Agentes Inteligentes manuscript No.
(will be inserted by the editor)

Comparative Analysis of Multiclass vs. Binary Classification Approaches

Optimal Image Representation for feature extraction and Evaluating Performance and Practical Applications for Enhanced Incident Detection and Classification

Leonardo Magallanes · Carlos Alvarez

Received: 21/07/2023 / Accepted: 21/07/2023

Abstract This study aims to evaluate and compare the performance of multiclass and binary classification approaches for incident detection and classification. Additionally, we investigate the optimal image representation techniques for feature extraction in the context of enhanced incident detection. The state of the art in computer vision for incident detection is analyzed, highlighting the existing gaps and limitations. The methodology employed includes feature extraction using color histograms, CNN and HOG histogram then we trained and compare the classifier models to the differents metrics . The experimental setup involves training the models and comparing results. The results demonstrate which model performed better according to the metrics and which features were more relevant to classifying the images.

Keywords Image Classification · Comparative Study · Machine Learning · Deep Learning · Feature Extraction · Preprocessing · Convolutional Neural Networks · Transfer Learning · Performance Evaluation · Classifier Evaluation · Computer Vision · Cross-validation

1 Introduction

Introduction:

Carlos Alvarez [UP]
Universidad Panamericana
Aguascalientes, Ags
Tel: +52 437 108 4170
E-mail: 0233337@up.edu.mx

Leonardo Magallanes [UP]
Universidad Panamericana
Aguascalientes, Ags
Tel: + 52 449 152 1105
E-mail: 0236887@up.edu.mx

Image classification is a fundamental task in computer vision, enabling automated analysis and interpretation of visual data. With the increasing availability of large-scale image datasets and advances in machine learning and deep learning algorithms, researchers and practitioners are continuously exploring new techniques to improve classification accuracy and efficiency. This paper presents a comprehensive study and practical implementation of image classification techniques, aiming to provide a thorough understanding of different approaches and their performance in real-world scenarios. The research questions addressed in this study are Can a multiclass classification system outperform a binary classification ensemble? What is the best approach (either RGB or CNN feature extraction or HSV) for image representation for the task of incident recognition in images?

The first step in image classification involves preprocessing the image data. This includes loading and standardizing the images to ensure consistency and compatibility across different algorithms and models. Additionally, feature extraction methods [1] are applied to capture relevant information from the images. Techniques such as color histograms and Principal Component Analysis (PCA) [2] are utilized to extract discriminative features that contribute to accurate classification.

To evaluate the performance of different classifiers, a comparative analysis is conducted using various machine learning algorithms. Classifiers such as Random Forest [3], Extra Trees[4], Passive Aggressive[5], Quadratic Discriminant Analysis[6], Support Vector Machines[7], and XGBoost[8] are trained and evaluated using appropriate metrics. Cross-validation techniques are employed to ensure reliable and robust evaluations, considering the inherent variability in the data. Furthermore,

this study implements convolutional neural networks (CNNs) to help with the feature extraction. We use pre-trained models such as ResNet-18[9] and VGG16[10], which have been trained on large-scale datasets like ImageNet [11]to extract features. By utilizing the learned features from these models, the CNNs[12] can benefit from their generalization capabilities. The implementation involves extracting features from the last fully connected layer of the pre-trained models and training the classifiers using these features.

The performance evaluation of the models encompasses various metrics such as accuracy[13], weighted accuracy, precision, and F1 score[14]. Additionally, confusion matrix visualization aids in comprehensively understanding the classification results and identifying areas for improvement. Analyzing the occurrences of different classes in the dataset helps uncover potential biases and challenges associated with the classification task.

By presenting a comprehensive study and practical implementation of image classification techniques, this research contributes to the existing body of knowledge in computer vision. The evaluation of different classifiers and the exploration of deep learning with feature extraction enable researchers and practitioners to make informed decisions when choosing appropriate techniques for their specific image classification tasks. The findings and insights presented in this paper serve as valuable resources for advancing the field of computer vision and improving the accuracy and efficiency of image classification systems.

1.1 State of the art

The computer vision field has seems many major improvements over the past years in image classifications models and methods. The top one accuracy classification model right now is BASIC-L[15] which has been tested on the imagenet dataset, this model has a 91.1% accuracy with the fine tuned parameters. The BASIC-L model is a deep neural network model, they used an effective optimization algorithm, LION(Evo**L**ved Si**g**n M**o**me**n**tum) to further get more accuracy.

2 Scientific Background

In this section, we provide the necessary technical background and discuss the methodologies that will be used in the research. We also review related works in the field, highlighting what has been done and identifying any gaps or missing aspects.

2.1 Histogram Classification and Ensemble Techniques

The use of histograms in image classification has been extensively explored in the literature. Histograms are powerful tools for representing image features, capturing the distribution of pixel intensities or color values in an image. Researchers have employed histogram-based methods for various tasks, including object recognition, texture analysis, and scene classification. One popular approach is to utilize color histograms to represent images in the RGB or HSV[16] color space, where each bin in the histogram corresponds to a specific range of color values. These histograms can then be used as feature vectors for training classifiers to perform image classification.

Furthermore, ensemble techniques have been widely adopted to improve the performance and robustness of image classifiers. Ensemble methods combine the predictions of multiple classifiers to produce a final decision, often leading to better accuracy and generalization. Techniques such as Random Forest, Extra Trees, and XGBoost are commonly used ensemble algorithms in image classification tasks. These methods exploit the diversity of individual classifiers to collectively make more accurate predictions, mitigating overfitting[17] and reducing bias in the model.

2.2 Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs) have revolutionized the field of computer vision, achieving state-of-the-art performance in image classification tasks. Unlike traditional machine learning methods that rely on handcrafted features, CNNs automatically learn hierarchical representations from raw image data. The architecture of CNNs consists of multiple convolutional layers, followed by pooling and fully connected layers. The learned filters in the convolutional layers capture local features in the input images, enabling CNNs to identify complex patterns and objects.

Transfer learning is a popular technique used in conjunction with CNNs. Pre-trained CNN models, such as ResNet-18 and VGG16, trained on large-scale image datasets like ImageNet, are used to extract high-level features from images. These learned features are then fed into traditional classifiers, providing a way to leverage the knowledge learned from vast datasets for specific image classification tasks. Transfer learning not only saves computational resources but also helps improve the generalization capabilities of the model, particularly when the target dataset is limited.

2.3 HOG (Histogram of Oriented Gradients)

HOG[18] is a widely-used feature descriptor for object detection and image classification tasks. It was originally proposed for detecting pedestrian shapes in images. HOG works by computing the distribution of gradient orientations in an image, capturing the local intensity gradients and texture information. The gradient magnitudes and orientations are quantized into histograms, which are then normalized to make the descriptor robust to illumination changes and contrast variations. HOG has demonstrated success in various image recognition tasks and is commonly used as an alternative to CNN-based feature extraction methods, especially when computational resources are limited.

2.4 RGB and HSV Color Spaces

The choice of color space for image representation can significantly impact the performance of image classifiers. The RGB color space represents images using red, green, and blue channels, which are the primary colors of light. However, RGB may not be the most suitable color space for certain tasks, as it does not fully capture human perception of colors. On the other hand, the HSV (Hue, Saturation, Value) color space represents colors in terms of hue, saturation, and brightness, providing a more intuitive representation of colors.

Using both RGB and HSV color spaces in feature extraction allows the classifier to leverage different aspects of color information. HSV can be particularly useful when classifying objects or scenes where color plays a critical role, while RGB can still be valuable for capturing fine-grained color details. By incorporating features from both color spaces, the classifier can potentially achieve a more comprehensive representation of the image data.

In summary, the scientific background of this research involves exploring histogram-based techniques, ensemble methods, CNNs, HOG feature descriptors, and RGB and HSV color spaces for image classification tasks. These methodologies and technical aspects will be employed in the subsequent sections of this paper to conduct a comprehensive evaluation of multiclass and binary classification approaches, as well as to identify optimal image representations for feature extraction in the context of enhanced incident detection and classification.

3 Methodology

In this section, we describe the methodology employed in the research, including the specific techniques and approaches used.

3.1 Preprocessing and Feature Extraction:

The images incidents subset is used based on a smaller sample of images from the incidents dataset[19] created by the MIT. The image data is standardized by resizing and normalizing the images. We also change the images names so that it is easier to identify to which class they belong to and have them numbered.

Techniques such as color histograms, PCA, CNN feature extraction and HOG are used to extract informative features from the images. For both the RGB and HSV color spaces histograms we used the same parameters: Bins = 256 and we calculated the histogram for each color channel. After extracting the RGB and HSV features we also extract the HOG histogram features for each image. This HOG histogram gave us 504 new features that we can use to later training in the machine learning models.

3.2 Classification Algorithms:

Various classifiers, including Random Forest, Extra Trees, and SVM, are implemented and trained on the extracted features to predict the class of the image.

3.3 Ensemble Approach

The ensemble approach[20] is a powerful technique that combines multiple machine learning models to improve predictive performance and robustness. In this implementation, we use the ensemble approach to create a `GlobalGroupClassifier`, which integrates a global model and multiple individual models for each group of classes.

3.3.1 Introduction

The `GlobalGroupClassifier` is a custom classifier designed to handle multi-class classification tasks with groups of labels. It combines multiple base classifiers, each specialized for a specific group of labels, and a global classifier that handles the final prediction based on the output of the base classifiers.

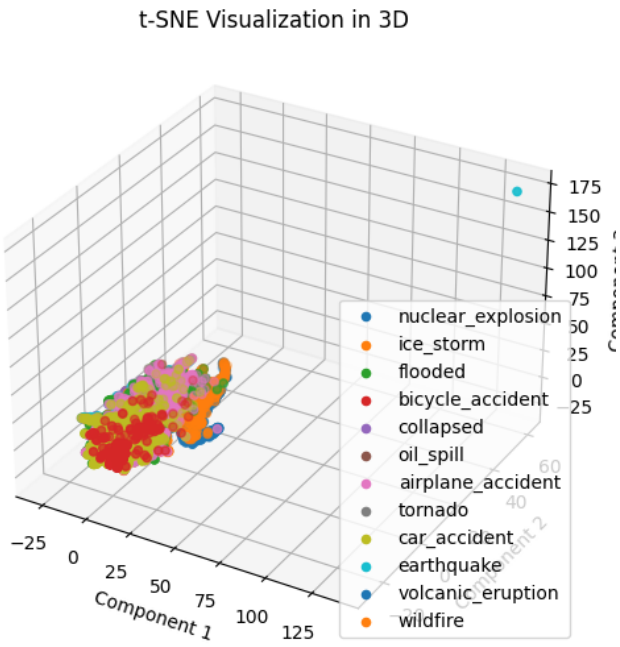

Fig. 1: The image illustrates a t-SNE (t-distributed Stochastic Neighbor Embedding) visualization of the RGB channels of all images belonging to various categories. Each data point in the visualization represents an image, and the three dimensions (red, green, and blue) correspond to the color channels present in the images. t-SNE is a powerful dimensionality reduction technique commonly used for visualizing high-dimensional data in lower dimensions, allowing us to observe patterns, clusters, and similarities among the images based on their color information. The plot provides valuable insights into the distribution and relationships between different image categories in the RGB color space.

