Classifying Quadcopter Operator Performance Using a Machine Learning Approach

Akshay Pala
Systems Design Engineering
University of Waterloo
Waterloo, Canada
apala@uwaterloo.ca

Ameya Sindgikar
Systems Design Engineering
University of Waterloo
Waterloo, Canada
arsindgikar@uwaterloo.ca

***Abstract*— Regulations for flying Unmanned Aerial Vehicles (UAVs) especially quadcopters and drones are becoming stricter in North America. Operators are required to pass written tests to ensure they know of the correct procedure and protocol before being certified to fly. However, there are no physical flying tests to gauge the abilities of operators. This study incorporates sensor readings from quadcopter takeoff and landing maneuvers to distinguish between expert and novice operators. As a result, it provides a better understanding of the skill level of these operators which can help examiners in awarding certifications. Certifying more expert operators will lead to a safer airspace without many accidents and mishaps. Several machine learning techniques have been used in this study including feature selection, feature extraction and classification. Classification accuracy has been compared across several algorithms like SVM, kNN, Decision Tree, Random Forest, Gradient Boosting, and Adaptive Boosting. Moreover, dimensionality reduction techniques like Principal Component Analysis (PCA) have been used to improve classification accuracy. It was found that dimensionality reduction significantly improves classification accuracy with the SVM classifier providing the highest accuracy (86%).**

Keywords—classification; SVM; kNN; PCA; ensemble methods; feature extraction; quadcopters; aerial vehicles

I. INTRODUCTION

The global consumer drone (quadcopter) market is forecasted to reach approximately $4 billion USD by 2024 [1]. Such acceleration in market growth can be associated with the popularity of hobbyist drone applications like aerial photography, racing and industrial applications. The increase in the drone market will inevitably lead to an increase in human and mechanical problems which can lead to accidents. Safe operation of a quadcopter is influenced by the skill level of the operator. Currently, Transport Canada requires non-recreational drone operators to obtain a Special Flight Operations Certificate (SFOC) [2]. The SFOC lacks a physical test where examiners can view the true flying abilities of potential operators. It simply requires operators to pass a theoretical test to raise awareness of safe flying. For example, it provides theoretical information on maximum altitudes, maximum distances from people and property and coordination requirements with air traffic services [2]. There is currently no physical examination in place that certifies operators based on their flying ability. A physical test would be a more effective certification method as examiners can analyze and understand the skill level of operators before awarding the license. This is analogous to obtaining a driver's license. Novice quadcopter operators who cannot make complex maneuvers or cannot takeoff and land in a stable manner will need more practice before being certified. Conversely, expert operators will be able to perform complex maneuvers and will be able to properly takeoff and land. Awarding licenses to expert operators will ensure a safe flying zone with minimal human error and ultimately minimal accidents.

This paper attempts to address a solution that can reduce the number of quadcopter accidents associated with human error. The addressed solution is associated with the flying abilities of quadcopter operators. The objective of the solution is to tune several classification algorithms so that it provides the highest accuracy in classifying between an expert and novice. Currently, takeoff and landing tests have been conducted to generate data that was used to classify between operators. Ideally, certification entities like Transport Canada and the Federal Aviation Administration (FAA) would be able to use the algorithms in this study to distinguish between expert and novice operators through several flying tests.

II. LITERATURE REVIEW

A. Aerial Vehicle Applications

There are existing quadcopters and drones that have autonomous features which minimizes human control. For example, DJI's Ground Station feature enables operators to label 16 GPS points for the quadcopter to travel to [3]. However, the quadcopters associated with this paper are less robust and lack autonomous features. These quadcopters require human control at all times.

Machine learning is currently used as a method to learn the environment around a quadcopter for autonomous navigation. BMT Defence Services in collaboration with the University of Bristol have demonstrated that machine learning principles can be used to generate a trajectory to perform a perched landing on the ground [4]. BMT used machine learning principles to essentially learn a flight controller.

A research team from the Indian Institute of Technology (IIT) has created a system that uses hierarchical structured

learning for indoor autonomous navigation of a quadcopter. The system is concerned with real time estimation of depth map of the environment from a single video frame [5]. The RGB video captured from the frontal camera of the quadcopter is used as an input to the hierarchical structured learning approach which generates a dense depth map of the environment [5]. This approach discretizes the overall depth range (0-10 meters) into multiple sets. Support Vector Machine (SVM) is then used as a method for binary classification for each internal set. The dense depth map generated from the structured learning approach is used as an input to a Convolutional Neural Network (CNN) which predicts the flight planning commands. The CNN used is trained using supervised learning and the depth images are used as inputs while the corresponding flight planning commands serve as the output. The navigation framework is implemented over the Robot Operating System (ROS) and executed over a conventional laptop, connected to the quadcopter via Wi-Fi. Video footage is captured from the quadcopter's frontal camera in real time and transferred to the HSL module via Wi-Fi which estimates the dense depth map for each individual frame. The training data set in this study was collected using a Microsoft Kinect RGB sensor which uses infrared light to compute the distance from the target objects. It consists of over 1600 RGB images of various indoor settings with their corresponding ground truth depth images representing depth between 0.5 to 10 metres. The dataset is divided into a 4:1 split for training and testing purposes with total training labels ((number of pixels in each image) × (number of images in the training set) = (640×480) × (1280) = 393216000). The structured learning approach requires around 0.18 seconds to estimate dense depth for images of resolution 640×368 while CNN takes around 0.2 seconds to generate flight planning commands. The average accuracy of dense depth estimation using structured learning is 16.89% while 46.14% accuracy is observed if the tolerance range is increased to +/−0.5 meters.

