Drone Image Classification using SVM and CNN

Eric Sung

A25300511

CS 588 - 01

Dr. Vineetha Menon

November 30, 2022

Table of Contents

Introduction………………………………………………………………………………………. 2

Discussion of Method……………………………………………………………………………. 4

Test Results………………………………………………………………………………………. 6

Analysis….………………………………………………………………………………………. 8

Conclusion……………………………………………………………………………………….. 9

References………………………………………………………………………………………..10

Appendix…………………………………………………………………………………………10

Introduction

Unmanned aerial vehicles (UAVs) or in the form of drones have become a widely accessible technology over the past few years for both the public and private sectors. Their ability to navigate and offer means for remote surveillance of challenging environments have become a great utility for many remote sensing defense applications. One of many humanitarian uses of drones include its operation in combat search and rescue missions as an AI-based assistive automation utility, in order to successfully detect our targets of interest using AI as well as collect information about the surrounding environments through navigation. Hence, in potentially hazardous situations like combat search and rescue missions, drone-based AI autonomous systems can provide geospatial locations of the victims and meta data of the environment through elevated perspectives and wider angles that are difficult to achieve from the ground.

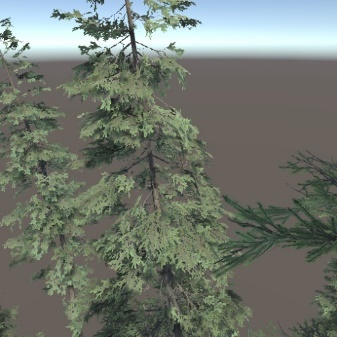
This project itself constitutes a small portion of an extensive research project that involves development of a fully autonomous assistive drone AI, assessment of human behavioral change with an assistive technology, and evaluation of overall utility AI-based assistive automation can provide in search and rescue scenarios in a simulated environment. A working simulation has been constructed with a dynamic, large-scale environment and controllable entities like a human agent and a drone. The human agent entity is able to rescue hostages that are scattered apart in the environment simply by colliding with the hostage entities, and the drone entity can survey the environment both autonomously and with manual control. The drone can also mark the hostage locations once it detects them. The goal of the human player in the simulation is to rescue all hostages in the environment, and the player can utilize the drone to expedite the process.

The assistive drone AI system consists of three key tasks: autonomous search, object detection, and rational decision making. Autonomous search includes path planning and navigation, object detection is mainly about image processing and classification, and rational decision making constitutes setting search priority, data display, and etc. These components should come together to form a reliable assistive drone AI that facilitates human agents during search and rescue missions without human control. This project focuses on the object detection aspect of the drone AI.

Most commercial drones, nowadays, have cameras attached for manual control and videography purposes although this has been an issue of surveillance on private property using sophisticated cameras on drones (“McNeal”). In dire situations like disaster relief and combat search and rescue (CSAR) missions, however, drone cameras are of great importance as they provide human-friendly video feeds unlike other sensors, such as infrared and sonar sensors, which, too, are not readily available in most drones. Acknowledging the abundance of drones with cameras and their utility for human monitoring, using the cameras for object and hostage detection is crucial.

In this project, two of the most commonly used image classification models were assessed for the implementation of object detection for the drone AI. The models used are the support vector machine (SVM) and the convolutional neural network (CNN). These classifiers are supervised learning models, meaning that they require labeled data to study the pattern, and they are both very powerful models in machine learning. These classification models were each tested with two different methods of preprocessing, grayscale and principal component analysis (PCA), to determine the ideal classifier and data processing method for object detection.

Fig. 1. Sample images from the Dataset



The training data were obtained from the author of the paper “Feasibility of AI-Driven Autonomous Systems for Target Detection in Operational Environment in Army Missions”, Daniel Pham. The dataset contains 2,000 (512x512 pixel) color images for each of the 7 different objects. The data were collected directly from the simulation by taking thousands of snapshots of each object at various different angles and distances. With a single source of light in space, change in sensitivity of light can be observed at different camera angles, which is true to the nature of the simulation and the real-world physics. The images were resized to 64 pixels by 64 pixels to facilitate the classification prior to preprocessing and classification, and only four of the seven objects were chosen (drone, hostage, house, and tree). Four sample images from the dataset is shown in Fig. 1.