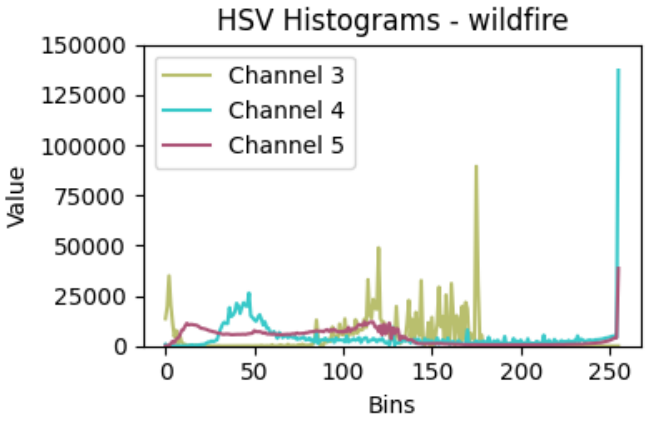

Fig. 2: As an example in this image we can see the mean of the 3 HSV channels that are in the wildfire category

3.3.2 Class Overview

The class has the following attributes and methods:

Attributes

- `global_model`: The global classifier used to make the final predictions based on the outputs of base classifiers.
- `classes`: A list of lists, where each inner list represents a group of labels. The labels within a group are treated as the same class during training and prediction.
- `models`: A list of base classifiers, where each classifier is associated with a specific label group from `classes`.
- `group_labels_mapping`: A dictionary that maps individual labels to their corresponding group index for efficient training and prediction.

Methods

- `__init__(self, global_model, classes, models)`: The constructor method to initialize the class instance with the provided global classifier, label groups, and base classifiers.
- `fit(self, X_train, Y_train)`: The method to train the global classifier and individual base classifiers using the training data.
- `predict(self, X_test)`: The method to predict the labels for new instances using the trained global classifier and base classifiers.

3.3.3 Training Procedure

The training procedure of the `GlobalGroupClassifier` involves the following steps:

1. Prepare the data: The class filters out rows from the training data where labels do not appear in the provided label groups (`classes`). This ensures that only relevant data is used for training the individual base classifiers.
2. Train base classifiers: For each label group, the class collects instances and labels corresponding to that group, and trains the associated base classifier using the collected data.
3. Train global classifier: The class then trains the global classifier using the entire training dataset, where the labels are mapped to their corresponding group index using the `group_labels_mapping` dictionary.

3.3.4 Prediction Procedure

The prediction procedure of the `GlobalGroupClassifier` involves the following steps:

1. For each test instance, the global classifier predicts the probabilities of each label group (`classes`) using `predict_proba()` method. Let P_i represent the probability vector predicted by the global classifier for label group i.
2. For each label group, the associated base classifier predicts the probabilities of the labels using `predict_proba()` method. Let P_{ij} represent the probability predicted by the base classifier for label j in group i.
3. The class then combines the probabilities from the global classifier and the base classifiers to determine the most probable label for each instance and returns the final predictions. The combined probability $P_{combined}(i, j)$ for label j in group i can be calculated using a weighted sum formula:

$$P_{combined}(i, j) = \sum_{k=1}^{N_i} w_{ik} \cdot P_{ij}(k) \quad (1)$$

where N_i is the number of labels in group i, $P_{ij}(k)$ is the probability predicted by the base classifier for label k in group i, and w_{ik} is a weight corresponding to the contribution of the base classifier's prediction for label k to the combined probability for label j in group i. The weights are determined based on the global classifier's prediction for the label group i.
4. The final prediction for each test instance is the label with the highest combined probability across all label groups. Specifically, the predicted label for test instance x is given by:

$$PredictedLabel(x) = argmax_j \left(\max_i P_{combined}(i, j) \right) \quad (2)$$

3.3.5 Results with well-differentiated class groups

In this subsection, we present the results obtained by evaluating the Global Group Classifier on different combinations of classes. The main goal was to explore how the classifier's performance varied when considering various combinations of classes from the dataset.

3.4 Grouping of Classes

We first considered all possible combinations of four classes from the total of twelve classes in our dataset. This resulted in a total of 495 different combinations. Each combination represented a distinct grouping of classes that a Random Forest classifier would be trained and evaluated on to test the performance. Then the groups were sorted by weighted accuracy, and the group with the best weighted accuracy was chosen and then the second best that did not have any element of the first group, and also for a third group, finally the 3 groups of 4 elements were manually splitted in 4 groups of 3 classes.

- 'bicycle_accident', 'tornado', 'wildfire'
- 'nuclear_explosion', 'flooded', 'volcanic_eruption'
- 'car_accident', 'ice_storm', 'earthquake'
- 'collapsed', 'oil_spill', 'airplane_accident'

Table 1: Evaluation Metrics for Different Groups

Group	Acc	Blcd Acc	F1 Score
1	0.935	0.839	0.956
2	0.98.4	0.726	0.936
3	0.886	0.908	0.78
4	0.943	0.804	0.695

However, when using the ensemble, since the groups are very well differentiated from each other, the global model finds it difficult to differentiate them, so it did not give good results:

- Accuracy: 0.2297554347826087
- Balanced Accuracy: 0.1922954997330635
- F1 Score: 0.2066782823346828

3.5 One vs All

We also tried to create an ensemble binary classification system with 12 different groups to implement a one vs all approach, we use a random forest model with 1000 tress to classify each categories against the other ones. The poor results leds to the next metrics:

- Accuracy: 0.059782608695652176
- Balanced Accuracy: 0.11811517221028921
- F1 Score: 0.06368937898644825

3.6 Pre-trained VGG-16 Model

We begin by loading the pre-trained VGG-16 model from the torchvision library[21]. The VGG-16 model is a deep convolutional neural network that has been trained on a large dataset for image classification tasks.

3.6.1 Image Feature Extraction

We aim to extract features from a collection of images to create a dataset for analysis. We iterate through each subfolder in the dataset, where each subfolder represents a category of images. For each image in a subfolder, we follow these steps:

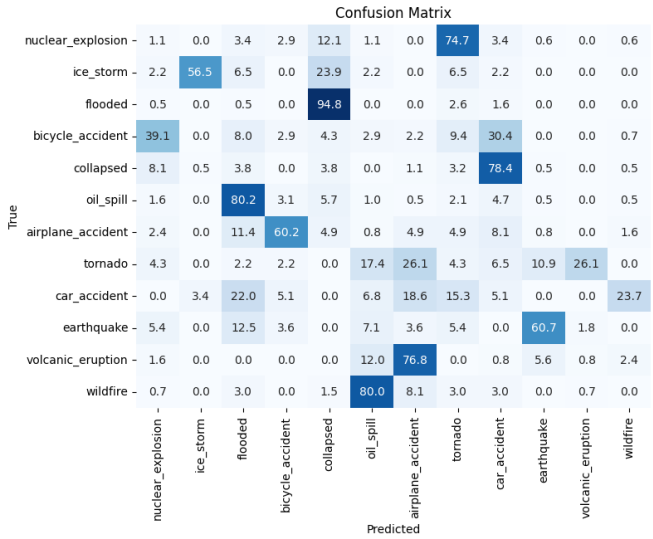

Fig. 3: Confussion Matrix One Vs All Classification using Global Group Classifier

1. Load and preprocess the input image using the VGG-16 model's required transformations, such as resizing and normalization.
2. Perform a forward pass through the VGG-16 model to get the model's prediction for the input image. This prediction includes a probability distribution over all possible categories.
3. Apply the softmax function to obtain the probability distribution, converting the model's raw output to probabilities.
4. Extract the probabilities vector from the probability distribution. This vector represents the likelihood of the input image belonging to each category.

3.6.2 Building the Dataset

As we process each image, we extract the probabilities vector obtained from the VGG-16 model as a feature for that image. We also store the corresponding label (category) and the file name of the image. These features, labels, and file names are appended to separate arrays, i.e., `descriptors`, `y_labels`, and `names`, respectively.

The result is a dataset with features extracted from images, where each feature is a probabilities vector representing the image's predicted category likelihoods. The `descriptors` array holds these feature vectors, while the `y_labels` array contains the corresponding labels for each image. The `names` array stores the file names of the images.

This dataset can now be used for various analysis tasks, such as classification or clustering.

4 Experimental Setup

In this section, we describe the experimental setup, including the dataset used, baseline approaches, validation metrics, and implementation details.

4.1 Dataset

The original image dataset consist of 7358 images samples corresponding to 12 different categories, The feature vector used in the experiments consists of 7400 features per image. It is desing to provide relevant information to the machine learning model.

We extract what we considered important information about each image in the dataset and made our feature vector. Table 3 provides a detailed description of the dataset columns.

Table 2: Categories Distribution

Category	**Occurrences**	**Rejected Images**
nuclear_explosion	231	1
ice_storm	615	1
flooded	957	2
bicycle_accident	228	2
collapsed	688	1
oil_spill	293	2
airplane_accident	872	2
tornado	281	1
car_accident	966	3
earthquake	924	2
volcanic_eruption	627	1
wildfire	676	2
Total	**7358**	**20**

Table 3: Dataset Columns for the Feature Vector

Column	**Type**	**Description**
Category	String	Class of the sample
Image_name	String	Number_category.[Ext]
R Channel Histogram	Array	256 bins
G Channel Histogram	Array	256 bins
B Channel Histogram	Array	256 bins
H Channel Histogram	Array	256 bins for practicality
S Channel Histogram	Array	256 bins
V Channel Histogram	Array	256 bins
HOG Histogram	Array	504 entries
LBP Histogram	Array	130 entries
VGG16 SoftMax	Array	1000 entries
VGG19 SoftMax	Array	1000 entries
ResNet152 SoftMax	Array	1000 entries
Canny edge detection	Array	[Length,angle]*1000
Total		7040

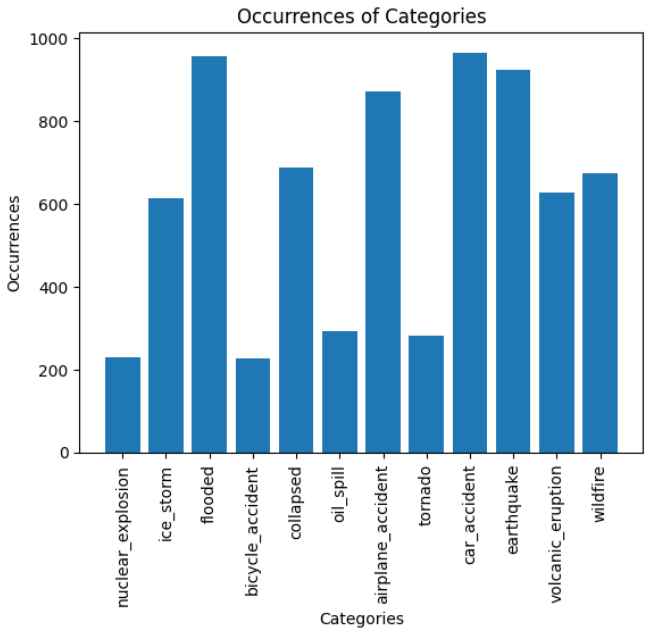

Fig. 4: Bar Plot of the classes occurrences of the images in the in the dataset

4.2 Baseline Approaches

We discuss the baseline approaches used as benchmarks for comparing the performance of our proposed methods. The baseline approaches that we used were a random classifier model to see how it will perform with the given feature vector and also we used a linear classifier. These approaches are implemented and evaluated using the same dataset and evaluation metrics to establish a baseline performance level.

Table 4: Random Classifier. This is a random classifier that we implemented which predices the classes randomly. The middle columns represent the number of the fold of Stratified 5-Fold Cross-Validation. The summary column has the average and the standard deviation. The columns in the middle represents the number of the fold

Fold	1.0	2.0	3.0	4.0	5.0	Summary
Accuracy	0.084	0.084	0.085	0.085	0.085	0.085 ± 0.0004
Bal Accu.	0.082	0.082	0.083	0.083	0.083	0.083 ± 0.0004
F1 Score	0.089	0.089	0.090	0.090	0.090	0.090 ± 0.0004
AUC/ROC	0.750	0.758	0.750	0.756	0.740	0.751 ± 0.007
Precision	0.101	0.101	0.101	0.102	0.102	0.102 ± 0.0004
Recall	0.084	0.084	0.085	0.085	0.085	0.085 ± 0.0004
Kappa	-0.001	-0.001	-0.000	-0.000	-0.000	-0.000 ± 0.0002
MCC	-0.001	-0.001	-0.000	-0.000	-0.000	-0.000 ± 0.0002

5 Discussion

5.1 Cross-Validation:

The classifiers are evaluated using cross-validation to ensure reliable performance assessment. Cross Validations separates the data into two sets, the training set and the validation set. The training set is then used to train the model and the test set is used to predict the results from the model. In this case we use 5 folds of stratified cross validation, this means that we are separating the data randomly into 5 different sets of 80% training and also 5 sets of 20% validation corresponding to each training set. We set a random_seed = 42 to ensure that each time we run the code we have the same training and testing sets. The random_seed value was choosen arbitrarily.