The use of neural networks has been employed with autonomous indoor navigation using Micro Aerial Vehicles (MAVs) including quadcopters. Outdoor autonomous navigation has been successful through the use of global positioning system (GPS). However, GPS is very limited in indoor settings which brings several challenges to indoor flight. An experiment done at Cornell University used a quadcopter to navigate autonomously indoors and find a specific target like a book bag [11]. The system does not use any type of sensors but relies on a camera. The approach is to train a classifier that mimics an expert pilot's choice of action. A deep learning, Convolutional Neural Network (ConvNet) was used to train the classifier with custom data. The classifier receives a purely visual input from the quadcopter and returns a flight command that best corresponds to an expert operator. The system is able to correctly find an object while performing autonomous navigation with a success rate of 70-80%.

The classifier (deep learning model of ConvNet) is trained through the actions of an expert operator who demonstrates the desired flight path given real-time images [11]. The training strategy allows the classifier to learn the most effective controller strategy with minimizing possible mistakes. The architecture of the network has five convolutional layers and three fully connected layers. The parameters are learned through fine tuning from the pre-trained CaffeNet model with the custom data used in this study. Finally, supervised learning is used to train the model.

Since the system predicts one of six commands as described below, the last fully connected layer is replaced with a layer composed of six nodes [11]. During the process of fine-tuning, the overall learning rate is decreased during training but the training rate of the newly introduced layer is increased to allow it to learn faster than the rest of the model. The study trained over 20,000 iterations with a batch size of 255 on an NVIDIA GTX 970M GPU and NVIDIA cuDNN. The overall training time took approximately six hours.

The dataset consists of images collected from seven different indoor locations at corridors or corners [11]. The quadcopter is controlled at a constant height of 1 meter using six flight commands including: move forward, right, left, spin right, left and stop. The system is tested at 5 locations which were chosen to test how well the classifier performs in terms of different objects, geometry and lighting. A success is registered if a quadcopter takes off and flies until it finds a correct target without colliding into any obstacles. The test results show a success rate of 70-80% for the first four test locations while a success rate of 60% is achieved in the last test location since it had the most unique appearance relative to the other locations.

B. External Applications

There were very limited machine learning studies done with quadcopters and with problems similar to the one in this paper. Therefore, other applications have been researched that used the same classifiers as the one in this paper. This includes Text Classification (TC) problems. TC is the problem of assigning predefined categories to free text documents [10]. A large number of statistical models have been employed for this problem including regression models, nearest neighbour classifiers and decision trees the latter two of which have been used in this paper. Cross method comparisons have been attempted with TC problems but often only for two or three methods. The results would lead to either overly general statements based on insufficient observations or provide little insight into a global comparison between a wide range of approaches.

A study done by Yiming Yang at the Carnegie Mellon University attempts to compare several classifiers to identify which ones are most suitable for TC problems [10]. The study considers the text categorization systems whose results on the various versions of the Reuters corpus have been published in the literature.

Decision trees were used to select informative words based on an information gain criterion, and predict categories of each document according to the occurrence of word combinations in the document [10]. The k-Nearest Neighbour (kNN) algorithm has also been employed. Given an arbitrary document, the system ranks its nearest neighbours among the training documents and uses the categories of the top *k* top-ranking neighbours to predict the categories of the input document.

The Reuters collection was used since it is the most commonly used collection for text classification problems [10]. It consists of over 20,000 Reuters newswire stories in the period between 1987 and 1991. The kNN classifier was applied to five versions of the Reuters collection. Pre-processing of the data was done through the benchmark retrieval system SMART. It is used for removing stop words, stemming and term weighting, where a term is a word after stemming.

Feature selection was the next step after the removal of stop words by SMART. Feature selection attempts to remove non-informative words from documents in order to improve categorization effectiveness and reduce computational power [10]. Five features were used with the kNN algorithm including information gain, mutual exclusion, chi-squared statistic, document frequency and term strength. It was found that the chi-squared statistic was the most effective as it was able to reduce the training set vocabulary from 24,858 unique words to 2485. Consequently, the category ranking performance of kNN improved from 90% to 93% in 11-point average precision. The category assignments by kNN improved from 0.81 to 0.85 in the F1 measure. The study found that kNN and Decision Trees were amongst the classifiers that exhibited the best performance on two versions out of five of the Reuters data.

These solutions may not be practical currently as quadcopters used in widespread recreational applications are controlled by a human operator. A more effective solution would have targeted flying abilities of human operators and use machine learning principles to learn and classify the operators. However, there are currently no such solutions.

III. Simulation Execution

A. Materials

The data used in this paper has been self-generated. An IMU sensor and an altimeter have been attached to the quadcopter and several tests have been done to fly the quadcopter with the sensors attached. These tests include having the operator to takeoff and land. The sensors were chosen since it best portrays the spatial dynamics of a quadcopter. The angular velocities, torques and forces created by the four rotors can be seen in Figure 1. The absolute linear position is defined in the inertial frame x, y, z axes while the attitude or the angular position is defined in the inertial frame with three Euler angles [9]. The pitch angle θ represents the rotation about the y axis and the roll angle φ represents the rotation about the x axis. Data was relayed from the sensors and have been stored on a local database. Operators who have had prior experience in flying quadcopters were labelled as experts as was the data generated from their tests. Similarly, operators who have had no experience in flying quadcopters were labelled as novices along with the data generated from their tests.

