كلية الهندسة
College of Engineering
QATAR UNIVERSITY جامعة قطر

CMPT-623/723
Distributed and Cloud Computing
(Spring 2022)

Voice recognition

A machine learning coding project

By:

Emran Mohannad A.Altamimi (1510662)

Submitted to

Dr. Abdulziz Al-Ali

College of Engineering,
Computer Science & Engineering,
Qatar University

1. Background

The task at hand is categorizing 4 different words and separating them from noise, the system should identify when a word is pronounced and decide which word it is. To handle this challenge approaches from automated speech recognition are utilized, where features are extracted and passed to a machine learning algorithm. Speech in automatic speech recognition is classified into three categories, isolated words, connected words and continuous speech [1].

Isolated word speech recognition takes in a single word at a time. These systems have "Listen/Not-Listen" states, where they require the speaker to wait between utterances. This is the domain of the project at hand. Connected word speech recognition uses pauses between words. it uses small-to-moderate vocabulary. Like isolated word speech recognition, this set's basic speech recognition unit is the word/phrase. Continuous speech recognition deals with speech that has no pauses between words. As a result, unknown word boundary information, co-articulation, surrounding phoneme production, and speech rate affect continuous speech recognition performance. Continuous speech recognition requires special methods to identify utterance boundaries.

Many methods of signal modeling for speech recognition exist [2], and one method of representing the features in the audio signal is by using the spectrogram [3]. A spectrogram depicts the signal intensity of a signal over time at different wavelengths in a waveform. One can examine not just if there is more or less energy at 1 kHz versus 40 kHz for example, but also how energy levels change over time.Spectrogram can be used in any type of signals classification for example in [4], to classify radio signals they used CNN to extract the features from the spectrogram images and used the extracted features to train SVM.

In the interest of voice recognition, spectrograms can be used in the same way. In [5] for example, recordings of one seconds of baby cries were converted to their respective spectrogram images and classified using transfer learning, SVM, and Ensemble learning. In [6], a novel feature extraction method for sound event classification using spectrogram images was proposed. The method was tested to classify 60 sound classes in a dataset using SVM. Reference [7] presents a straightforward audio classification system that is based on the treatment of sound spectrograms as texture images. The technique is based on a previous visual classification scheme that was particularly effective in classifying textures. In [8], CNN was used to classify Uzbek spoken digits from the spectrogram images.

In this project, audio files were converted to mel-spectrogram images and then SVM and KNN were used to classify these images.

2. Dataset

In this work, the system is developed to recognize four keywords ‘Google’, ‘Alexa’, “Seven”, and “Happy”. The dataset for training the machine learning models is constructed by recording the voices manually, retrieving the voices from the existing public dataset, and data augmentation techniques. Specifically, the voice recordings of keywords 'Seven’ and ‘Happy’ are retrieved from Google V2 spoken keywords dataset, The Google V2 spoken keywords dataset is one of the popular public datasets for keyword spotting problems and it contains voice recordings of 35 keywords. More than 800 recordings of each keyword “Seven” and “Happy” are retrieved from the Google V2 spoken keyword dataset for dataset creation in the proposed work.

The voice recordings of ‘Google’ and ‘Alexa’ are manually created by making multiple people speak the keyword and record using their smartphones. About 50 people from the Arab region and Asian region are chosen for recording the voices. They recorded 4-5 varieties of the same keywords using their smartphone’s mic and transferred them to us via social media platform. Two hundred recordings of each keyword are collected using this approach. Since the recordings are only about 200 in numbers for each keyword, we employed data augmentation methods to generate more examples in each class for keywords ‘Google’ and ‘Alexa’. The data generation process for keywords ‘Google’ and ‘Alexa’ is illustrated in figure 1.

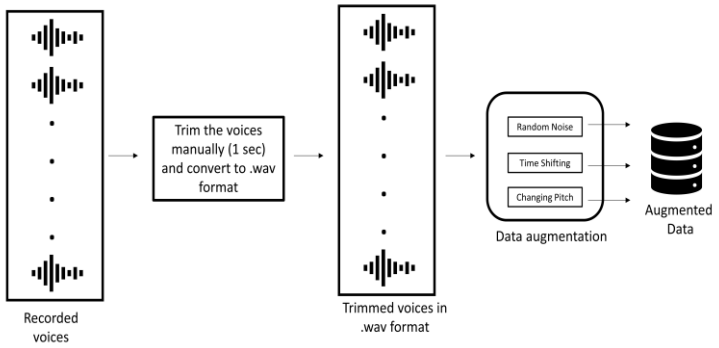

Fig 1 Dataset generation process for manually recorded voices

Initially, the recorded voices are manually trimmed to 1-second length and converted to .wav format. Then, for each trimmed voice three different operations, the addition of random noises, shifting the time, and changing the pitch are applied to generate new data. After generating the new data, the original trimmed voice recordings are added along with augmented data to make more than 800 voices for each keyword ‘Google’ and ‘Alexa’. The plot of the original recorded trimmed voice and its augmented versions for one sample from each class ‘Google’ and ‘Alexa’ are shown in figure 2.