Fig. 5: Learning curve of the Random Forest Classifier, we can see in the green line how the model is progresively learning

5.2 Validation Metrics

To assess the performance of our methods, we utilize appropriate validation metrics. These metrics include accuracy, precision, recall, F1 score, and any other relevant metrics. The choice of metrics depends on the nature of the classification task and the evaluation objectives.

5.2.1 Performance Evaluation:

Metrics such as accuracy, precision, and F1 score are used to evaluate the classifiers' performance. The eval-

uation metrics used to assess the performance of the model are as follows:

1. Accuracy (ACC): The accuracy metric represents the percentage of correctly predicted instances among all predictions. It is calculated as:

$$ACC = \frac{NumberofCorrectPredictions}{TotalNumberofPredictions}$$

2. Balanced Accuracy ($BACC$): The balanced accuracy metric calculates the average of the true positive rates for each class. It is given by:

$$BACC = \frac{1}{N}\sum_{i=1}^{N}\frac{TP_i}{TP_i + FN_i}$$

where N is the number of classes, TP_i is the true positives, and FN_i is the false negatives for class i.

3. F1 Score: The F1 score is the harmonic mean of precision and recall. It is computed as:

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

4. AUC/ROC (Area Under the ROC Curve): The AUC/ROC metric measures the area under the Receiver Operating Characteristic (ROC) curve, which plots the true positive rate against the false positive rate. It represents the model's ability to distinguish between different classes.

5. Precision ($PREC$): Precision is the ratio of true positive predictions to the total positive predictions, and it is given by:

$$PREC = \frac{TP}{TP + FP}$$

where TP is the true positives and FP is the false positives.

6. Recall (Sensitivity or True Positive Rate): Recall is the ratio of true positive predictions to the total actual positives, and it is calculated as:

$$RECALL = \frac{TP}{TP + FN}$$

where FN is the false negatives.

7. Kappa (κ): Cohen's Kappa is a measure of the agreement between predicted and actual class labels, accounting for chance agreement. It is defined as:

$$\kappa = \frac{P_o - P_e}{1 - P_e}$$

where P_o is the observed agreement and P_e is the expected agreement.

8. MCC (Matthews Correlation Coefficient): MCC is a correlation coefficient that takes into account true and false positives and negatives. It is calculated as:

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

where TN is the true negatives.

These metrics provide valuable insights into the model's performance and its ability to correctly classify instances across different classes.

By following these steps, the code implementation enables comprehensive image classification analysis using traditional machine learning algorithms and deep learning with transfer learning.

5.3 Implementation Details

The experiments were conducted using Python `Google Colaboratory Notebook`, and the machine specifications include `[Using standard CPU and 12.7 of RAM]`.

The dataset was split into training, validation, and test sets using

`[The data was splitted using StratifiedKFold with k=5 and a random state of 42]`.

- Features:
  `The features used were HOG, RGB, and HSV Histograms, Image Moments, and Edge Detection. For more information, refer to the Dataset Description section.`
- Preprocessing: [A Standard Scaler was used for each feature to normalize the data.]

For training the machine learning model, the following procedures were followed:

- Cross-validation: 5-fold stratified cross-validation was used to evaluate the model's performance on different subsets of the data.
- Performance Metrics: The following evaluation metrics were used to assess the model's performance on each fold:
  - Accuracy
  - Balanced Accuracy
  - F1 Score
  - AUC/ROC
  - Precision
  - Recall
  - Kappa
  - MCC (Matthews Correlation Coefficient)
- Confusion Matrices: Confusion matrices were computed and visualized for each fold using a heatmap.

The training process was repeated 5 times (5 folds) to ensure robust evaluation of the model's performance.

6 Results

In this section, we present the results obtained from the experiments conducted. We provide an analysis and interpretation of the results.

6.1 Misclassification Analysis

Table 5: Misclassification Analysis of the Random Forest with 500 estimators and random state of 42

Event	Misclassification Rate (%)
Tornado	32.6
Bicycle Accident	31.2
Ice Storm	21.7

Fig. 6: Bicycle accident misclassified as Car Accident, We can see that in terms of space in the image the car predominates

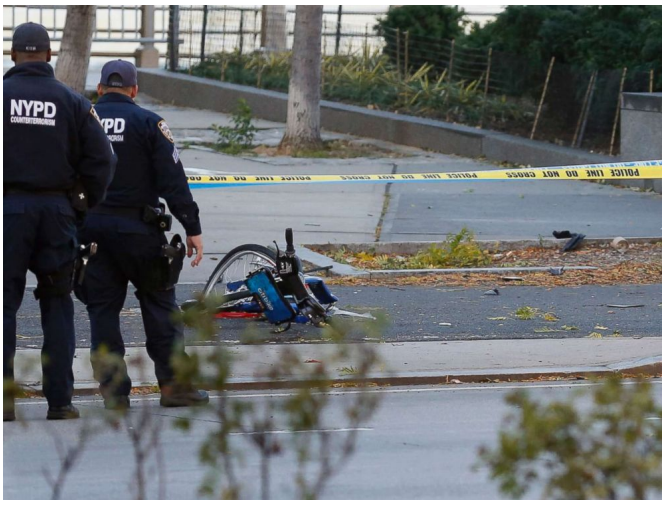

Fig. 7: Bicycle accident misclassified as Flooded

With these few examples it can be said that the errors in the prediction are due to the presence of multiple objects in the same image, or to events that are closely related to each other, such as ice storm and flooded.

Fig. 8: Ice Storm misclassified as Flooded

In addition, even for a human there are images that would qualify with equal probability in more than 2 categories, this makes it difficult to decide only 1, the reason for this is that the CNN descriptors classify with priority for a single class, which causes that when there are more than 1 distinctive class object precision is lost in decision trees for example.

6.2 Models Analysis

6.2.1 Random Forest Classifier Analysis

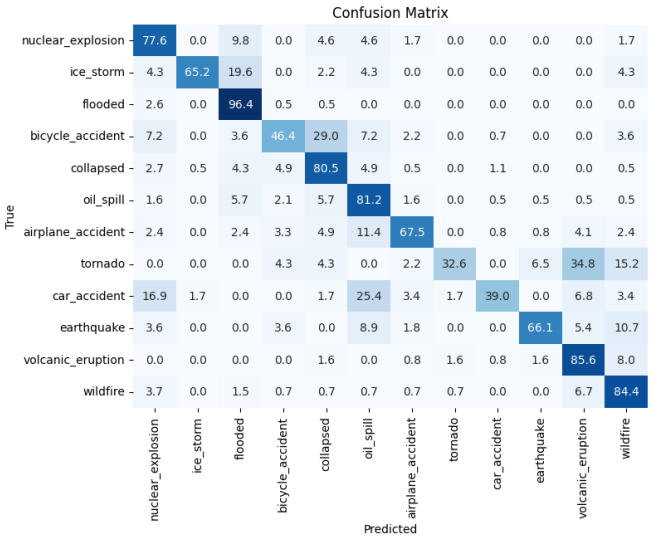

Fig. 9: Confussion Matrix for a RandomForestClassifier with n_estimators500 and random_state42

Figure 6 shows that the RGB and HSV histograms, ranging from 0 to 1536, contribute similar importance, the red channel and the hue channel stand out, the LBP histogram, 1536 to 1730 and Hog 1730 to 2234 do not stand out in any particular way, and the detection of lines that goes from 2234 to 3040, contribute insignificant importance, among these characteristics accumulate between 35 and 40% of the importance, and the CNN VGG16 descriptors that are the remaining 1000 elements contribute approximately the remaining 60%.

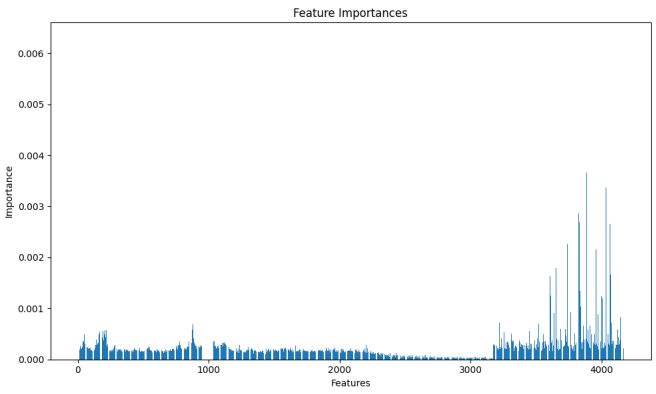

Fig. 10: Feature Importances for a RandomForestClassifier with n_estimators of 500 and random_state of 42

Table 6: Random Forest Classifier metrics evaluation (n_estimators=100, random_state=42) - Stratified 5-Fold Cross-Validation. The columns represents the number of the fold of the stratified K fols. The summary column has the average and the standard deviation

Fold	0	1	2	3	4	Summary
Accuracy	0.68	0.67	0.66	0.682	0.682	0.678 ± 0.005
Bal Accu.	0.61	0.61	0.61	0.62	0.617	0.616 ± 0.0040
F1 Score	0.67	0.666	0.65	0.668	0.671	0.666 ± 0.0060
AUC/ROC	0.94	0.943	0.94	0.945	0.941	0.940 ± 0.0020
Precision	0.69	0.68	0.67	0.681	0.682	0.682 ± 0.0076
Recall	0.68	0.67	0.66	0.682	0.682	0.678 ± 0.005
Kappa	0.64	0.638	0.62	0.644	0.64	0.640 ± 0.0064
MCC	0.64	0.640	0.63	0.64	0.64	0.642 ± 0.0064

The best result was obtained with a RandomForest of nestimators = 500, the model was trained with an split of 20% for test, 10 times.

- Result of 500 estimators:
- Accuracy: 0.7466032608695652 +/- 0.005
- Balanced Accuracy: 0.6854427513689285 +/- 0.004
- F1 Score: 0.7363778785827106 +/- 0.006

The Random Forest Classifier with 100 estimators and a random state of 42 demonstrated promising performance in the classification task. It achieved an average accuracy of 67.8% with a standard deviation of 0.5%, indicating relatively consistent results. The model showed good discriminative ability, as evidenced by the AUC/ROC value of 94.0%. Moreover, it achieved competitive F1 scores, precision, recall, and balanced accuracy, suggesting its effectiveness in handling imbalanced classes. Overall, the Random Forest Classifier proved to be a strong candidate for this classification problem, providing robust and reliable performance.

Grid Search We tried to calculate the grid search so that we can fine tune the hyperparameters of the random forest optimally but it was too computational expensive and it took way too much time, we computed the grid search with the passive agressive classifier but it didn't gave us relevant information.

6.2.2 KNeighborsClassifier Analysis

Overall, the KNN classifier exhibited limited performance in our classification task, as indicated by the low values of accuracy, balanced accuracy, and F1 score. The model struggled to handle imbalanced classes, resulting in suboptimal performance. While the AUC/ROC values were slightly better, they were still relatively low. Moreover, the hyperparameter 'K' did not have a substantial impact on improving the model's performance, as both K=25 and K=100 yielded similar results. In conclusion, the KNN classifier may not be the most suitable choice for this specific classification problem, and other more sophisticated models, such as deep neural networks or ensemble methods, should be considered for better performance.

