBF kernel and with PCA using Full Image Analysis pipeline								
	Austin	Berlin	Boston	Matthew	Sandy	Baltimore	Portland	Oshkosh
Austin	0.575	1.000	1.000	1.000	1.000	1.000	1.000	1.000
Berlin	1.000	0.650	1.000	1.000	1.000	1.000	1.000	1.000
Boston	0.000	0.000	-1.000	0.000	0.000	0.000	0.000	0.000
Matthew	1.000	1.000	1.000	0.674	1.000	1.000	1.000	1.000
Sandy	1.000	1.000	1.000	1.000	0.875	1.000	1.000	1.000
Baltimore	1.000	1.000	1.000	1.000	1.000	0.538	1.000	1.000
Portland	1.000	1.000	1.000	1.000	1.000	1.000	0.436	1.000
Oshkosh	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.825

Table 15: $\nu = 0.15$

Results for Linear kernel and without PCA using Object of Interest Analysis pipeline								
	Austin	Berlin	Boston	Matthew	Sandy	Baltimore	Portland	Oshkosh
Austin	0.725	0.785	0.315	0.735	0.605	0.455	0.515	0.900
Berlin	0.255	0.850	0.275	0.660	0.550	0.390	0.405	0.275
Boston	0.255	0.690	0.650	0.685	0.560	0.385	0.390	0.860
Matthew	0.160	0.500	0.115	0.550	0.370	0.320	0.285	0.710
Sandy	0.175	0.395	0.165	0.435	0.375	0.375	0.325	0.550
Baltimore	0.240	0.685	0.235	0.660	0.565	0.325	0.410	0.855
Portland	0.205	0.660	0.220	0.655	0.525	0.395	0.700	0.860
Oshkosh	0.110	0.070	0.135	0.500	0.385	0.350	0.260	0.875

Table 16: $\nu = 0.05$

Results for RBF kernel and without PCA using FObject of Interest Analysis pipeline								
	Austin	Berlin	Boston	Matthew	Sandy	Baltimore	Portland	Oshkosh
Austin	0.550	0.835	0.345	0.860	0.695	0.465	0.605	0.920
Berlin	0.390	0.675	0.320	0.825	0.670	0.465	0.545	0.545
Boston	0.515	0.865	0.450	0.865	0.675	0.470	0.615	0.920
Matthew	0.365	0.710	0.245	0.150	0.615	0.425	0.475	0.860
Sandy	0.465	0.740	0.335	0.790	0.075	0.455	0.515	0.885
Baltimore	0.500	0.760	0.380	0.870	0.710	0.225	0.560	0.900
Portland	0.470	0.775	0.330	0.850	0.665	0.460	0.450	0.915
Oshkosh	0.425	0.380	0.335	0.800	0.650	0.460	0.480	0.475

Table 17: $\nu = 0.05$

Results for RBF kernel and with PCA using Object of Interest Analysis pipeline								
	Austin	Berlin	Boston	Matthew	Sandy	Baltimore	Portland	Oshkosh
Austin	0.000	0.990	0.925	0.950	0.845	0.635	0.900	0.970
Berlin	0.985	0.000	0.930	0.950	0.845	0.635	0.900	0.970
Boston	0.985	0.990	0.000	0.950	0.840	0.630	0.895	0.965
Matthew	0.985	0.990	0.930	0.000	0.850	0.640	0.900	0.970
Sandy	0.985	0.990	0.930	0.950	0.000	0.640	0.900	0.970
Baltimore	0.985	0.990	0.930	0.950	0.850	0.000	0.900	0.970
Portland	0.985	0.990	0.930	0.950	0.850	0.640	0.000	0.970
Oshkosh	0.985	0.990	0.930	0.950	0.850	0.640	0.900	0.000

Table 18: $\nu = 0.05$

Results for Linear kernel and with PCA using Object of Interest Analysis								
	Austin	Berlin	Boston	Matthew	Sandy	Baltimore	Portland	Oshkosh
Austin	0.700	0.800	0.365	0.740	0.625	0.460	0.540	0.915
Berlin	0.255	0.825	0.280	0.670	0.580	0.405	0.420	0.305
Boston	0.270	0.715	0.625	0.710	0.575	0.395	0.405	0.870
Matthew	0.165	0.530	0.120	0.500	0.375	0.335	0.305	0.725
Sandy	0.185	0.435	0.200	0.455	0.300	0.375	0.355	0.565
Baltimore	0.245	0.685	0.240	0.665	0.575	0.275	0.410	0.860
Portland	0.225	0.760	0.290	0.665	0.540	0.405	0.650	0.875
Oshkosh	0.110	0.095	0.150	0.520	0.395	0.350	0.270	0.850