Presumably, since CNN is one of the most popular image classification models, it would be the best model in general, and PCA is very effective at extracting key features from data. Thus, CNN with PCA would be the best pair for the application.

Discussion of Methods

Support Vector Machine (SVM) is a supervised machine learning model used in classification and pattern recognition. SVM, first, plots each data as a point in a n-dimensional space with its dimension or feature representing an axis in the hyperspace (“Ray”). The SVM algorithm finds the hyperplane, or the decision boundary, that maximizes the margin distances between data points of different classes (Fig. 2). Once the optimal hyperplane is calculated, new data points being inserted are classified by their positions relative to the hyperplane.

In this project, SVM was pulled directly from scikit-learn framework’s svm package. The svm package contains svc (Support Vector Classifier) class for classification purposes. Linear kernel was chosen for the method of hyperplane calculations, and each input images was flattened into 1-dimensional data prior to being fed into the model.

Fig. 2. Support Vector Machine Visualization

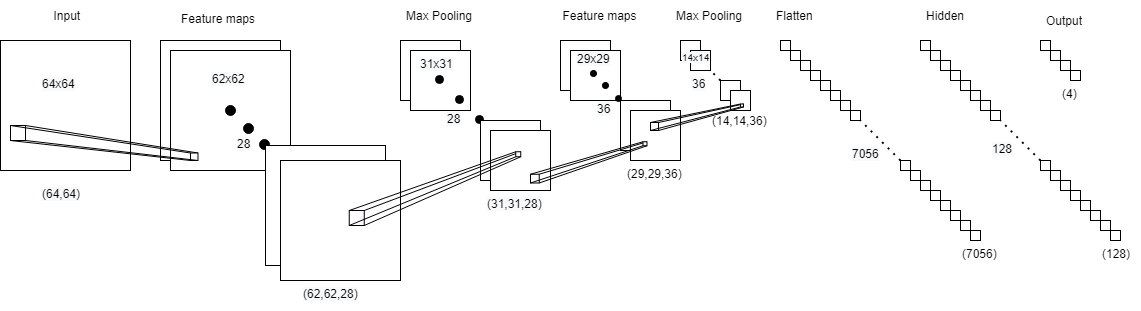
(“Ray”)



Convolutional Neural Network (CNN) is a supervised machine learning model most commonly used for image classification and object recognition tasks. CNN is a type of artificial neural network specialized in analysis of visual imagery. Unlike traditional neural networks, CNN has a convolutional layer that generates a feature map from an image data using filter. A filter is a matrix (usually 3x3) with weights that translates across the input matrix to find the dot product of itself to a portion of the image it is masking, which represents the feature of the image in the form of a feature map. Then, a pooling layer is applied for down-sampling or dimensionality reduction, which involves resizing the input matrix with another filter (usually 2x2) that simply chooses one value for the making portion of the overall input matrix. Max pooling, choosing the largest value, and average pooling, choosing the average value of the masked portion, are two most common pooling strategies. Once CNN extracts the features from the input data, hidden layers use the features to complete the classification (“IBM Cloud Education”).

Keras, formally known as Tensorflow Keras, is a popular deep learning API written in Python. In this project, the CNN model was constructed using Keras’s sequential model and convolutional layers. This CNN model consists of 7 distinct layers excluding the input layer. First layer is the convolutional layer with the kernel size (filter size) of 3x3, 24 total filters to generate 24 filter maps, and Rectified Linear Unit (ReLU) transform to finalize the output into values between 0 and 1. The second layer is the max pooling layer with the pooling size of 2x2 to reduce the output size by half. The third layer is another convolutional layer with kernel size 3x3, 36 filters, and ReLU. The third layer is another max pooling layer with the same 2x2 pooling size. The fifth layer is the flatten layer, which transforms the input matrix into a 1-dimensional list or vector. The sixth layer is the hidden layer with 128 neurons and ReLU. Lastly, the seventh layer is the output layer with 4 neurons representing the four classes with softmax as the activation function to convert the outputs as values between 0 and 1 like a probability. The epoch was set to 30, and the batch size was set to 25; these values were determined from several small scale testing during validation phase.