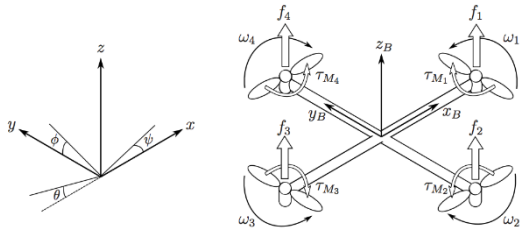

Fig. 1. Inertial and Body Frames of a Quadcopter [9].

A total of 61 novice samples and 35 expert samples were generated. Each sample had roughly 100 data points while each data point had 10 features. The consolidated size of the expert and novice raw datasets is 8085 x 10.

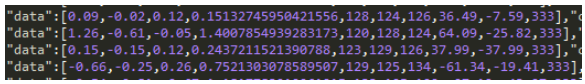

Fig. 2. Screenshot of the data that is collected from the sensors

A sample screenshot of the data collected can be seen in Figure 2. The first four values in the data array are the accelerometer components x, y, z, and magnitude of those three components. The next three are the gyroscope components x, y, z. The final three values in the data array are pitch, roll, and altitude. These values were the primary features of the raw data.

B. Methodology

Raw data was collected from tests conducted with the sensors attached to the quadcopter. The raw data included quantitative results for each one of the ten features. Employing classification algorithms on raw data may not be as accurate since there may be redundancies or noise. In fact a few of the generated data points were corrupted due to constant sensor hits from flying. Since the data collected is time variant with no set hard limit on the time to complete a single test, the data had to be made time-invariant to ensure that the classifiers do not memorize specific sensor values from the flight test. Feature extraction methods were primarily used to mitigate this problem and to avoid overfitting. Statistical features like mean, standard deviation and variance along with gradient were used as the primary features to omit any redundancies or noise in the data. In doing so, the size of the expert and novice data increased from 10 columns to 80 columns. A total of eight features were used namely mean, standard deviation, variance, max and then the gradient of each. The data increased to 80 columns by multiplying each of the eight extracted features by the 10 initial features.

Dimension reduction algorithms like PCA was then used on the extracted features in the second iteration of classification. PCA was chosen due its low noise sensitivity. The other main advantages of PCA over other algorithms are its simpler representation of the extracted data, reduction in memory and faster classification [7]. Moreover, PCA is a primary technique and is regarded as the theoretical foundation of many

dimension reduction techniques. It works by finding a linear projection that best fits a data set in the least-square sense and has been widely used for feature extraction in pattern classification due to its computational and analytical simplicity [12].

Several classification algorithms were then used on the projection matrix from PCA. Classification was also done without dimensionality reduction to compare the results with classification on the projection matrix from PCA. Algorithms like SVM, Decision Trees, Boosting, kNN, and Random Forest were used in this paper. Supervised learning techniques were used since the data was labelled. Multiple algorithms were chosen to compare the classification performance. Other algorithms like CNNs and auto-encoders could have been employed but due to the small size of the dataset they were deemed ineffective.

IV. Proposed Approach

A. Preliminary Analysis

The main instrument used to analyze data was the Python programming language. This was used since it possessed several machine learning libraries. SciKit Learn, NumPy, and SciPy were some of the machine learning libraries used to conduct classification of the data.

Preliminary data manipulation was carried out upon completing data generation from the flying tests. Preliminary data manipulation included consolidating all the expert data into a NumPy array and all novice data into another NumPy array. Feature extraction like computing the mean, standard deviation, variance, max, and gradient was done once the raw data were stored in their respective arrays. The resulting data from feature extraction for both the novice and expert arrays were then consolidated into a single array to be used for classification purposes. The dimension of this consolidated array is 95 x 80.

Labels were then generated for the expert and novice data. Expert data was assigned an arbitrary label of '0' while novice data was assigned an arbitrary label of '1.' This was necessary since the samples were already identified as either an expert or a novice during data collection.

Validation techniques are required in order for the classification algorithms to yield high accuracy and to prevent over-fitting. Several validation techniques could have been employed such as random sampling, leave one out validation, hold-out and k-fold cross validation. K-fold cross validation was chosen as the primary validation method since it minimizes the bias and variance within the training and testing data [6]. Another advantage of this method is that it matters less how the data gets divided. Every data point gets to be in a test set exactly once, and gets to be in a training set *k-1* times. The variance of the resulting estimate is reduced as k is increased. The problem with the holdout technique and one of the reasons why it was not chosen is due to the fact that the test set chosen may not be reliable and could provide bias to the results. The K-fold cross validation technique was employed in multiple iterations by using 4, 8, and 12 cross validations to compare the results. It was used in conjunction with SVM, kNN, Decision Trees, Boosting algorithms and Random Forest. The training data had to be reshaped into a NumPy array before running the classifier. Several SVM kernels were employed to compare their classification accuracies. These include the linear, RBF, sigmoid, and polynomial kernels. Generally, the RBF kernel is a satisfactory first choice [14]. This type of kernel nonlinearly maps samples into a higher dimensional space. This, unlike linear kernels, enables it to handle cases when the relation between class labels and attributes is nonlinear. Moreover, the linear kernel is a special case of RBF Keerthi and Lin (2003) since the linear kernel with a penalty parameter C has the same performance as the RBF kernel with some parameters (C, γ). The RBF kernel has fewer numerical difficulties when compared to the other kernels.