While developing a system for voice recognition, we must consider the noises and speaking voices in the background. To make the system more robust we introduced the fifth class namely ‘Noise’ to the dataset. The recordings for the ‘Noise’ class are constructed by retrieving the ambient noise audios from MS-SNSD (Microsoft Scalable Noisy Speech Dataset). We considered ambient noises recorded from the airport with background announcement, park, office, and another undisclosed venue where the nearby people are speaking random things. All these audio recordings are divided into chunks of 1-second duration and added to the dataset. The statistics of the dataset constructed for the experiment are given in table 1. All together the constructed dataset consists of 4175 voice recordings falling under 5 classes.

Table 1 Dataset statistics

Class Name	Alexa	Google	Happy	Seven	Noise
Number of examples	832	844	880	896	723

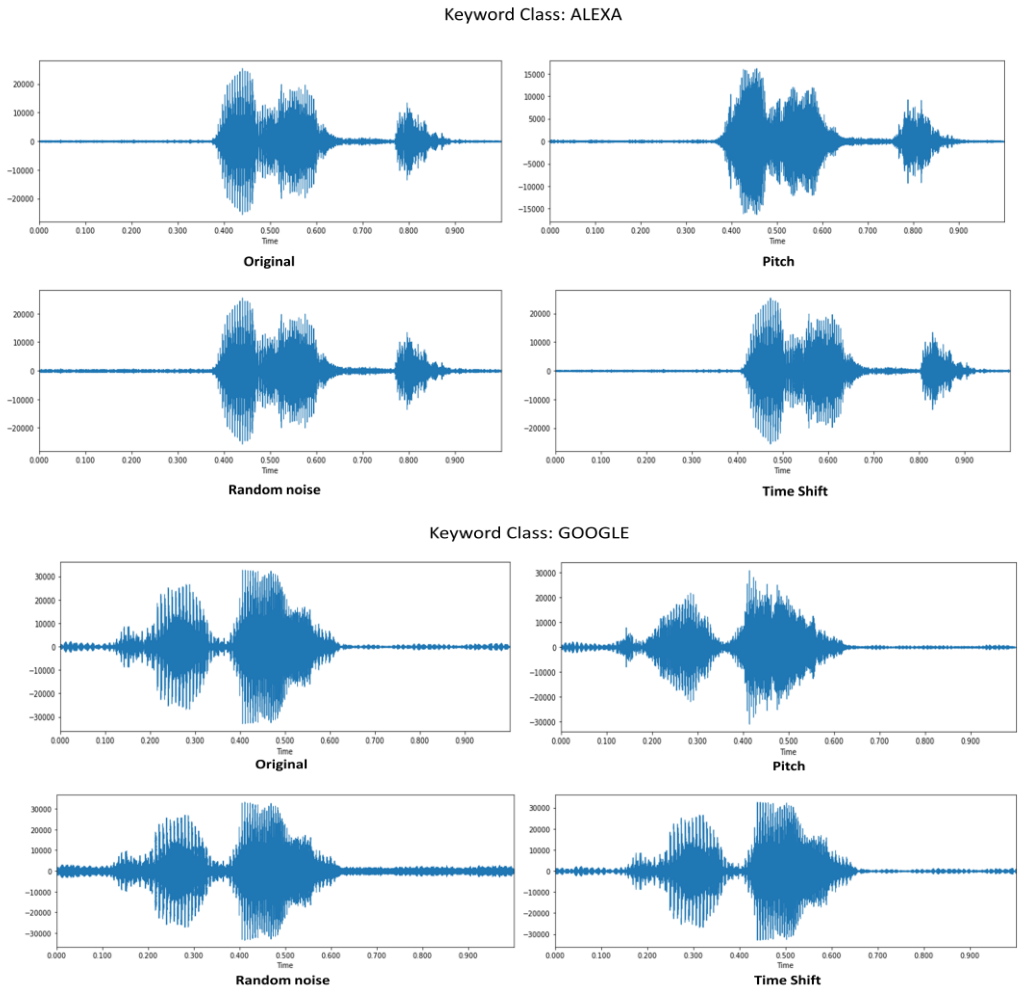

Fig 2 Original audio signal and its augmented versions

In this work, we represented the keyword recognition problem as an image classification problem. All of the voices in the dataset are preprocessed and converted to images before feeding to the classifiers. The audio signal can be represented as visual images by converting them to spectrograms. In this work, voices are converted to Mel spectrograms.

Audio is the signal where the air pressure varies over time. To convert the audios to digital forms, the samples of air pressure over time are considered. At different rates, the audio signals can be sampled. The audio signal is consists of multiple single frequency waves. And fast Fourier transform (FFT) can be applied to the audio signal to decompose it into the frequency domain. Since the audio signal varies over time, the short-time Fourier transform is applied to overlapping window segments of the audio signal to create the frequency spectrum. The FFT computed on each windows segments of the audio signals are stacked on top of each other to construct the spectrogram. In a spectrogram, the y-axis represents frequency in log scale, x-axis time, and color dimension (in decibels) as amplitude. A sample of a spectrogram is shown in figure 3.