Fig. 3. CNN LeNet Diagram



These image classification models were trained on the dataset with two different preprocessing methods and a control case without any preprocessing. The first method involves modifying the images to grayscale, which compresses RGB pixel values into fixed grayscale value, reducing the dimension of the dataset by 3 times. The second method involves PCA, reducing the dimension to the same dimension as grayscale method to keep the consistency.

The total computation time for data preprocessing, training, and testing were collected as computational performance is crucial for the real-time classification application of drone AI. The accuracy of the models were also measured.

Test Results

Fig. 4. Model Accuracies

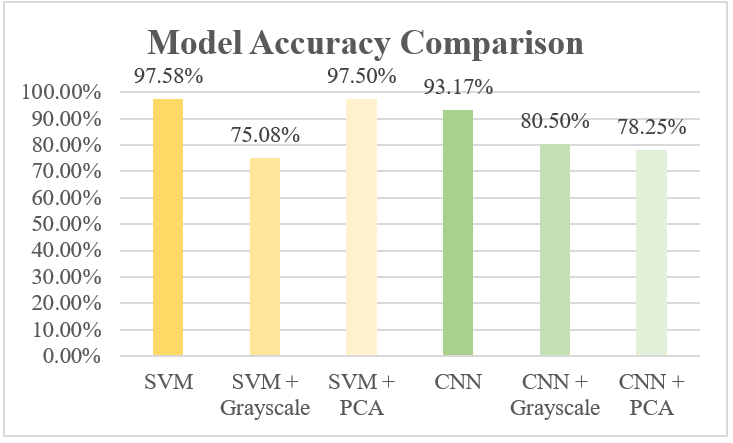
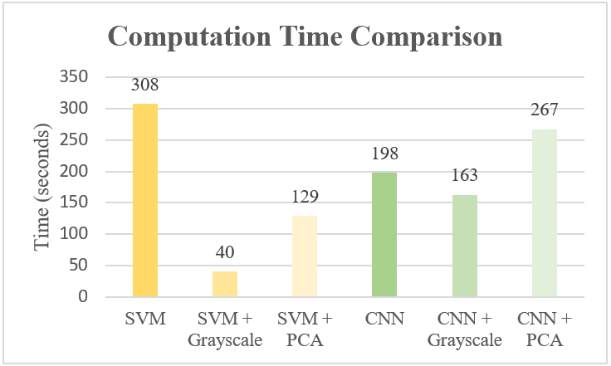


Fig. 5. Model Computation Times



In Fig. 4, overall accuracies of the models are sufficiently high with the lowest accuracy among the models being 75.08%. The highest accuracy is 97.58% from SVM without preprocessing. Surprisingly, both SVM and CNN models without preprocessing performed the best among their groups, and the grayscale preprocessing did not show any improvements in accuracy. PCA, on the other hand, was effective for SVM, but for CNN, it dropped the accuracy by approximately 20%. On the category of accuracy, SVM without preprocessing performed the best, followed by SVM with PCA and CNN.

Unlike the accuracies, there were much more variance in the computation time among the models. The greatest computation time, which is undesirable, was 308 seconds by SVM without preprocessing, and the smallest time was 40 seconds from SVM with grayscale. SVM and CNN without preprocessing had relatively high computation time. Grayscale processing reduced the computation time drastically by roughly 7 times for SVM and 15% for CNN. PCA was effective for SVM, but for CNN, it negatively affected the computation time. Overall, grayscale seems to be the best method of preprocessing in terms of the computation time, and PCA seems to be adequate for SVM but detrimental to CNN.

Fig. 6 and 7 show the confusion matrix of the models’ results. SVM, SVM with PCA, and CNN have high accuracies above 90% as shown in Fig. 4 and Fig.5, and quite clearly, these models have the strongest main diagonals in their matrices. For CNN, however, there was a frequent occurrence of mis-prediction of drone images for classifying them as hostage. Similar trends appear in SVM with Grayscale, CNN with Grayscale, and CNN with PCA, hinting an underlying similarity between drone and hostage images. Tree class, on the other hand, shows a uniformly unbiased result across the models.