It should be noted that two simulations have been carried out. The first simulation classifies the extracted data without any dimensionality reduction techniques while the second simulation employs dimensionality reduction through PCA.

B. Principal Component Analysis

PCA was the primary dimensionality reduction method used on the extracted data in the second simulation. The array that stored all the extracted data needed to be standardized prior to running PCA. The array needed to be standardized as PCA yields a subspace that maximizes the variance along the axes. The following implementation shows the standardization of the data array.

```
from sklearn.preprocessing import StandardScaler
X_std = StandardScaler().fit_transform(allExtractedData)
```

Fig. 3. Standardization of the Feature Extracted Data \array to Conduct PCA

Mathematically, PCA performs a linear projection A that maximizes the variance through a cost function [13]. This may be done by the n principal components of the covariance matrix. Therefore, PCA solves the problem of eigenvalues as follows [13]:

$$cov(X)A = \beta A \qquad (1)$$

Where A is an orthogonal matrix that corresponds to the highest values β from $cov(X)$ (13).

The eigenvectors or the principal components determine the direction of the new feature space while the eigenvalues determine the magnitude [8]. A covariance matrix was computed to generate the eigenvalues and eigenvectors which can be seen below. The shape of this covariance matrix is 80 x 80.

The covariance matrix is calculated as follows [8]:

$$\delta_{jk} = \frac{1}{n-1}\sum_{i=1}^{n}(x_{ij} - \bar{x}_j)(x_{ik} - \bar{x}_k) \qquad (2)$$

This can be summarized by the matrix equation [8]:

$$\sum = \frac{1}{n-1}\left((X - \bar{x})^T(X - \bar{x})\right) \qquad (3)$$

Where $\bar{x}$ is the mean vector [8]:

$$\bar{x} = \frac{1}{n}\sum_{k=1}^{n} x_i \qquad (4)$$

The mean vector is a *d* dimensional vector where each value in the vector represents the sample mean of a feature column in the data [8]. Eigen decomposition was then done on the covariance matrix to generate the eigenvectors and eigenvalues. The eigenvalues generated were inspected by sorting them in descending order. There were a total of 80 eigenvalues. The first 15 can be seen in Figure 4 while the eigenvectors can be seen in Figure 5.

```
28.5278342273
11.5395902334
7.73091705125
4.3729367644
3.13686440319
2.5955113832
2.5229683075
2.20285563525
1.82645206784
1.67553349412
1.53752229377
1.32568956516
1.16401174595
0.986432009536
0.976989333203
0.810531267904
```

Fig. 4. First 15 Eigenvalues of the Covariance Matrix

```
Eigenvectors
[[ 0.13700173  0.13210233  0.0261403  ..., -0.08797019  0.15492052
   0.17561105]
 [ 0.14057337 -0.0466219   0.18169634 ...,  0.00763657  0.12661383
  -0.07096999]
 [ 0.15859445  0.07183032 -0.05918365 ..., -0.40893699 -0.02736253
   0.21784764]
 ...,
 [ 0.11872245  0.11421128  0.03168593 ..., -0.00512006 -0.00374953
  -0.0113645 ]
 [ 0.14344281 -0.07431115  0.08108685 ..., -0.00944703 -0.01520144
   0.00872374]
 [-0.03367797  0.26706278  0.00178557 ...,  0.09639401  0.14277256
  -0.00229002]]
```

Fig. 5. Eigenvectors corresponding to the Generated Eigenvalues

The lowest eigenvalues provide the least information with regards to the distribution of the data. As a result, several of these eigenvalues were discarded from the data as they started to converge to zero. Once the eigenvalues were sorted in descending order, the top 10 (*k)* values were chosen to form the projection matrix. Variance also needs to be computed to determine the number of principal components to be chosen for the new subspace. The components with the highest variance can be chosen as the basis for the new subspace. Therefore, the variance was calculated using the eigenvalues. Each of the 80 components along with their corresponding variance is visually illustrated in Figure 6.

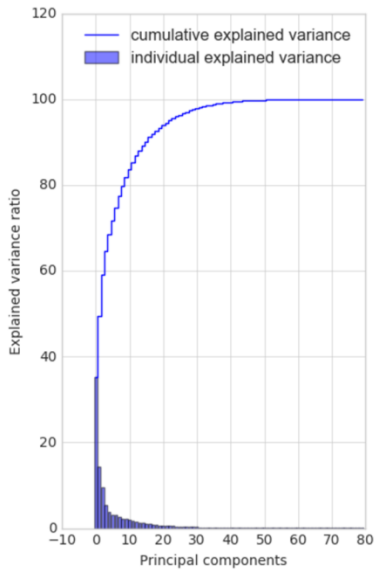

Fig. 6. Variance from each of the 80 Features

As visible, the first 10 features seem to provide the highest variance. Any subsequent features provided minimal variance and as such were not selected to generate the projection matrix. Their individual variances started to converge to zero thereby not providing any valuable information. The first component consists of approximately 37% variance while the second feature consists of approximately 17% of variance. Cumulatively, the first 10 features contributed approximately 70% of the variance in the data.