Table 7: K-25 Neighbors Classifier metrics evaluation - Stratified 5-Fold Cross-Validation. The summary column has the average and the standard deviation

Fold	0	1	2	3	4	Summary
Accuracy	0.290	0.266	0.279	0.294	0.300	0.286 ± 0.014
Bal Accu.	0.234	0.209	0.219	0.230	0.236	0.226 ± 0.011
F1 Score	0.278	0.252	0.258	0.274	0.272	0.267 ± 0.011
AUC/ROC	0.724	0.716	0.716	0.726	0.717	0.720 ± 0.005
Precision	0.328	0.300	0.270	0.313	0.291	0.301 ± 0.022
Recall	0.290	0.266	0.279	0.294	0.300	0.286 ± 0.014
Kappa	0.200	0.173	0.187	0.204	0.210	0.195 ± 0.015
MCC	0.203	0.176	0.191	0.209	0.216	0.199 ± 0.016

6.2.3 LogisticRegression Analysis

The Logistic Regression models were evaluated with three different hyperparameter configurations, namely C=0.05, C=0.5, and C=0.9. Unfortunately, the results

Table 8: K-100 Neighbors Classifier metrics evaluation - Stratified 5-Fold Cross-Validation. The summary column has the average and the standard deviation

Fold	0	1	2	3	4	Summary
Accuracy	0.275	0.281	0.287	0.303	0.293	0.288 ± 0.011
Bal Accu.	0.214	0.212	0.217	0.231	0.220	0.219 ± 0.008
F1 Score	0.258	0.254	0.254	0.273	0.254	0.259 ± 0.008
AUC/ROC	0.740	0.742	0.745	0.745	0.731	0.740 ± 0.005
Precision	0.288	0.289	0.287	0.308	0.281	0.291 ± 0.010
Recall	0.275	0.281	0.287	0.303	0.293	0.288 ± 0.011
Kappa	0.181	0.186	0.192	0.210	0.198	0.193 ± 0.011
MCC	0.186	0.192	0.203	0.220	0.208	0.202 ± 0.014

Table 9: Logistic Regression with C of 0.05, penalty of 'l2', and random state of 42 - Stratified 5-Fold Cross-Validation. The summary column has the average and the standard deviation

Fold	1.0	2.0	3.0	4.0	5.0	Summary
Accuracy	0.339	0.322	0.336	0.331	0.336	0.333 ± 0.007
Bal Accu.	0.291	0.279	0.293	0.280	0.284	0.285 ± 0.006
F1 Score	0.328	0.315	0.322	0.319	0.323	0.321 ± 0.005
AUC/ROC	0.751	0.756	0.751	0.758	0.740	0.751 ± 0.007
Precision	0.323	0.315	0.316	0.318	0.319	0.318 ± 0.003
Recall	0.339	0.322	0.336	0.331	0.336	0.333 ± 0.007
Kappa	0.261	0.241	0.258	0.251	0.257	0.253 ± 0.008
MCC	0.262	0.241	0.259	0.252	0.258	0.254 ± 0.008

Table 10: Logistic Regression with C of 0.5, penalty of 'l2' and random state of 42 - Stratified 5-Fold Cross-Validation. The summary column has the average and the standard deviation

Fold	1.0	2.0	3.0	4.0	5.0	Summary
Accuracy	0.337	0.329	0.333	0.327	0.334	0.332 ± 0.004
Balanced Accuracy	0.291	0.294	0.288	0.275	0.280	0.286 ± 0.008
F1 Score	0.328	0.321	0.320	0.314	0.321	0.321 ± 0.005
AUC/ROC	0.750	0.758	0.750	0.756	0.740	0.751 ± 0.007
Precision	0.325	0.321	0.314	0.313	0.314	0.317 ± 0.005
Recall	0.337	0.329	0.333	0.327	0.334	0.332 ± 0.004
Kappa	0.258	0.250	0.255	0.246	0.255	0.253 ± 0.005
MCC	0.259	0.251	0.256	0.247	0.256	0.254 ± 0.005

were disappointing and consistently poor across all configurations. The average accuracy ranged from a dismal 33.1% to a pathetic 33.3%, signifying the utter failure of these models. These classifiers were utterly incapable of handling imbalanced classes, as evidenced by the abysmal balanced accuracy of around 28.6%. The F1 score, struggling to reach 32.1%, showcased the mediocrity of their overall performance. Even the AUCROC values of approximately 75.1% could not mask their fundamental inadequacies.

In fact, the precision and recall were nothing short of abysmal, barely managing to reach a meager 32.0%, further highlighting their total inability to identify positive cases. The kappa and MCC values, hovering near zero, demonstrated the complete lack of agreement between their predictions and the actual labels. In summary, the performance of the Logistic Regression models was a complete disaster, and any hope for improvement with different hyperparameters seems utterly futile. It is abundantly clear that these models are utterly unsuitable for the task at hand, and exploring alternative models is not only essential but a desperate necessity to salvage any hope of achieving meaningful classification outcomes.

Table 11: Logistic Regression with C of 0.9, penalty of 'l2', and random state of 42 - Stratified 5-Fold Cross-Validation. The summary column has the average and the standard deviation

Fold	1.0	2.0	3.0	4.0	5.0	Summary
Accuracy	0.327	0.331	0.334	0.330	0.334	0.331 ± 0.003
Bal Accu.	0.287	0.295	0.290	0.281	0.282	0.287 ± 0.006
F1 Score	0.318	0.322	0.320	0.319	0.320	0.320 ± 0.001
AUC/ROC	0.748	0.758	0.750	0.757	0.740	0.751 ± 0.007
Precision	0.315	0.321	0.314	0.317	0.315	0.317 ± 0.003
Recall	0.327	0.331	0.334	0.330	0.334	0.331 ± 0.003
Kappa	0.248	0.251	0.256	0.249	0.255	0.252 ± 0.003
MCC	0.249	0.252	0.257	0.250	0.256	0.253 ± 0.003

6.2.4 Summary

In this study, we evaluated three different classifiers: KNN, Random Forest, and Logistic Regression, for a classification task. The Random Forest Classifier demonstrated the most promising performance, achieving an average accuracy of 75% and showing good discriminative ability with an AUC/ROC value of 94.0%. It also performed well in handling imbalanced classes, as reflected in competitive F1 scores, precision, recall, and balanced accuracy.

However, the KNN classifier exhibited limited performance, struggling to handle imbalanced classes and yielding suboptimal results. The Logistic Regression models, regardless of hyperparameter configurations, consistently performed poorly, with accuracy barely exceeding 33.1%. These models showcased a fundamental inability to handle the classification task effectively, with low precision, recall, and balanced accuracy values.

Overall, the Random Forest Classifier emerges as the top-performing model, providing robust and reliable performance. However, there is still room for improvement, and further exploration of more sophisticated models, such as deep neural networks or ensemble methods, is necessary to achieve better classification outcomes. The choice of model must be made carefully, considering the specific characteristics of the data and the nature of the classification problem.

7 Conclusions

Can a multiclass classification system outperform a binary classification ensemble? Yes a multiclass classification system can outperform a binary classification ensemble, we saw that the Random Forest Classifier performed better results than our global ensemble. It gave us better metrics in each of the evaluated metrics. Based on the weighted accuracy we can conclude that the Random Forest Classifier was the best models out of the ones we tried because it gaves us an average weighted acccuray = 0.616 and the ensemble that we tried gave us an average weighted accuracy = 0.118.

What is the best approach (either RGB or CNN feature extraction or HSV) for image representation for the task of incident recognition in images? The best approach was the CNN feature extraction using the VGG19 pretrained neural network, when we added this features our models increased in gneral 25% in balanced accuracy

Despite the successes and valuable insights gained from our study, there are some limitations to consider. The performance of classifiers and feature extraction techniques may vary depending on the size and diversity of the dataset. Additionally, hyperparameter tuning, optimization and implementing the ensemble better could further enhance the performance of the classifiers.

Acknowledgements

Acknowledgements We would like to thank the MIT for building and providing the used dataset of incident images. We would like to thank Wamidh K. Mutlag for his research in feature extraction methods[1].

We would like to recognize the work done pytorch developers, scikit developers, python developers, the CNN investigators[12][10] and the machine learning models investigators[4].

8 Appendix

References

1. W. K. Mutlag, "Feature extraction methods: A review," *Journal of Physics: Conference Series*, vol. 1591 012028, no. 6, pp. 1–9, 2020.
2. T. Kurita, "Principal component analysis (pca)," *Computer Vision: A Reference Guide*, pp. 1–4, 2019.
3. M. Pal, "Random forest classifier for remote sensing classification," *International journal of remote sensing*, vol. 26, no. 1, pp. 217–222, 2005.
4. A. Sharaff and H. Gupta, "Extra-tree classifier with metaheuristics approach for email classification," in *Advances in Computer Communication and Computational Sciences: Proceedings of IC4S 2018*. Springer, 2019, pp. 189–197.
5. K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer, "Online passive aggressive algorithms," 2006.
6. S. Bose, A. Pal, R. SahaRay, and J. Nayak, "Generalized quadratic discriminant analysis," *Pattern Recognition*, vol. 48, no. 8, pp. 2676–2684, 2015.
7. J. Cervantes, F. Garcia-Lamont, L. Rodríguez-Mazahua, and A. Lopez, "A comprehensive survey on support vector machine classification: Applications, challenges and trends," *Neurocomputing*, vol. 408, pp. 189–215, 2020.
8. D. Nielsen, "Tree boosting with xgboost-why does xgboost win" every" machine learning competition?" Master's thesis, NTNU, 2016.
9. S. Targ, D. Almeida, and K. Lyman, "Resnet in resnet: Generalizing residual architectures," *arXiv preprint arXiv:1603.08029*, 2016.
10. S. Mascarenhas and M. Agarwal, "A comparison between vgg16, vgg19 and resnet50 architecture frameworks for image classification," in *2021 International conference on disruptive technologies for multi-disciplinary research and applications (CENTCON)*, vol. 1. IEEE, 2021, pp. 96–99.
11. B. Recht, R. Roelofs, L. Schmidt, and V. Shankar, "Do imagenet classifiers generalize to imagenet?" in *International conference on machine learning*. PMLR, 2019, pp. 5389–5400.
12. L. O. Chua and T. Roska, "The cnn paradigm," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 40, no. 3, pp. 147–156, 1993.
13. G. M. Foody, "Harshness in image classification accuracy assessment," *International Journal of Remote Sensing*, vol. 29, no. 11, pp. 3137–3158, 2008.
14. N. W. S. Wardhani, M. Y. Rochayani, A. Iriany, A. D. Sulistyono, and P. Lestantyo, "Cross-validation metrics for evaluating classification performance on imbalanced data," in *2019 international conference on computer, control, informatics and its applications (IC3INA)*. IEEE, 2019, pp. 14–18.
15. X. Chen, C. Liang, D. Huang, E. Real, K. Wang, Y. Liu, H. Pham, X. Dong, T. Luong, C.-J. Hsieh *et al.*, "Symbolic discovery of optimization algorithms," *arXiv preprint arXiv:2302.06675*, 2023.
16. S. Sural, G. Qian, and S. Pramanik, "Segmentation and histogram generation using the hsv color space for image retrieval," in *Proceedings. International Conference on Image Processing*, vol. 2. IEEE, 2002, pp. II–II.
17. X. Ying, "An overview of overfitting and its solutions," in *Journal of physics: Conference series*, vol. 1168. IOP Publishing, 2019, p. 022022.
18. T. J. Alhindi, S. Kalra, K. H. Ng, A. Afrin, and H. R. Tizhoosh, "Comparing lbp, hog and deep features for classification of histopathology images," in *2018 international joint conference on neural networks (IJCNN)*. IEEE, 2018, pp. 1–7.
19. E. Weber, N. Marzo, D. P. Papadopoulos, A. Biswas, A. Lapedriza, F. Ofli, M. Imran, and A. Torralba, "Detecting natural disasters, damage, and incidents in the wild," in *The European Conference on Computer Vision (ECCV)*, August 2020.
20. L. Rokach, "Ensemble methods for classifiers," *Data mining and knowledge discovery handbook*, pp. 957–980, 2005.
21. S. Marcel and Y. Rodriguez, "Torchvision the machine-vision package of torch," in *Proceedings of the 18th ACM international conference on Multimedia*, 2010, pp. 1485–1488.