Fig. 6. SVM Model Confusion Matrices

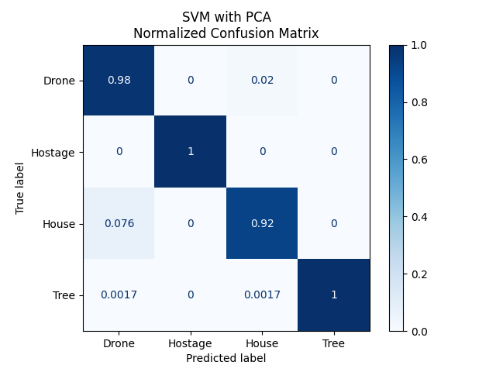
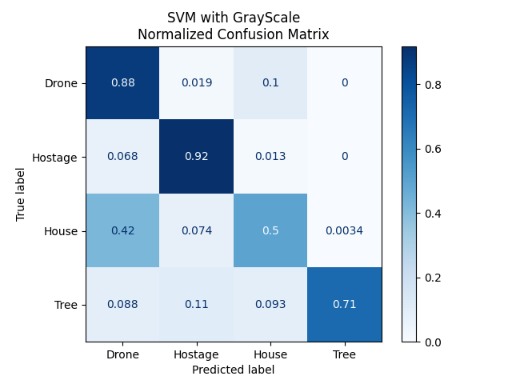
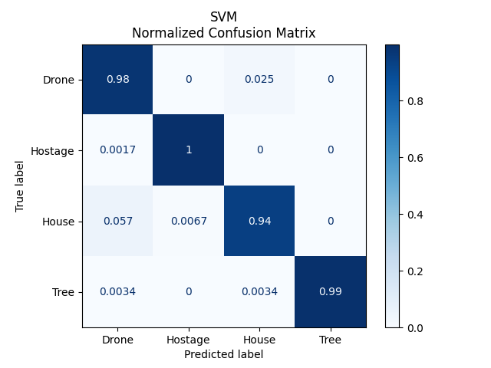


Fig. 7. CNN Model Confusion Matrices

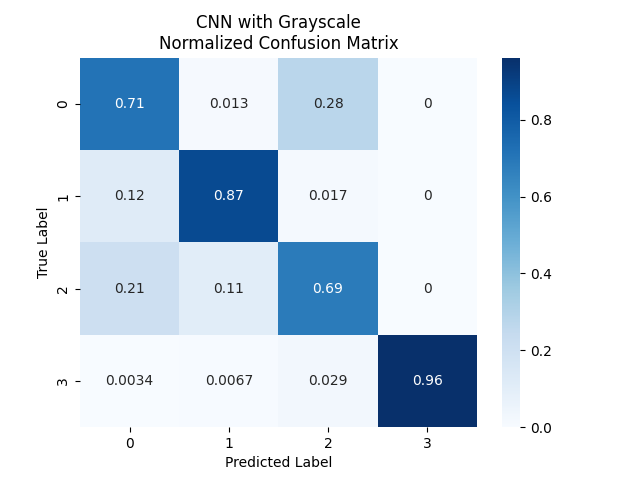
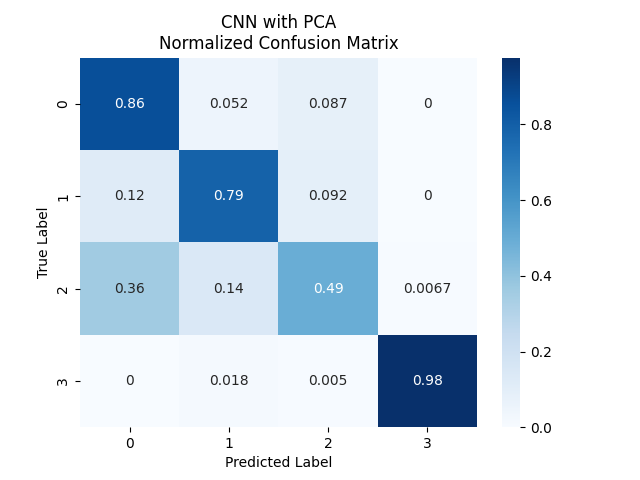
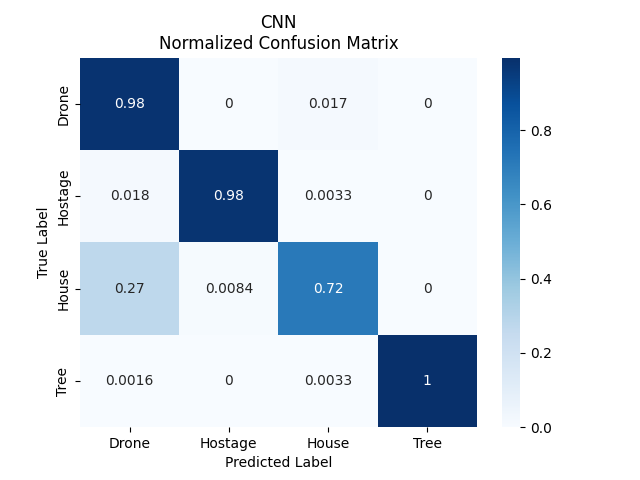
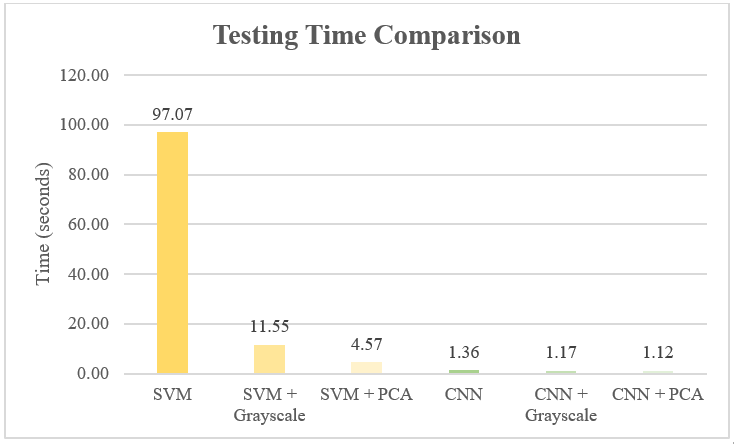


