Automatic Target Recognition for CT-based Airport Checkpoint Screening

Hetansh Patel
Computer Science Department
Georgia State University
Atlanta, USA

Meet Patel
Computer Science Department
Georgia State University
Atlanta, USA

Patric Kun
Computer Science Department
Georgia State University
Atlanta, USA

***Abstract*— This paper aims to develop a multiclass automatic target recognition system for airport security screening using CT-based images. The system uses Machine Learning techniques to train classifiers capable of recognizing different targets, including saline, rubber, and clay. The classifiers are trained using a histogram feature extraction technique and include Support Vector Machines (SVM), SVM with Principal Component Analysis (PCA), K-Nearest Neighbors (KNN), Logistic Regression, and AdaBoost with Decision Tree. The performance of each classifier is evaluated based on the recall rate and precision rate.**

I. Introduction

The current airport screening process utilizes 2-D X-ray-based screening, which requires the removal of laptops and 3-1-1 liquids from the bags during checkpoint screening. This process can be inconvenient for passengers and can lead to longer wait times. CT-based screening is gaining popularity as it can produce a 3-D image that can be viewed and rotated 360 degrees for a more comprehensive analysis. The goal is to keep laptops and 3-1-1 liquids inside the bag during checkpoint screening.

To effectively analyze the CT-based images and predict the objects inside the luggage, a machine learning algorithm can be incorporated into the screening system. In this project, we explore and implement several machine-learning algorithms to evaluate their effectiveness in predicting and detecting targets such as saline, rubber, or clay from the image dataset. In cases where the image does not contain any target, the algorithm should predict it as a non-target to avoid false predictions.

The proposed system can improve the accuracy and efficiency of airport screening, thereby enhancing the safety and security of air travel. This study contributes to the development of automatic target recognition systems for airport screening and can be a basis for future research in this area.

II. Dataset

The dataset used in this study consists of 3D CT images collected for airport security screening purposes. The training dataset includes a total of 1005 3D images, including 707 non-target images and 298 target images, with 106 images of Saline, 112 images of rubber, and 80 images of clay. The testing dataset includes unseen 467 3D images for testing predictions.

The images in the dataset may vary in size and orientation. A sample of the 3D images is shown in Fig. 1, which demonstrates the variation in object sizes and orientations.

Overall, this dataset provides a diverse set of images that can be used to train and evaluate our target recognition system. However, it should be noted that the dataset may not be representative of all possible objects that can be encountered during airport security screening, and additional data may be needed to further improve the system's performance.

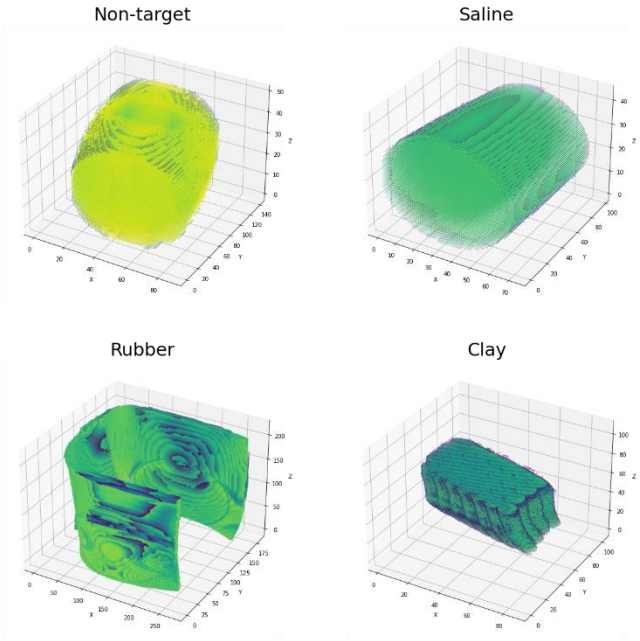

Fig. 1. Sample of 3D images in the dataset showing different objects with varying sizes and orientations

III. Feature Extraction

To extract features from the CT-based images, we used a histogram feature extraction technique. The technique involves creating a histogram of pixel intensities in the image and using the resulting distribution as a feature vector for each image.

In our implementation, we used a histogram with 250 bins to capture the intensity variations in the image. We set the pixel intensity range from 500 to 3000 to exclude irrelevant intensities such as 0 pixels, which are typically just background. We also normalized the histogram to ensure that each feature has a similar scale.

The resulting feature vector for each image represents the distribution of pixel intensities in the image, which can be used to train classifiers for target recognition. By using a histogram feature extraction technique, we were able to capture important information about the image's intensity distribution, which can be useful in differentiating between different target types.

IV. Classifier

In this study, we explored several machine learning algorithms to develop an automatic target recognition system for airport security screening using CT-based images. We used Python to implement and evaluate the performance of each classifier. We also used scikit-learn[6] library for

machine learning classifiers and GridSearchCV to train our models and select the best hyperparameters to achieve optimal performance.