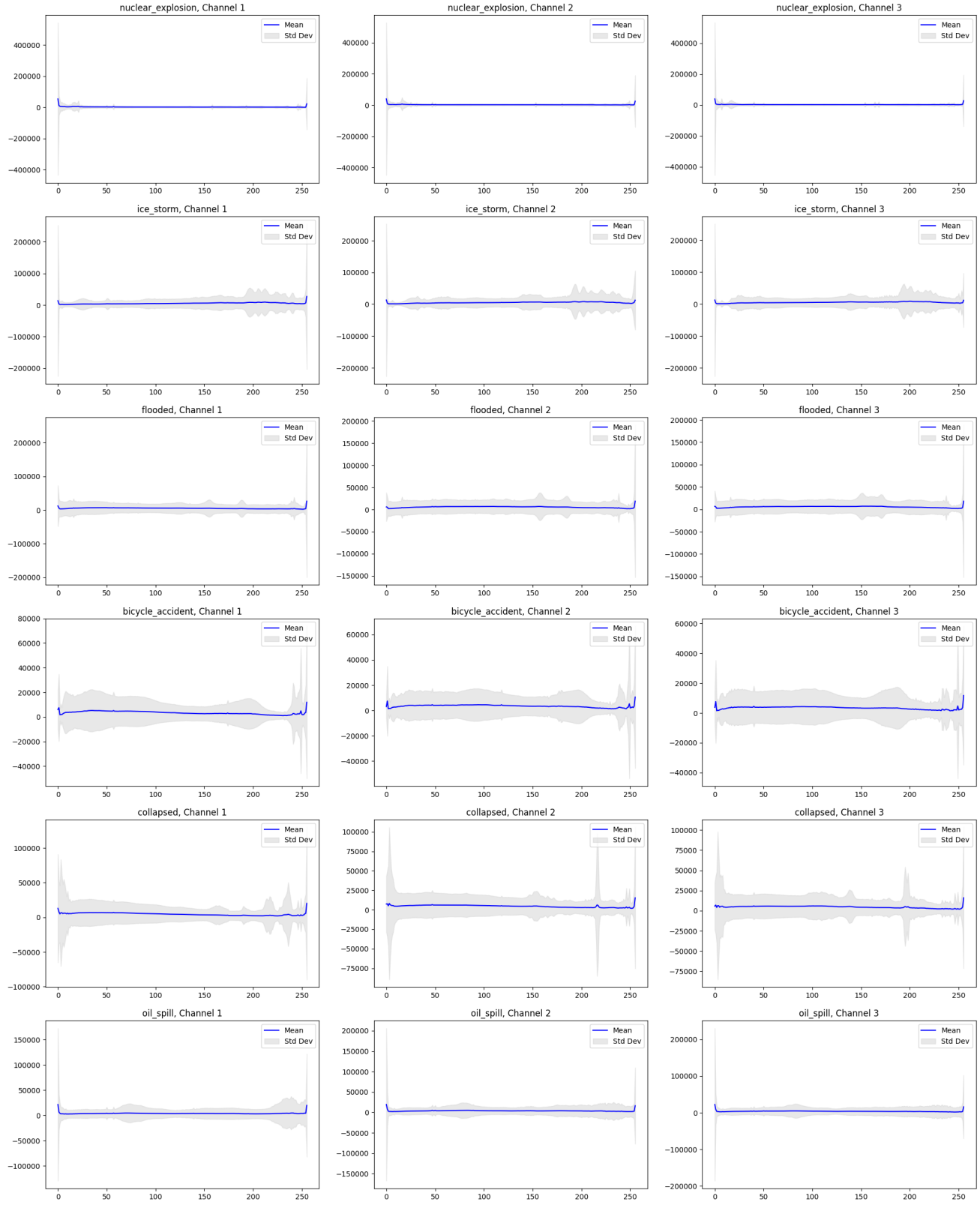

Fig. 11: Plot of the mean with std dev of the histograms of the RGB Channels

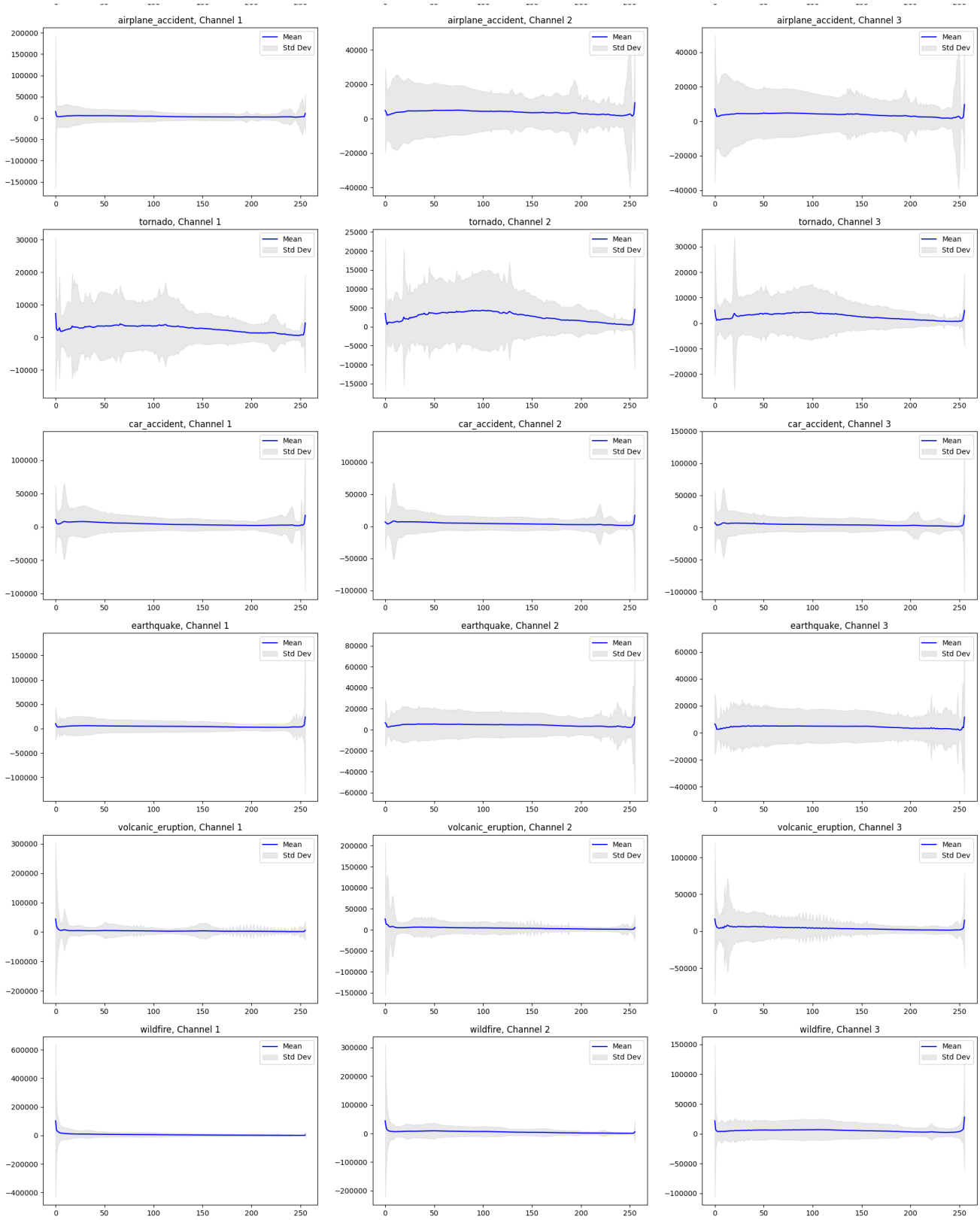

Fig. 12: Plot of the mean with std dev of the histograms of the RGB Channels

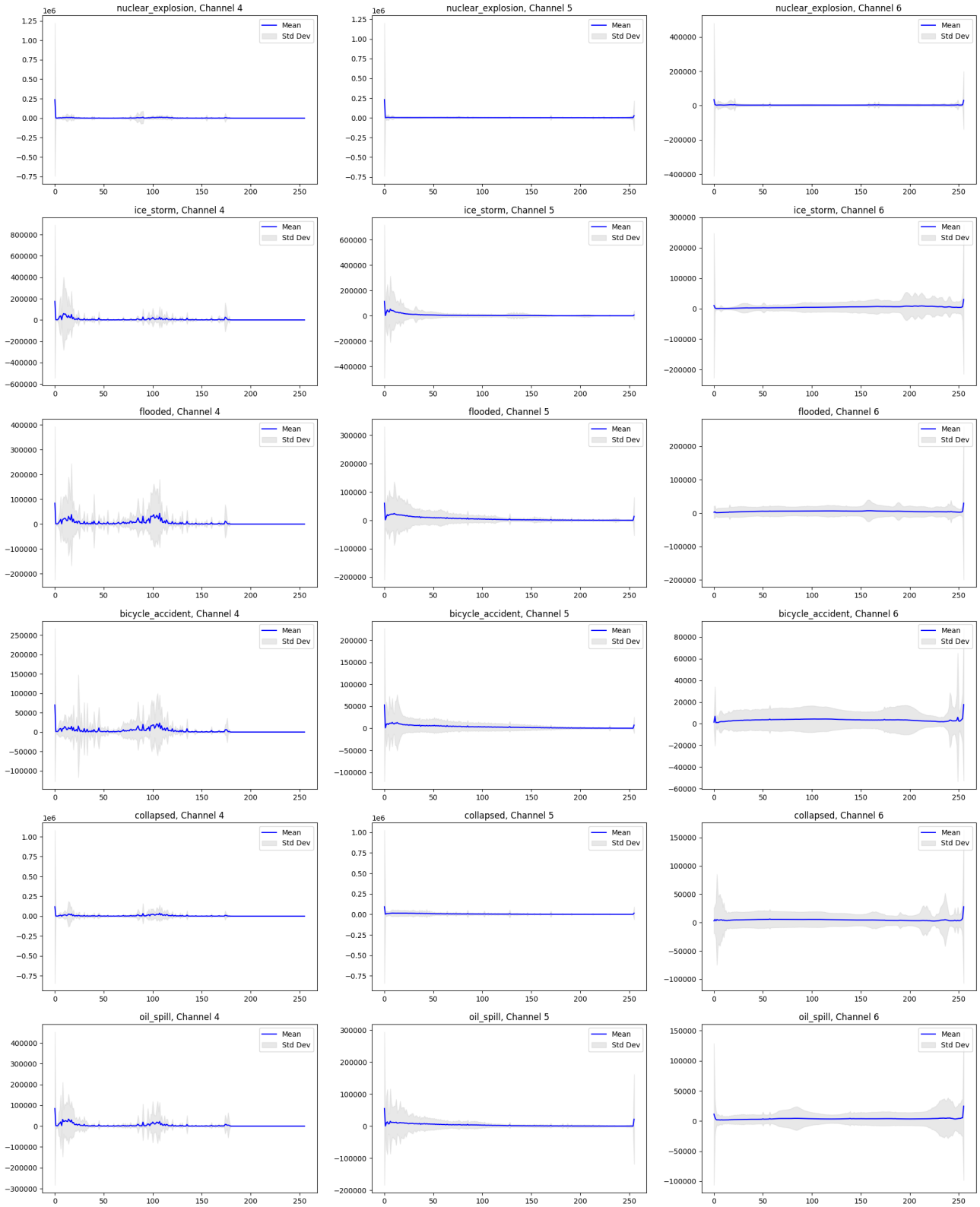

Fig. 14: Plot of the mean with std dev of the histograms of the HSV Channels

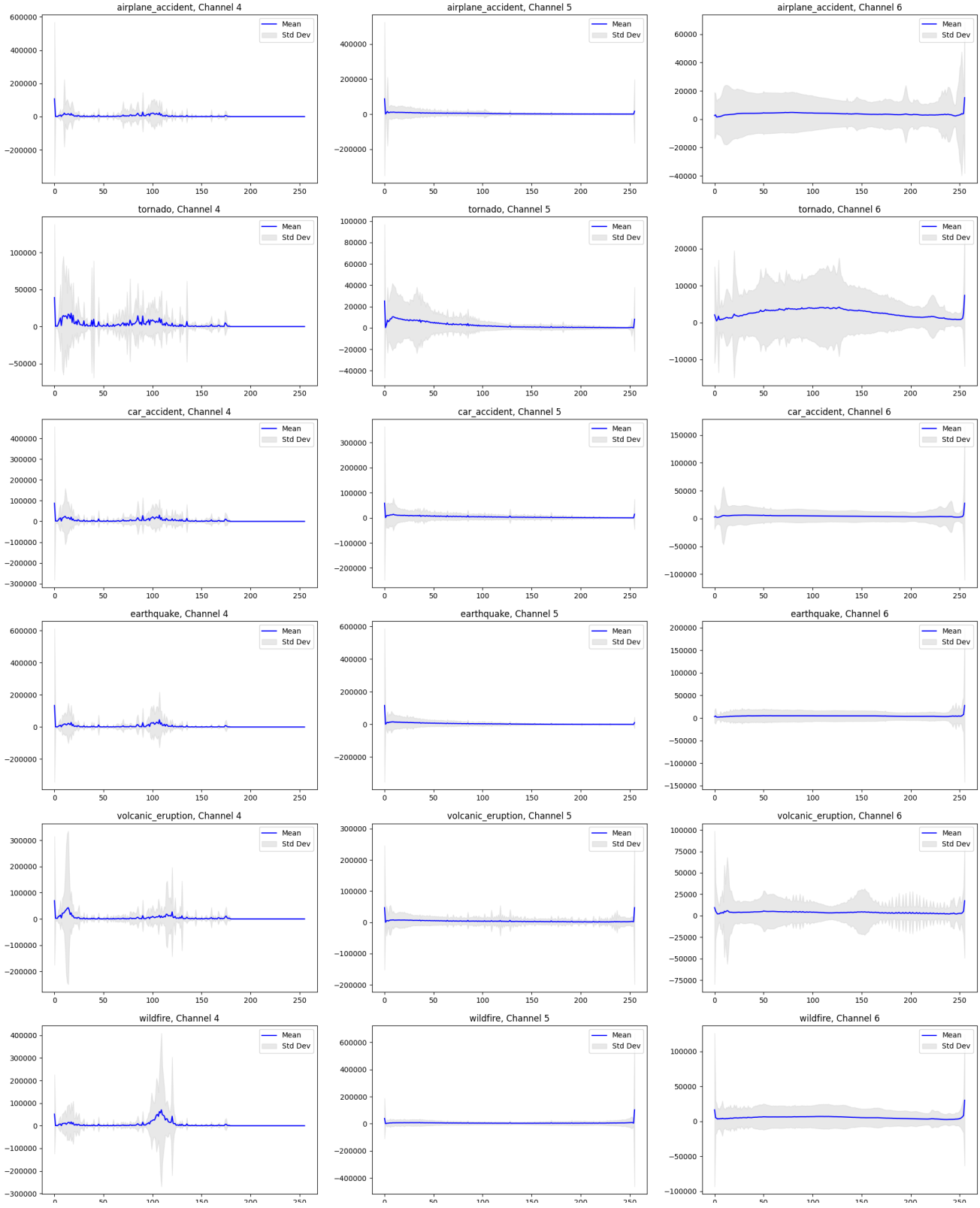

Fig. 15: Plot of the mean with std dev of the histograms of the HSV Channels