Table 19: $\nu = 0.10$

Results for RBF kernel and without PCA using Object of Interest Analysis								
	Austin	Berlin	Boston	Matthew	Sandy	Baltimore	Portland	Oshkosh
Austin	0.475	0.840	0.375	0.865	0.700	0.480	0.610	0.915
Berlin	0.455	0.675	0.350	0.835	0.685	0.485	0.560	0.555
Boston	0.540	0.875	0.450	0.865	0.690	0.475	0.625	0.925
Matthew	0.370	0.710	0.260	0.150	0.615	0.425	0.475	0.860
Sandy	0.465	0.740	0.345	0.795	0.075	0.455	0.520	0.885
Baltimore	0.500	0.760	0.380	0.870	0.710	0.225	0.560	0.900
Portland	0.500	0.780	0.350	0.855	0.695	0.465	0.450	0.915
Oshkosh	0.425	0.385	0.335	0.805	0.650	0.460	0.480	0.475

Table 20: $\nu = 0.10$

Results for RBF kernel and with PCA using Object of Interest Analysis								
	Austin	Berlin	Boston	Matthew	Sandy	Baltimore	Portland	Oshkosh
Austin	0.000	0.990	0.925	0.950	0.845	0.635	0.900	0.970
Berlin	0.985	0.000	0.930	0.950	0.845	0.635	0.900	0.970
Boston	0.985	0.990	0.000	0.950	0.840	0.630	0.895	0.965
Matthew	0.985	0.990	0.930	0.000	0.850	0.640	0.900	0.970
Sandy	0.985	0.990	0.930	0.950	0.000	0.640	0.900	0.970
Baltimore	0.985	0.990	0.930	0.950	0.850	0.000	0.900	0.970
Portland	0.985	0.990	0.930	0.950	0.850	0.640	0.000	0.970
Oshkosh	0.985	0.990	0.930	0.950	0.850	0.640	0.900	0.000

Table 21: $\nu = 0.10$

Results for Linear kernel and without PCA using Object of Interest Analysis								
	Austin	Berlin	Boston	Matthew	Sandy	Baltimore	Portland	Oshkosh
Austin	0.625	0.870	0.420	0.775	0.640	0.480	0.555	0.925
Berlin	0.290	0.700	0.310	0.710	0.605	0.425	0.465	0.375
Boston	0.270	0.730	0.600	0.710	0.580	0.395	0.410	0.875
Matthew	0.175	0.575	0.145	0.500	0.380	0.340	0.325	0.740
Sandy	0.200	0.490	0.250	0.475	0.300	0.380	0.385	0.595
Baltimore	0.280	0.735	0.340	0.680	0.615	0.275	0.445	0.875
Portland	0.265	0.795	0.380	0.680	0.565	0.415	0.600	0.885
Oshkosh	0.135	0.115	0.160	0.535	0.405	0.360	0.285	0.775

Table 22: $\nu = 0.15$

Results for RBF kernel and without PCA using Object of Interest Analysis								
	Austin	Berlin	Boston	Matthew	Sandy	Baltimore	Portland	Oshkosh
Austin	0.425	0.845	0.410	0.865	0.710	0.485	0.615	0.915
Berlin	0.515	0.625	0.390	0.860	0.710	0.500	0.610	0.555
Boston	0.585	0.895	0.400	0.870	0.725	0.480	0.640	0.925
Matthew	0.380	0.715	0.270	0.150	0.620	0.450	0.485	0.865
Sandy	0.485	0.745	0.355	0.810	0.075	0.460	0.540	0.885
Baltimore	0.505	0.760	0.385	0.870	0.710	0.225	0.565	0.900
Portland	0.555	0.800	0.380	0.865	0.705	0.470	0.400	0.925
Oshkosh	0.445	0.390	0.340	0.810	0.650	0.465	0.485	0.475

Table 23: $\nu = 0.15$

Results for RBF kernel and with PCA using Object of Interest Analysis								
	Austin	Berlin	Boston	Matthew	Sandy	Baltimore	Portland	Oshkosh
Austin	0.025	0.990	0.925	0.950	0.845	0.635	0.900	0.970
Berlin	0.985	0.000	0.930	0.950	0.845	0.635	0.900	0.970
Boston	0.985	0.990	0.000	0.950	0.840	0.630	0.895	0.965
Matthew	0.985	0.990	0.930	0.000	0.850	0.640	0.900	0.970
Sandy	0.985	0.990	0.930	0.950	0.000	0.640	0.900	0.970
Baltimore	0.985	0.990	0.930	0.950	0.850	0.000	0.900	0.970
Portland	0.985	0.990	0.930	0.950	0.850	0.640	0.000	0.970
Oshkosh	0.985	0.990	0.930	0.950	0.850	0.640	0.900	0.000

Table 24: $\nu = 0.15$