A. SVM

The Support Vector Machine was first proposed in 1995 by Cortes and Vapnik. The basic concept of SVM is to define a linear space on which we can find an optimal hyperplane for a convex programming problem. Based on the Mercer kernel expansion theorem constructing a non-linear mapping sample space is mapped to a higher dimension so that a linear machine can be applied to the feature space to solve the non-linear classification and regression problems. The most common classification is to accurately separate the two types of samples and maximize the distance between two types of data points.[1]

We used the radial basis function (RBF) kernel for the SVM classifier with the parameter C set to 5. We also performed hyperparameter tuning using the GridSearchCV method provided by scikit-learn. We varied the C parameter from 1.0 to 10.0 and tested three different kernels: sigmoid, RBF, and polynomial.

B. SVM with PCA

Principal Component Analysis (PCA) is a widely used technique for dimensionality reduction and feature extraction in machine learning. The mathematical foundations of PCA were developed by Karl Pearson in 1901 and were later extended by Harold Hotelling in the 1930s. PCA has been applied to a wide range of fields, including image and signal processing, finance, and biology.[2] SVM with PCA is used to reduce the dimensionality of high-dimensional datasets while maintaining information from the data. The first step is to apply PCA to input data to reduce its dimensionality. The data is then input into the SVM algorithm for classification. By reducing the dimensionality of the data, the complexity of the SVM algorithm is reduced and the model is less likely to overfit the training data. The biggest benefit of this algorithm is improved accuracy and speed of classification.

To reduce the dimensionality of the feature vector, we applied Principal Component Analysis (PCA) with n_components = 50. We then used SVM with kernel='rbf' and C=10.0 to classify the reduced feature vectors. To obtain the best performance, we applied GridSearchCV with the hyperparameters n_components, C, and kernel tuned as follows: n_components = [2, 5, 10, 20, 50], C = [1.0, 5.0, 7.0, 10.0], kernel = ['sigmoid', 'rbf', 'poly'].

C. KNN

The K-Nearest Neighbors algorithm, also known as KNN is a supervised learning classifier. This algorithm uses the proximity of other k samples to classify the selected sample and group it to the most appropriate set of data points. The main difference between using KNN and regression algorithms is that KNN is a classifier algorithm that is used for discrete data points and regression is used for continuous data points. We need to assign the value of K by observing the algorithm's behavior. A very high value of K might lead to underfitting, while a low value of K might lead to overfitting. Some disadvantages of using KNN are the curse of dimensionality, and it is prone to overfitting.[3]

For the K-Nearest Neighbors (KNN) algorithm, we utilized the following parameters: n_neighbors=5, metric=manhattan, and weights=distance. To fine-tune the hyperparameters and optimize the performance of the algorithm, we employed GridSearchCV with the following values: n_neighbors=[3, 5, 7, 9], weights=['uniform', 'distance'], metric=['euclidean', 'manhattan'].

D. Logistic Regression

In regression problems, we usually have a dependent variable that varies with more than one variable. Logistic regression is used for classifications as well as prediction. It is based on a statistical model called logit. It estimates the probability of an event occurring which is between 0 and 1. There are different types of Logistic regression such as Binary, Multinomial, and Ordinal. As the name implies, binary has only two outcomes, 0 and 1. Multinomial, dependent variable has more than three outcomes, and ordinal has more than three outcomes but in a defined order.[4]

We used the following parameters for Logistic Regression: C=1, penalty= l1, and solver= saga. To improve its performance, we applied GridSearchCV with the following hyperparameters tuned: solver=['newton-cg', 'sag', 'saga', 'lbfgs'], C=[0.01, 0.1, 1, 10], penalty= ['l1', 'l2', 'elasticnet'].

E. AdaBoost with Decision Tree

AdaBoost with a decision tree is an ensemble learning algorithm that combines the power of a decision tree with the AdaBoost algorithm. In this algorithm, multiple decision trees are created and each one is trained on a modified version of the original dataset. The modifications include increasing the weight of misclassified examples and decreasing the weight of correctly classified examples. For each iteration, AdaBoost will select the most informative feature to split the data and will create a decision tree based on that feature.[5]

For the Decision Tree, we set the following parameters: criterion='gini', max_depth=8, and min_samples_split=2. For the AdaBoost, we set the number of estimators to 100. To optimize the performance of the model, we applied GridSearchCV with the following hyperparameters tuned: max_depth=[6, 7, 8], min_samples_split=[2, 3, 4, 5], and criterion=['gini', 'entropy'] for the Decision Tree, and n_estimators=100 for AdaBoost.