Fig. 8. Model Testing Times



Analysis

SVM without any preprocessing suffered the most from the high dimensionality of the image dataset. The total training time for SVM was 209 seconds, which was 10~20 times the training time for SVM models with preprocessing. This indicates that for computational performance, SVM prefers a smaller number of dimensions. Although the SVM model without preprocessing performed the best in terms of accuracy, it was marginally better compared to the SVM model with PCA, which had roughly ½ times computation time. The SVM model with grayscale had the fastest computation time but ended with 75% accuracy, so it would be preferred in time-intensive applications at the cost of higher error. Overall, SVM with PCA seems to be the best candidate.

The CNN models, on the other hand, had more irregular patterns. The CNN model without preprocessing performed the best amongst the classification group but was the second in computation time. Noticeably, CNN with PCA only increased the computation time but decreased the accuracy, so the pair is unacceptable. Grayscale preprocessing reduced the overall computation time at a small margin by 35 seconds, but it reduced the accuracy by 13%.

PCA was inevitably slow due to the large sample and dimension sizes. Across the board, PCA added approximately 105 seconds to the overall computation. For SVM, the dimensionality reduction from PCA cut down the training and testing time effectively from roughly 300 seconds to 15 seconds, so the role of PCA was crucial for SVM. For CNN, however, PCA is not able to make a drastic reduction in training time because training artificial neural networks involves epochs of repeated prediction and backpropagation with batch number of samples, so it is hard to expect much change in computation time from CNN unless the epoch and batch numbers or the CNN layers are changed.

Grayscale preprocessing was an overall ineffective method of preprocessing despite the greater reduction in computation time compared to PCA. The accuracies were sacrificed at a significant margin with grayscale. Grayscale reduces the dimension of samples by compressing RGB pixels into a grayscale pixel, so without proper edge-detection, it may confuse the models with less unique values, which is evident in the resulting data.

In Fig.8, the training time for the models show a more surprising outcome. SVM without preprocessing took roughly 100 seconds for testing, and SVM with PCA with the fastest time among the SVM group took 4.5 seconds for testing. However, all three CNN models uniformly took less than 2 seconds for testing despite the small deviations. Considering the testing sample size of the model (2400 images), the CNN models were significantly faster. This could be the reason why CNN models are more popular for image classification in general.