Figure 6 made it easier to select the components with the highest variance. A projection matrix was then computed once the components were selected. The 80 dimensional extracted dataset was reduced to 10 dimensions through the projection matrix (i.e. by choosing the top 10 features with the highest variance). This matrix has a dimension of 80 x 10. The projection matrix was finally transformed to the new subspace as follows (8):

$$Y = X \times W \qquad (5)$$

Where X is the feature extracted data (95 x 80) and W (80 x 10) is the projection matrix computed earlier. The subspace or the Y matrix has a size of 95 x 10. The new subspace was then used to compute the classification accuracy from SVM, kNN, Decision Trees, Boosting algorithms and Random Forest which will be discussed.

C. Algorithm Explanation

The main library used to perform classification on the data was SciKit Learn. Specifically, the *sklearn.model_selection.cross_val_score* function was used with each one of the classification algorithms. This was done twice for the non-PCA approach and the PCA approach. In the non-PCA approach, there was no dimensionality reduction. The data simply consisted of the extracted features like standard deviation, variance, mean, max and gradient. On the other hand, the data in the PCA approach was dimensionally

reduced. The major parameters for the cross validation function include:

- *Estimator*: this is the object used to fit the data. In other words, the estimator was defined as one of SVM, kNN, Decision Tree, Random Forest, Gradient Boosting, and Adaptive Boosting (AdaBoost).
- *X* (array like): this is the data to be classified or fit. In this case, *X* was defined as the consolidated feature extracted array with size 95 x 80.
- *y* (array like): this is the target variable to try to predict in the case of supervised learning. In this case, *y* was defined as the consolidated labels array.
- *cv* (integer): this determines the cross validation strategy. In other words it is an integer based parameter that in this case was defined as one of: 4, 8 and 12.

The estimator parameter discussed above was defined with correspondence to each of the classifiers used as described below.

D. kNN

The kNN algorithm works by calculating the Euclidean distance to the various neighbours for each data point [17]. This classifier performs on the premise that classification of unknown instances can be done by relating the unknown to the known based on some distance/similarity function. The main objective is that two instances far apart in the instant space that are defined by a distance function are less similar than two nearly situated instances that belong to the same class.

sklearn.neighbors.KNeighboursClassifier: this is the classifier used for the kNN implementation. There were no user defined parameters for this function.

E. SVM

SVMs are hyperplanes that separate the training data by a maximal margin [16]. All vectors that lie on one side of the hyperplane are labelled as '1' for expert or '0' for novice. The instances that lie closest to the hyperplane are called support vectors. Generally, SVMs allow projection of the original training data in space *X* to a higher dimensional feature space *F* through kernel selection. By choosing different kernel functions, the training data can be projected from *X* into spaces *F* for which hyperplanes in *F* correspond to more complex decision boundaries in the original space *X*.

sklearn.svm.SVC: this is the classifier used for the SVM implementation. The only user defined parameter for this function is the kernel type. The kernels used in this paper include linear, RBF, polynomial, and sigmoid.

F. Decision Tree

Decision Tree classifiers are a decision tool that use decision trees to classify discrete-value targets [21]. Classification is done after information is passed down the tree from the root to a leaf node. Every node in the tree includes a test attribute with the descending node being possible values for the attribute. Generally, decision trees represent a disjunction of conjunctions of constraints on the attribute values of the instances. Each path from the root node to a leaf corresponds to a conjunction of attribute tests and the tree to a disjunction of these junctions.

sklearn.tree.DecisionTreeClassifier: this is the classifier used for the Decision Tree implementation. There were no user-defined parameters for this function. The sklearn implementation of the decision tree classifier uses an optimized version of Classification and Regression Trees (CART) algorithm which is similar to C4.5 but differs since it supports numerical target variables [21]. It constructs binary trees that make use of features and thresholds that lead to the highest information gain at each node.

G. Random Forest

A Random Forest is a classifier consisting of a collection of tree structured classifiers $\{h(\boldsymbol{x}, \vartheta_k)\}$ where the ϑ_k are independent identically distributed random vectors and each tree casts a unit vote for the most popular class at input $\boldsymbol{x}$ [18].

sklearn.ensemble.RandomForestClassifer: this is the classifier used for the Random Forest implementation. The only user defined parameter for this function was n_estimators which was defined as either 10, 50, or 100. Depending on the integer definition, n_estimators signifies the number of trees in the forest. The larger the better, but also the longer it will take to compute [15].

H. Gradient Boosting

The basic idea of gradient boosting is to fit the base learner to the negative gradient of the loss function evaluated at the previous iteration (19). In every boosting iteration, the base learner directly fits the errors made in the previous iterations (19).

sklearn.ensemble.GradientBoostingClassifier: this is the classifier used for the Gradient Boosting implementation. There were no user defined parameters for this function. However, the default parameter to be noted is the 'loss' parameter. It corresponds to the loss function that needs to be optimized. The default setting is 'deviance' which refers to logistic regression for classification with probabilistic outputs.

I. Adaptive Boosting

Adaptive Boosting (AdaBoost) works by calling a weak or base learning algorithm repeatedly in a series of rounds [20]. One of the main ideas of this algorithm is to maintain a distribution or set of weights over the training set. All weights

are initially set equally, but on each round, the weights of incorrectly classified examples are increased so that the weak learner focuses on the hard examples in the training set. Both Gradient and AdaBoost algorithms boost the performance of a base learner by iteratively shifting the focus towards problematic observations that are difficult to predict [19]. With AdaBoost, however, this shift is done by up-weighting observations that were misclassified before. On the other hand, gradient boosting identifies observations by large residuals computed in previous iterations.

sklearn.ensemble.AdaBoostClassifier: this is the classifier used for the AdaBoost implementation. The only user defined parameter for this function was n_estimators which was defined as 10, 50 or 100. It controls the number of weak learners [15].