V. Results

The performance of the classifiers was evaluated using various evaluation metrics. The classifiers used in this study were Support Vector Machine (SVM) with PCA, K-Nearest Neighbors (KNN), AdaBoost with Decision Tree (ABC-DTC), and Logistic Regression (LR). For each classifier, we report the recall rate, precision rate, and recall rate for each of the three different types of soil samples: Saline, Rubber, and Clay.

A. SVM

```
'svm_predictions.csv'
Recall Rate:0.60976 75/123
Precision Rate:0.88081 303/344
Saline Recall Rate:0.56098 23/41
Rubber Recall Rate:0.59091 26/44
Clay Recall Rate:0.68421 26/38
```

Fig. 2. Results for SVM

For SVM, we obtained a recall rate of 0.60976 and a precision rate of 0.88081. The recall rates for Saline, Rubber, and Clay samples were 0.56098, 0.59091, and 0.68421, respectively.

B. SVM with PCA

```
'svm_pca_predictions.csv'
Recall Rate:0.63415 78/123
Precision Rate:0.87209 300/344
Saline Recall Rate:0.56098 23/41
Rubber Recall Rate:0.5 22/44
Clay Recall Rate:0.86842 33/38
```

Fig. 3. Results for SVM with PCA

For SVM with PCA, we obtained a recall rate of 0.63415 and a precision rate of 0.87209. The recall rates for Saline, Rubber, and Clay samples were 0.56098, 0.5, and 0.86842, respectively.

C. KNN

```
'knn_predictions.csv'
Recall Rate:0.6748 83/123
Precision Rate:0.82267 283/344
Saline Recall Rate:0.7561 31/41
Rubber Recall Rate:0.56818 25/44
Clay Recall Rate:0.71053 27/38
```

Fig. 4. Results for KNN

For KNN, we obtained a recall rate of 0.6748 and a precision rate of 0.82267. The recall rates for Saline, Rubber, and Clay samples were 0.7561, 0.56818, and 0.71053, respectively.

D. Logistic Regression

```
'lr_predictions.csv'
Recall Rate:0.41463 51/123
Precision Rate:0.88953 306/344
Saline Recall Rate:0.29268 12/41
Rubber Recall Rate:0.40909 18/44
Clay Recall Rate:0.55263 21/38
```

Fig. 5. Results for Logistic Regression

For LR, we obtained a recall rate of 0.41463 and a precision rate of 0.88953. The recall rates for Saline, Rubber, and Clay samples were 0.29268, 0.40909, and 0.55263, respectively.

E. AdaBoost with Decision Tree

```
'abc_dtc_predictions.csv'
Recall Rate:0.60976 75/123
Precision Rate:0.90698 312/344
Saline Recall Rate:0.56098 23/41
Rubber Recall Rate:0.5 22/44
Clay Recall Rate:0.78947 30/38
```

Fig. 6. Results for AdaBoost with Decision Tree

Finally, for ABC-DTC, we obtained a recall rate of 0.60976 and a precision rate of 0.90698. The recall rates for Saline, Rubber, and Clay samples were 0.56098, 0.5, and 0.78947, respectively.

VI. Conclusion

In conclusion, we have explored several classifiers including Support Vector Machine with PCA, K-Nearest Neighbor, Logistic Regression, and AdaBoost with Decision Tree, and evaluated their performance on our dataset. Among these classifiers, K-Nearest Neighbor achieved the highest recall and precision rates, followed closely by Support Vector Machine with PCA and AdaBoost with Decision Tree.

Future work could focus on exploring ensemble methods that combine the strengths of multiple classifiers to achieve even higher performance. Additionally, we could investigate the impact of different feature selection techniques and data preprocessing methods on classifier performance. Lastly, we could also consider using deep learning techniques such as convolutional neural networks or recurrent neural networks for classification tasks.

References

[1] Chang, C.-C., & Lin, C.-J. (2011). LIBSVM: A library for support vector machines. ACM Transactions on Intelligent Systems and Technology, 2(3), 1-27.

[2] K. Pearson, "On lines and planes of closest fit to systems of points in space," Philosophical Magazine, vol. 2, no. 11, pp. 559-572, Nov. 1901.

[3] "KNN - Overview | IBM." [Online]. Available: https://www.ibm.com/topics/knn. [Accessed: 02-May-2023].

[4] "Logistic Regression - Overview | IBM." [Online]. Available: https://www.ibm.com/topics/logistic-regression. [Accessed: 02-May-2023].

[5] Zhang, J., Zhang, Y., & Hu, W. (2019). AdaBoost and its extensions for improving decision tree learning: A review. Journal of Artificial Intelligence and Soft Computing Research, 9(3), 209-218.

[6] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Vanderplas, J. (2011). Scikit-learn: Machine learning in Python. Journal of machine learning research, 12(Oct), 2825-2830.