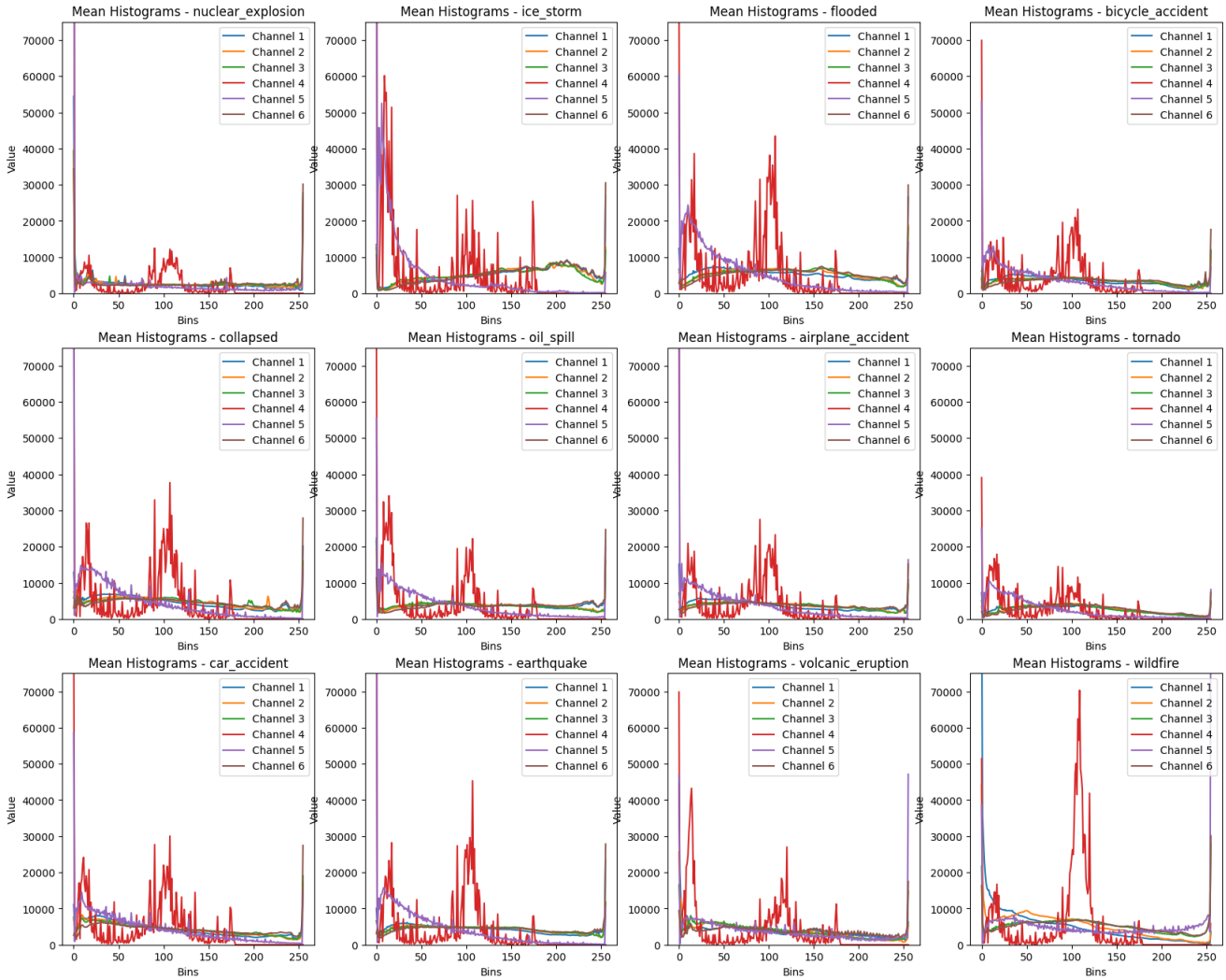

Fig. 17: Plot of the mean of RGB and HSV Channels for the categories

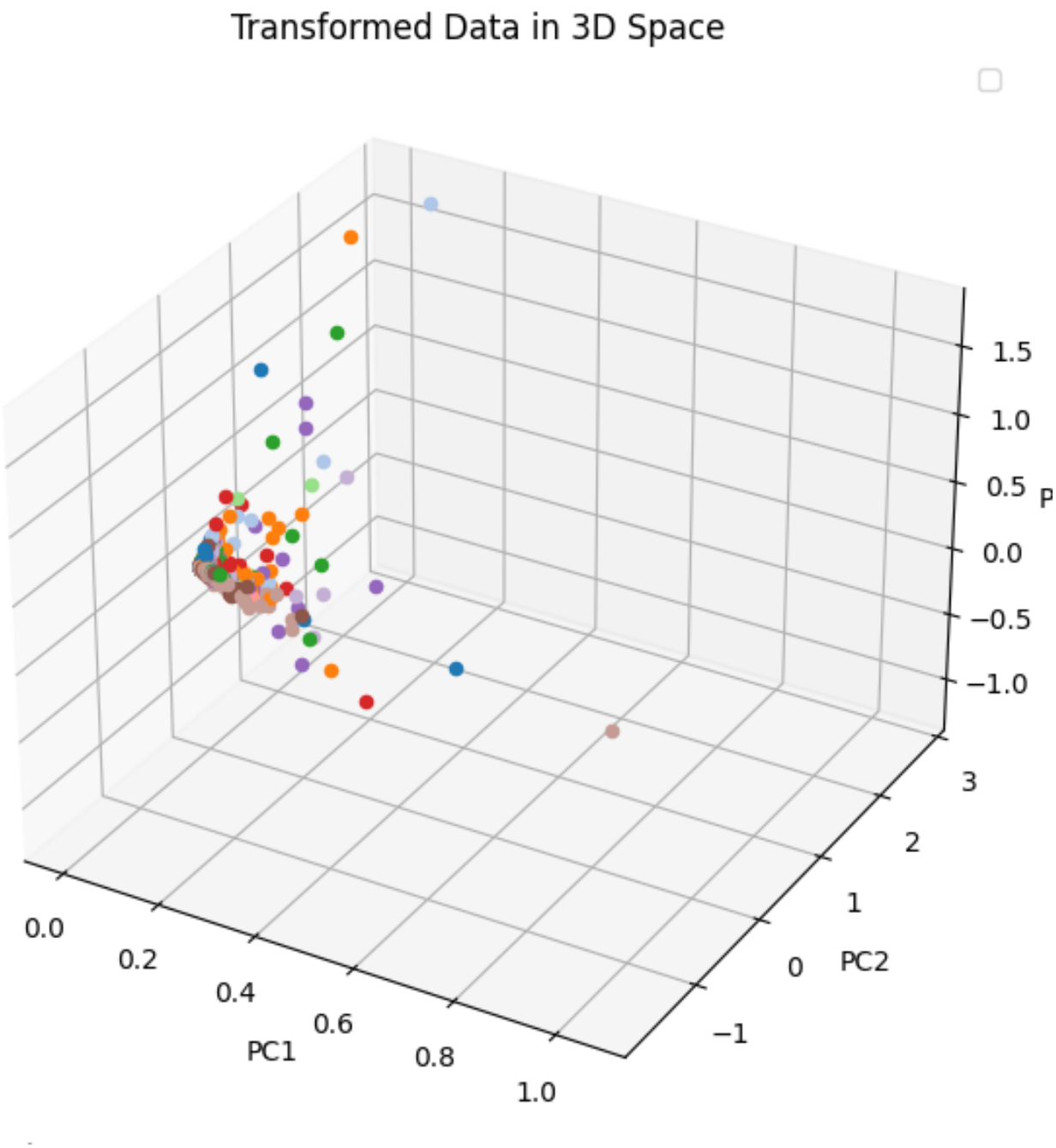

Fig. 19: Plot of the PCA of the red channel histogram for the categories

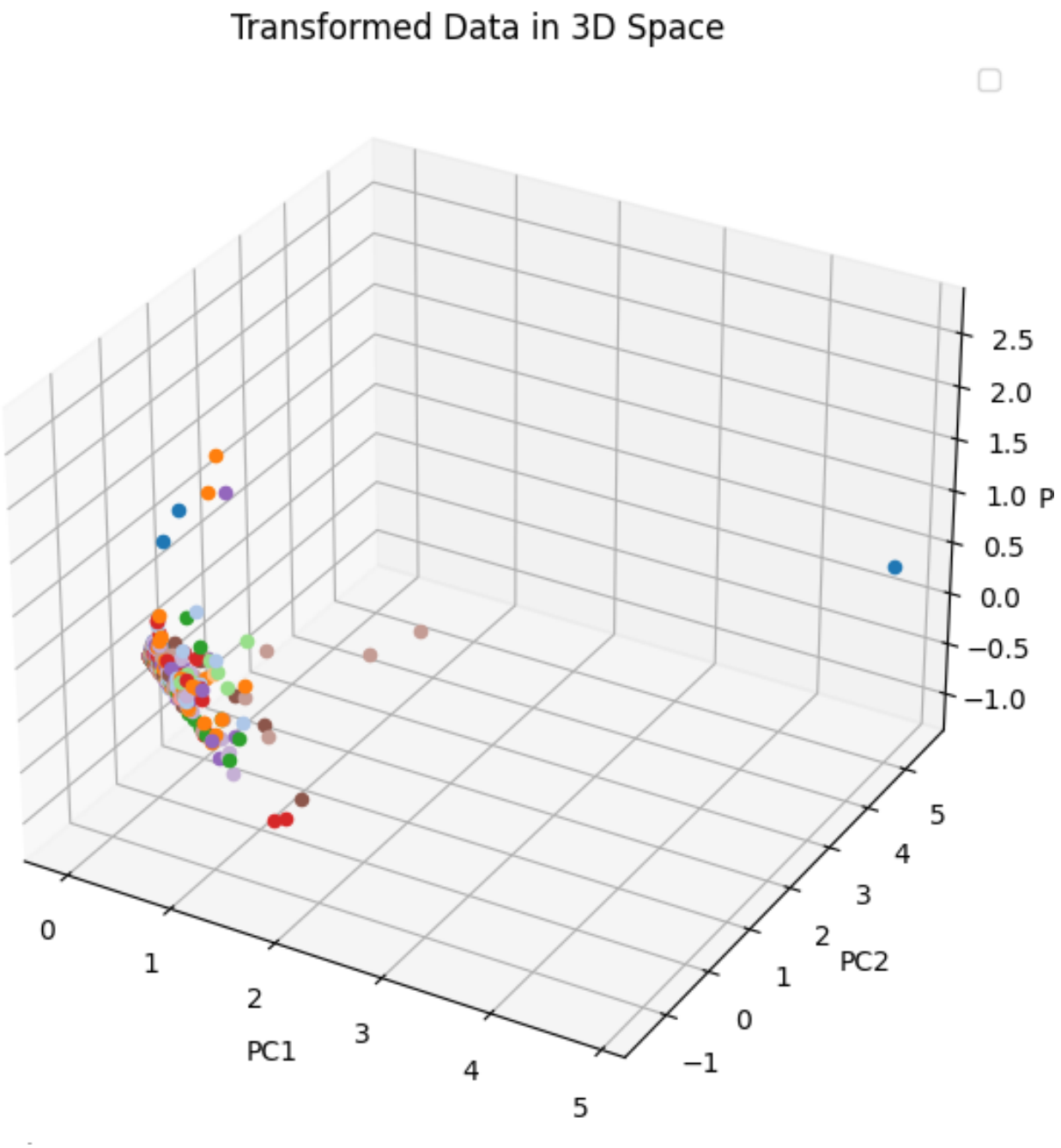

Fig. 21: Plot of the PCA of the green channel histogram for the categories

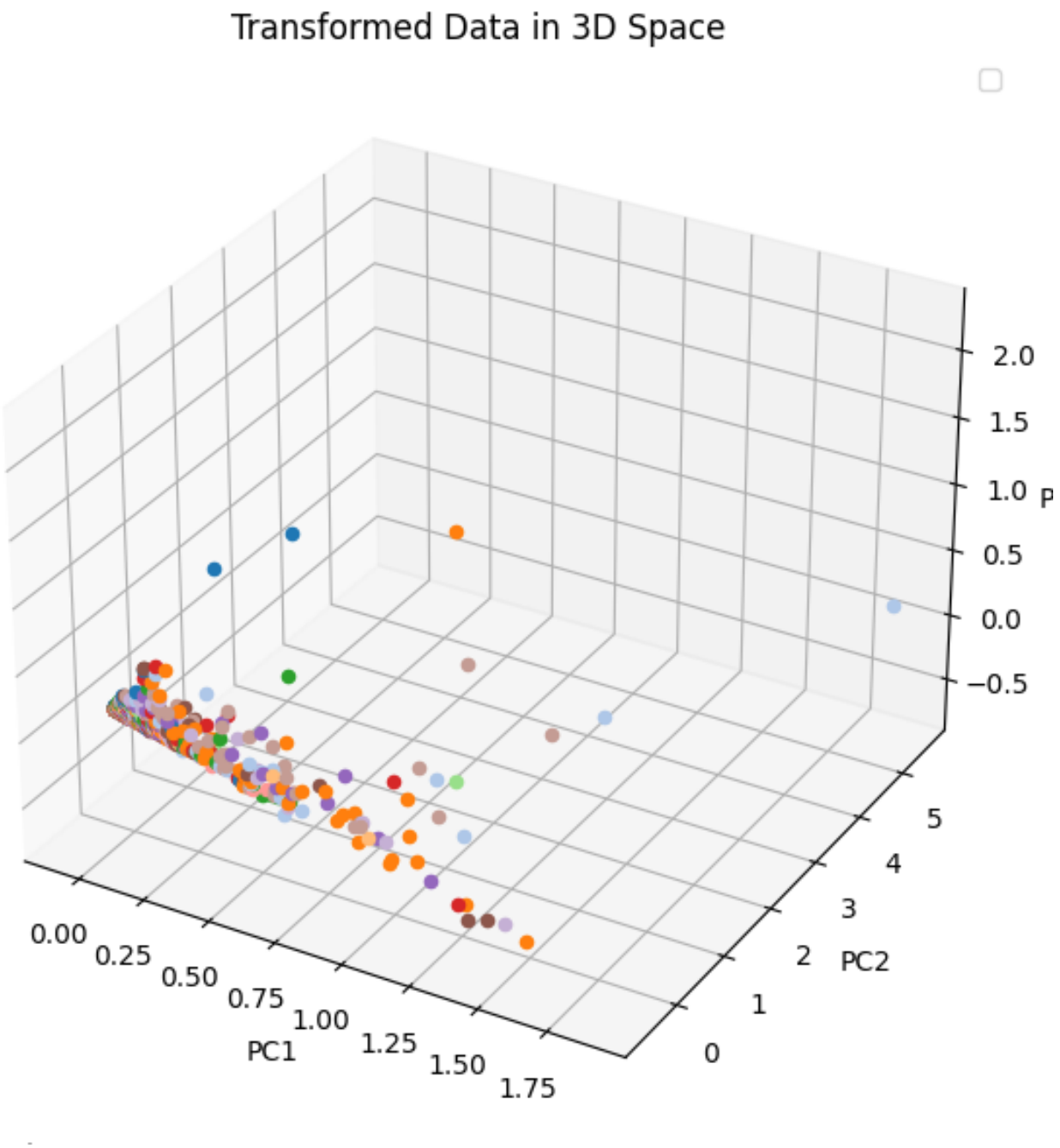

Fig. 23: Plot of the PCA of the blue channel histogram for the categories

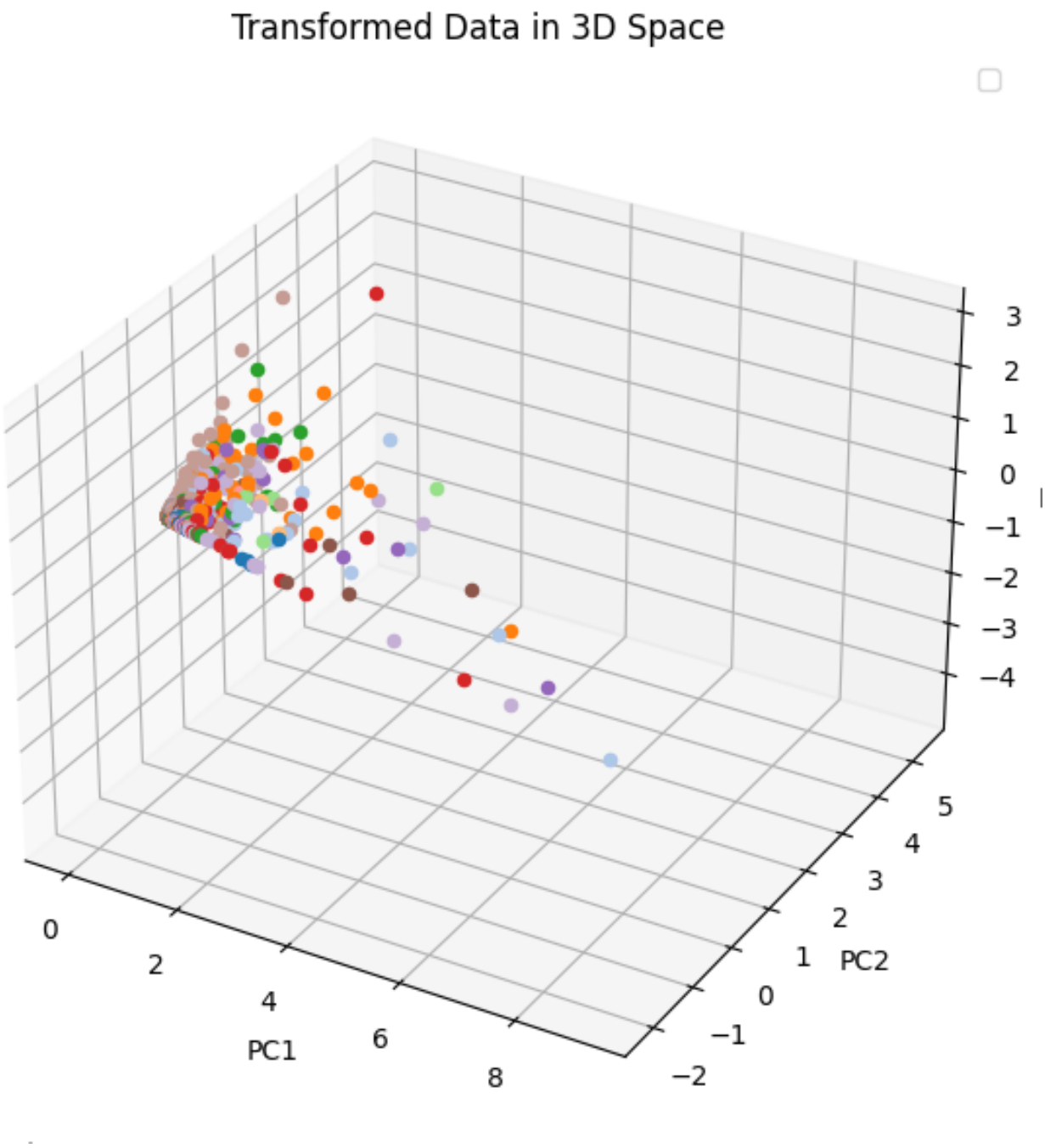