Conclusion

Both SVM and CNN proved to be reliable image classification models as they both scored accuracy above 90%, but for SVM, the computation time was exceedingly high, similar to CNN. With the discovery of CNN’s fast testing time, however, and the importance of computation speed in the nature of the application for this project, it is reasonable to conclude that CNN without preprocessing is the best for implementation in object detection model for the drone AI.

This proves that my hypothesis that CNN with PCA would show the best result. I believe the reason why neither grayscale or PCA provided better result to the model is that the CNN has internal convolutional and pooling layers that act as preprocessing steps for image processing, so, perhaps, grayscale took away the potentially crucial RGB features, PCA’s linear decomposition destructed the 2D features that convolution layer depends on for feature extractions. Thus, this project reveals that for convolutional neural networks, the unique layers provide sufficiently more effective dimensionality reduction and features extraction to secure high accuracy and computation time.

References

DATAI. (2020, October 29). *Convolutional Neural Network (CNN) tutorial*. Kaggle. Retrieved November 30, 2022, from https://www.kaggle.com/code/kanncaa1/convolutional-neural-network-cnn-tutorial#kln-67

IBM Cloud Education. (n.d.). *What are convolutional neural networks?* IBM. Retrieved November 29, 2022, from https://www.ibm.com/cloud/learn/convolutional-neural-networks

McNeal, G. (2022, March 9). *Drones and aerial surveillance: Considerations for Legislatures*. Brookings. Retrieved November 29, 2022, from https://www.brookings.edu/research/drones-and-aerial-surveillance-considerations-for-legislatures/

Ray, S. (2021, August 26). *SVM: Support Vector Machine Algorithm in machine learning*. Analytics Vidhya. Retrieved November 29, 2022, from https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/

scikit-learn. (n.d.). *1.4. Support Vector Machines*. scikit-learn. Retrieved November 29, 2022, from https://scikit-learn.org/stable/modules/svm.html

Appendix

Time and Accuracy Data.xlsx contains the test results that are not referenced in this paper.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Column1** | **SVM** | **SVM + Grayscale** | **SVM + PCA** | **CNN** | **CNN + Grayscale** | **CNN + PCA** |
| Accuracy | 0.975833333 | 0.750833333 | 0.975 | 0.931666672 | 0.805000007 | 0.782500029 |
| Data Preprocessing Time | 2.424514055 | 2.534220457 | 113.916374 | 20.52209449 | 2.391601801 | 110.2406168 |
| Model Training Time | 208.9290984 | 28.46983099 | 10.91330767 | 175.7716281 | 159.0485268 | 155.7632983 |
| Model Testing Time | 97.06690764 | 11.54910016 | 4.572765589 | 1.358365774 | 1.171864748 | 1.122995853 |
| Total Time | 308.4205201 | 40.06181741 | 129.4024472 | 198.0071383 | 162.9620564 | 267.4660032 |

Ex. Test Result Data Table from “Time and Accuracy Data.xlsx”.

Source code project is titled, “CS588\_Final\_Project” directory. The project was written in Python version 3.9 in PyCharm. Necessary packages used in the project must be installed in order to properly run the project. Each file is independent from each other, but they do depend on the root address of the dataset.

**data\_aquisition.py**: extracts images from source dataset into a designated location

**data\_preprocessing.py**: selects accessing classes and resizes images from the selected classes from 512x512 to 64x64

**svm\_model.py**: builds, trains, and tests a SVM model and outputs the model results

**svm\_grayscale\_model.py**: modify the dataset to grayscale images then builds, trains, and tests a SVM model and outputs the model results

**svm\_pca\_model.py**: applies PCA to reduce the dimension of the dataset then builds, trains, and tests a SVM model and outputs the model results

**cnn\_model.py**: builds, trains, and tests a CNN model and outputs the model results

**cnn\_grayscale\_model.py**: modify the dataset to grayscale images and builds, trains, and tests a SVM model and outputs the model results

**cnn\_pca\_model.py**: applies PCA to reduce the dimension of the dataset and then builds, trains, and tests a SVM model and outputs the model results