V. Results

In general, the classification accuracy from dimensionality reduction through PCA was much higher than classification without dimensionality reduction. As well, the 12-fold cross validation method generally provided the highest accuracy out of 4-fold and 8-fold cross validations. The results of the various techniques can be seen below. Algorithms like SVM, Random Forest and AdaBoost have been tuned by defining parameters that affects classification accuracy. These results can also be seen in the following sections.

It is also visible that the accuracy range with the 95% confidence interval is not affected by dimensionality reduction. Both simulations provide similar confidence interval ranges for all the algorithms.

The classification accuracy of each algorithm was calculated by finding the average of all the classification scores from each k-fold iteration. Each accuracy result is presented with its corresponding 95% confidence interval (CI) range. A screenshot of a typical result can be seen in Figure 7 below.

```
[ 1.     1.     0.625  0.75   0.625  1.     1.     1.     0.625  0.75   1.
  1.   ]
Accuracy: 0.86 (+/- 0.33)
```

Fig. 7. Screenshot of Classification Results for a Typical Simulation

In this case, a 12-fold cross validation was employed and the accuracy was computed by taking the average of all 12 iterations and providing the 95% confidence interval for that particular score. This means that the score falls within the range of 85.67% and 86.33% with a confidence of 95%.

Consult Table XII and Table XIV to see a quick summary of results using the 12-fold cross validation. The summary provides results of non-PCA and PCA simulations.

A. SVM

TABLE I. SVM Results Without PCA

Kernel	RBF	Linear	Polynomial	Sigmoid
K=4				
Accuracy	63%	67%	67%	63%
95% CI	+/- 0.02	+/- 0.11	+/- 0.35	+/- 0.02
K=8				
Accuracy	63%	74%	64%	63%
95% CI	+/- 0.03	+/- 0.45	+/- 0.42	+/- 0.03
K=12				
Accuracy	63%	74%	67%	63%
95% CI	+/- 0.05	+/- 0.47	+/- 0.50	+/- 0.05

The mean accuracies for the RBF, Linear, Polynomial, and Sigmoid kernels across all three cross validation iterations are 63%, 72%, 66%, and 63% respectively. As visible, the linear kernel seems to provide the highest classification accuracy. This suggests that the data is indeed linearly separable and that SVM is able to find a margin that perfectly separates the data.

TABLE II. SVM Results With PCA

Kernel	RBF	Linear	Polynomial	Sigmoid
K=4				
Accuracy	80%	75%	74%	55%
95% CI	+/- 0.10	+/- 0.24	+/- 0.21	+/- 0. 29
K=8				
Accuracy	83%	87%	80%	60%
95% CI	+/- 0.20	+/- 0.29	+/- 0.19	+/- 0.23
K=12				
Accuracy	86%	86%	77%	60%
95% CI	+/- 0.28	+/- 0.33	+/- 0.26	+/- 0.19

The average classification accuracies for the RBF, linear, polynomial and sigmoid kernels across all three fold iterations are 83%, 83%, 77%, and 58% respectively. This is a significantly better accuracy when compared to classifying the data without dimensionality reduction techniques. RBF and linear kernels provided the highest accuracies out of the remaining kernels. This is consistent with the first iteration as well. Therefore, it suggests that the data is linearly separable and that there is a clear distinction between the two classes of novice and expert.

B. kNN

TABLE III. kNN Results Without PCA

K=4	
Accuracy	68%
95% CI	+/- 0.15
K=8	
Accuracy	74%
95% CI	+/- 0.29
K=12	
Accuracy	72%
95% CI	+/- 0.37

As the number of folds increase, the accuracy of the kNN algorithm seems to increase.

TABLE IV. KNN RESULTS WITH PCA

K=4	
Accuracy	87%
95% CI	+/- 0.13
K=8	
Accuracy	85%
95% CI	+/- 0.19
K=12	
Accuracy	84%
95% CI	+/- 0.25

Dimensionality reduction seems to provide higher accuracy with kNN as well. The accuracy results from this experiment far exceed those from the first iteration.

C. Decision Tree

TABLE V. DECISION TREE RESULTS WITHOUT PCA

K=4	
Accuracy	63%
95% CI	+/- 0.12
K=8	
Accuracy	63%
95% CI	+/- 0.44
K=12	
Accuracy	69%
95% CI	+/- 0.44

TABLE VI. DECISION TREE RESULTS WITH PCA

K=4	
Accuracy	74%
95% CI	+/- 0.08
K=8	
Accuracy	70%
95% CI	+/- 0.23
K=12	
Accuracy	75%
95% CI	+/- 0.25

D. Random Forest

TABLE VII. RANDOM FOREST RESULTS WITHOUT PCA

#Trees	**10**	**50**	**100**
K=4			
Accuracy	76%	79%	82%
95% CI	+/- 0.32	+/- 0.26	+/- 0.24
K=8			
Accuracy	77%	83%	84%
95% CI	+/- 0.38	+/- 0.28	+/- 0.29
K=12			
Accuracy	78%	80%	84%
95% CI	+/- 0.25	+/- 0.33	+/- 0.27