Fig. 25: Plot of the PCA of the hue channel histogram for the categories

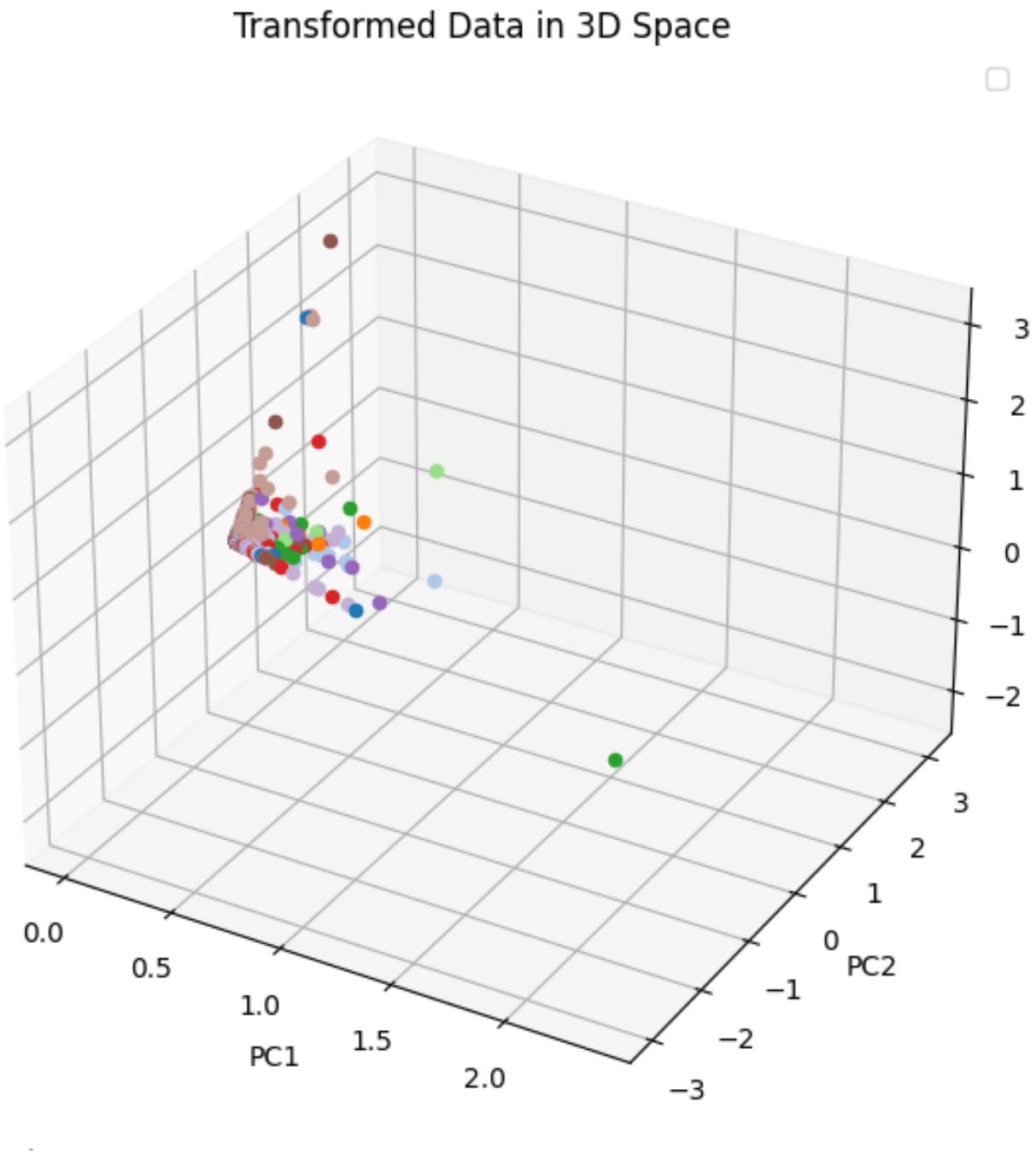

Fig. 27: Plot of the PCA of the saturation channel histogram for the categories

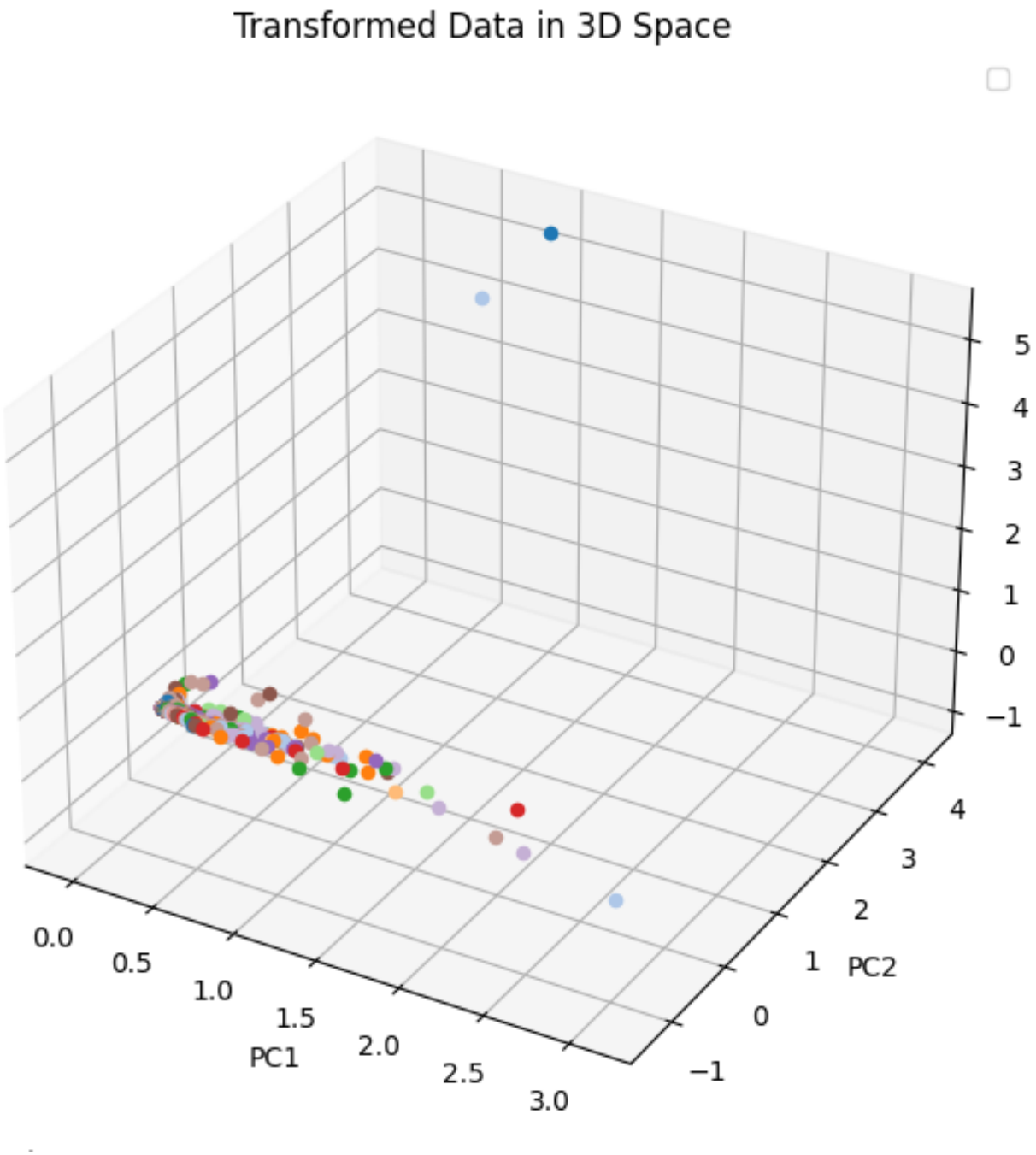

Fig. 29: Plot of the PCA of the value channel histogram for the categories

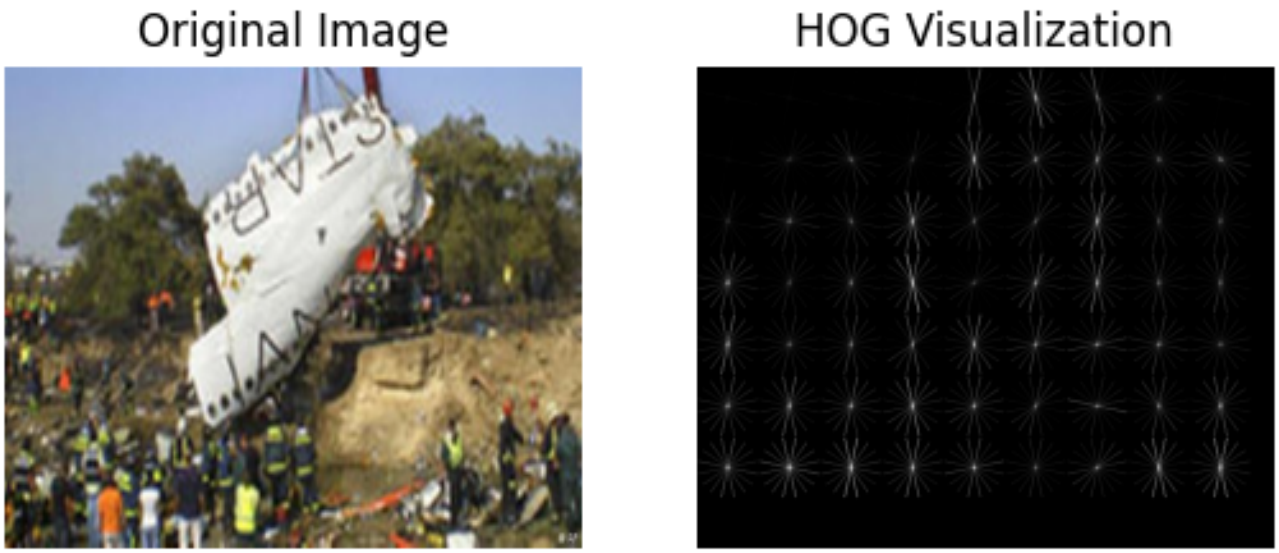

Fig. 31: In the left we have the original image and in the right we have the HOG image, we can see that it