TABLE VIII. RANDOM FOREST RESULTS WITH PCA

#Trees	**10**	**50**	**100**
K=4			
Accuracy	76%	84%	85%
95% CI	+/- 0.33	+/- 0.16	+/- 0.12
K=8			
Accuracy	77%	86%	84%
95% CI	+/- 0.25	+/- 0.17	+/- 0.29
K=12			
Accuracy	81%	89%	86%
95% CI	+/- 0.22	+/- 0.26	+/- 0.26

The average accuracies for 10 trees, 50 trees and 100 trees across all cross validation iterations are 77%, 81%, and 83% respectively. This is consistent with the fact that as the number of trees increase, the accuracy tends to increase as well. However, there is a trade-off between computation time and the number of trees. As the number of trees increase, the computation time will increase as well.

E. Gradient Boosting

TABLE IX. GRADIENT BOOSTING RESULTS WITHOUT PCA

K=4	
Accuracy	70%
95% CI	+/- 0.21
K=8	
Accuracy	75%
95% CI	+/- 0.36
K=12	
Accuracy	76%
95% CI	+/- 0.39

TABLE X. GRADIENT BOOSTING RESULTS WITH PCA

K=4	
Accuracy	76%
95% CI	+/- 0.16
K=8	
Accuracy	80%
95% CI	+/- 0.21
K=12	
Accuracy	82%
95% CI	+/- 0.26

F. AdaBoost

TABLE XI. ADABOOST RESULTS WITHOUT PCA

n_estimators	**10**	**50**	**100**
K=4			
Accuracy	72%	76%	76%
95% CI	+/- 0.22	+/- 0.18	+/- 0.21
K=8			
Accuracy	72%	79%	83%
95% CI	+/- 0.17	+/- 0.36	+/- 0.27
K=12			
Accuracy	72%	81%	80%
95% CI	+/- 0.43	+/- 0.25	+/- 0.35

TABLE XII. ADABOOST RESULTS WITH PCA

n_estimators	10	50	100
K=4			
Accuracy	72%	75%	72%
95% CI	+/- 0.29	+/- 0.17	+/- 0.28
K=8			
Accuracy	74%	75%	81%
95% CI	+/- 0.24	+/- 0.13	+/- 0.16
K=12			
Accuracy	79%	82%	82%
95% CI	+/- 0.26	+/- 0.19	+/- 0.26

The results show that as the number of estimators or weak learners increase, the classification seems to increase as well. This is consistent with all three fold iterations. In fact the average classification accuracies for the three iterations and for 10, 50, and 100 estimators are 72%, 79%, and 80% respectively.

Below is a summarized list of the classification results of all the algorithms conducted in both simulations. 12-fold cross validation was used to summarize the results as it provided the best accuracy amongst other folds. As visible, dimensionality reduction played a key role in improving the accuracy of the classification. SVM seems to provide the most accurate result for the PCA simulation amongst the other algorithms.

Generally, it took less computational time to perform the classification with PCA performed on the extracted data with a few exceptions. SVM, Random Forest, and AdaBoost seem to have taken considerably longer on the data that has not been dimensionally reduced.

G. Consolidated Results

TABLE XIII. CONSOLIDATED RESULTS FOR 12-FOLD WITHOUT PCA

Classifier	Accuracy (K=12)	95% CI (+/-)	Time (s)
SVM (linear kernel)	74%	0.47	11.370
kNN	72%	0.37	0.064
Decision Tree	69%	0.40	0.068
Random Forest (100 trees)	83%	0.29	5.044
Gradient Boosting	77%	0.41	1.554
AdaBoost (100 estimators)	72%	0.43	0.439

TABLE XIV. CONSOLIDATED RESULTS FOR 12-FOLD WITH PCA

Classifier	Accuracy (K=12)	95% CI (+/-)	Time (s)
SVM (linear kernel)	86%	0.33	0.089
kNN	84%	0.25	0.064
Decision Tree	74%	0.28	0.056
Random Forest (100 trees)	75%	0.34	0.549
Gradient Boosting	82%	0.26	0.593
AdaBoost (100 estimators)	82%	0.26	3.393

VI. CONCLUSION

It is clearly evident that dimensionality reduction is an important step in improving the accuracy of data. In this study, PCA was used to remove redundant parameters by selecting the highest eigenvalues or those that provided the highest variance. Classification accuracy significantly improved by running PCA on the extracted data. The linear kernel with SVM was the most accurate algorithm in classifying the data with 86% accuracy. This may be the case since this is a binary (i.e. expert/novice) classification problem. Therefore, the linear kernel SVM should have performed the best as it creates a distinct margin between the two classes.

It was also found that increasing the number of folds in cross validation methods can slightly improve the classification results. In this study, 4, 8 and 12-fold cross validations were used in conjunction with the classification algorithms. Out of the three validation folds, the 12-fold cross validation provided the highest accuracy for majority of the cases. Other techniques that allowed for better classification accuracy includes tuning the weak estimators and the number of trees for the AdaBoost and Random Forest classifiers respectively. It was found that AdaBoost performed slightly better with increasing the number of weak estimators while Random Forest performed better with more trees.

More robust machine learning methods like neural networks and autoencoders could have been employed but since the data was very limited they would not have been effective. More data should be collected going forward to improve the classification accuracy and also to implement more complex algorithms. Another limitation of this study is time domain features like mean, standard deviation, variance, max and gradient. Frequency domain could have been used to compare the classification accuracies with time domain features. Frequency-domain features such as impulse response and other frequency domain analysis methods could be compared with time-domain features.

REFERENCES

[1] Consumer Drone Market Size To Reach $4.19 Billion By 2024. (n.d.). Retrieved April 25, 2017, from http://www.grandviewresearch.com/press-release/global-consumer-drone-market

[2] Government of Canada; Transport Canada; Safety and Security Group, Civil Aviation. (2017, April 20). Getting permission to fly your drone. Retrieved April 25, 2017, from https://www.tc.gc.ca/eng/civilaviation/opssvs/getting-permission-fly-drone.html#sfoc

[3] Estes, A. C. (2014, July 02). The DJI Phantom 2 Quadcopter Is Now a Real Autonomous Drone. Retrieved April 25, 2017, from http://gizmodo.com/the-dji-phantom-2-quadcopter-is-now-a-real-autonomous-d-1599334139

[4] University of Bristol. (2017, January 11). First ever perched landing performed using machine learning algorithms. *ScienceDaily*. Retrieved April 24, 2017 from www.sciencedaily.com/releases/2017/01/170111102838.htm

[5] Duggal, V., Bipin, K., Shah, U., & Krishna, K. M. (2016). Hierarchical structured learning for indoor autonomous navigation of Quadcopter. Proceedings of the Tenth Indian Conference on Computer Vision, Graphics and Image Processing - ICVGIP '16. doi:10.1145/3009977.3009990

[6] Schneider, J. (1997, February 7). Cross Validation. Retrieved April 25, 2017, from https://www.cs.cmu.edu/~schneide/tut5/node42.html

[7] Blei, D. (2008, April 24). COS 424: Interacting with Data (Rep.). Retrieved April 25, 2017, from http://www.cs.princeton.edu/courses/archive/spr08/cos424/scribe_notes/0424.pdf

[8] Raschka, S. (2015, January 27). Principal Component Analysis in 3 Simple Steps. Retrieved April 25, 2017, from http://sebastianraschka.com/Articles/2015_pca_in_3_steps.html

[9] Luukkonen, T. (2011, August 22). Modelling and control of quadcopter. Retrieved April 25, 2017, from http://sal.aalto.fi/publications/pdf-files/eluu11_public.pdf

[10] Yang, Y. (1999). An Evaluation of Statistical Approaches to Text Categorization. Kluwer Academic Publishers. Retrieved April 25, 2017, from http://parnec.nuaa.edu.cn/xtan/IIR/readings/iryang1999.pdf

[11] Kim, D., & Chen, T. (2015, November 26). Deep Neural Network for Real-Time Autonomous Indoor Navigation. Retrieved April 25, 2017, from https://arxiv.org/pdf/1511.04668.pdf

[12] Huang, W., & Yin, H. (2009). Linear and nonlinear dimensionality reduction for face recognition. 2009 16th IEEE International Conference on Image Processing (ICIP). doi:10.1109/icip.2009.5413898

[13] Arechiga, A., Barocio, E., Ayon, J., & Garcia-Baleon, H. (2016). Comparison of Dimensionality Reduction Techniques for Clustering and Visualization of Load Profiles . 2016 IEEE PES Transmission & Distribution Conference and Exposition. Retrieved April 25, 2017.

[14] Hsu, C., Chang, C., & Lin, C. (2015, April 15). A Practical Guide to Support Vector Classification. Retrieved April 25, 2017, from http://www.cs.sfu.ca/~oschulte/teaching/726/spring11/svmguide.pdf

[15] Ensemble Methods. (n.d.). Retrieved April 25, 2017, from http://scikit-learn.org/stable/modules/ensemble.html

[16] Tong, S., & Koller, D. (2001, November 07). Support Vector Machine Active Learning with Applications to Text Classification (Rep.). Retrieved April 25, 2017, from Kluwer Academic Publishers website: http://citeseer.ist.psu.edu/viewdoc/download?doi=10.1.1.16.4036&rep=rep1&type=pdf

[17] Kataria, A., & Singh, M. (2013). A Review of Data Classification Using K-Nearest Neighbour Algorithm. International Journal of Emerging Technology and Advanced Engineering, 3(6). Retrieved April 25, 2017, from http://citeseer.ist.psu.edu/viewdoc/download?doi=10.1.1.413.3893&rep=rep1&type=pdf

[18] Breiman, L. (2001, January). Random Forests (Rep.). Retrieved April 25, 2017, from http://citeseer.ist.psu.edu/viewdoc/download?doi=10.1.1.125.5395&rep=rep1&type=pdf

[19] Mayr, A., Binder, H., Gefeller, O., & Schmid, M. (2014, November 18). The Evolution of Boosting Algorithms From Machine Learning to Statistical Modelling. Retrieved April 25, 2017, from https://arxiv.org/pdf/1403.1452.pdf

[20] Freund, Y., & Schapire, R. E. (1999, September). A Short Introduction to Boosting. Retrieved April 25, 2017, from http://citeseer.ist.psu.edu/viewdoc/download?doi=10.1.1.107.3285&rep=rep1&type=pdf

[21] Decision Tree Learning. (n.d.). Retrieved April 25, 2017, from http://www.cs.princeton.edu/courses/archive/spr07/cos424/papers/mitchell-dectrees.pdf