barely distingues the shape of the airplane object in the image, in this image generation we use a parameter of $\text{pixels}_per_cell = (64, 64) cells_per_block = (1, 1)$

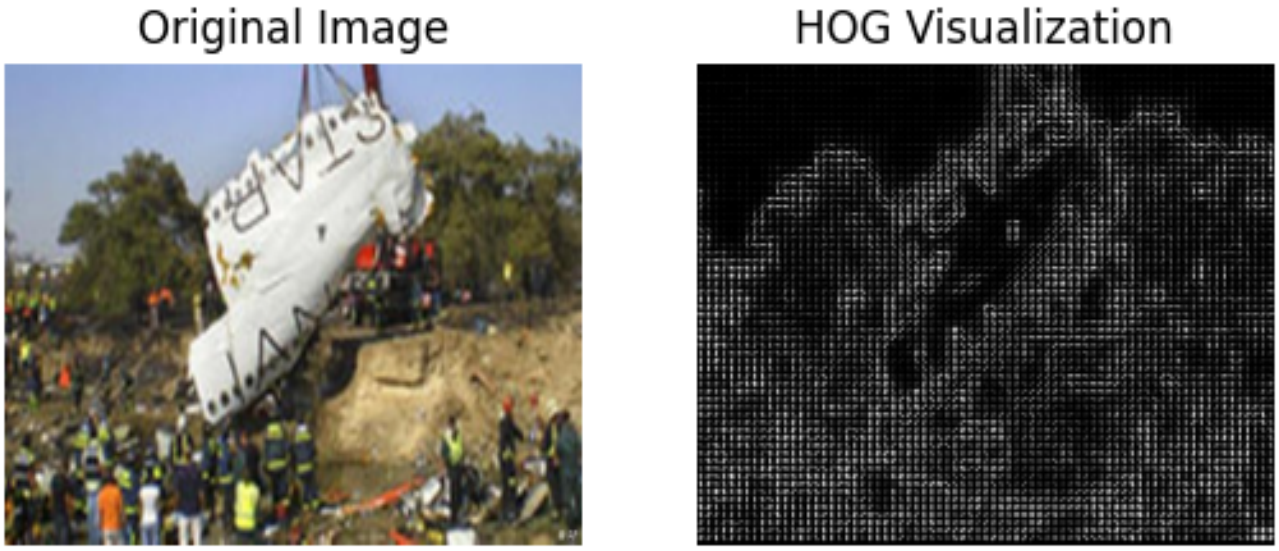

Fig. 33: In the left we have the original image and in the right we have the HOG image, we can see that it distingues the shape better than in the previous figure, in this image generation we use a parameter of $\text{pixels}_per_cell = (8, 8) cells_per_block = (2, 2)$