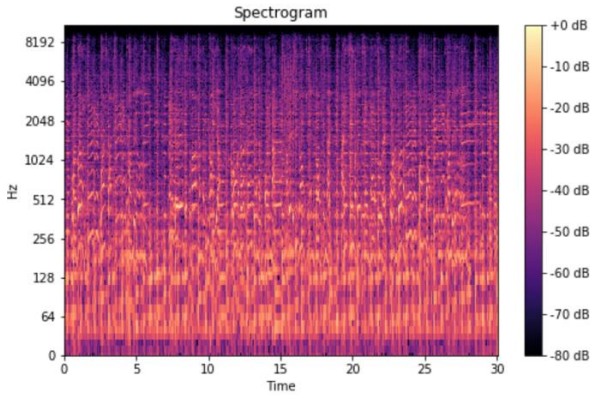

Fig 3 Sample of spectrogram

Instead of normal spectrograms, Mel spectrograms are used in this work to visualize the audio signals. To generate Mel spectrograms, the y-axis/frequency in the normal spectrogram is mapped to the Mel scale. The Mel spectrograms generated for audios from each class are shown in figure 4.

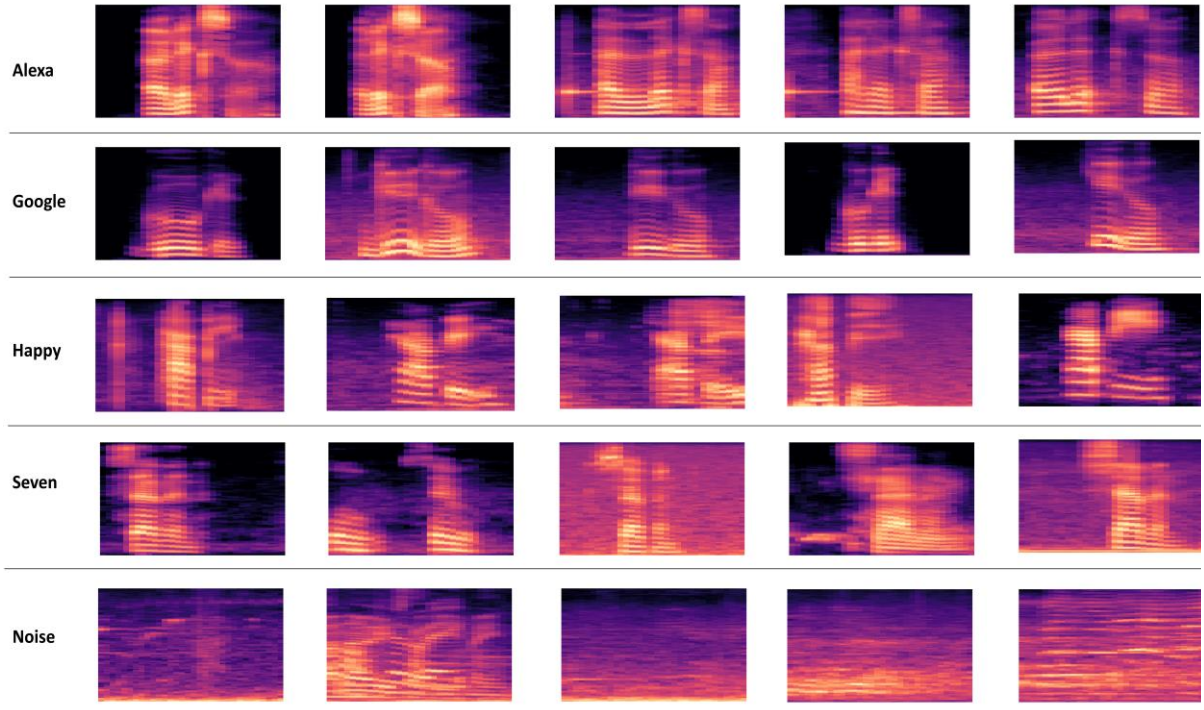

Fig 4 Samples of Mel spectrograms generated for voices from each class of the dataset

3. Methodology

The spoken keyword spotting in the constructed dataset is a multiclass classification problem. In this work, the multiclass classification problem is implemented as the binary classification problem. Two heuristic methods One-vs-Rest (OvR) and One vs One (OvO) is used in this work to implement the multiclass classification problem as the binary classification problem. SVM and KNN classifiers are implemented using One-vs-Rest (OvR) and One vs One (OvO) methods.

3.1 Classifiers

3.1.1 Support Vector Machine (SVM)

Support Vector Machines are a kind of supervised machine learning algorithms that perform regression and classification analysis on data. While SVMs are capable of regression, they are mostly employed for classification. Plotting is performed in n-dimensional space. Each feature's value is also the coordinate's value. The appropriate hyperplane for separating the two classes is then discovered.

These support vectors represent the coordinates of each observation. It is a cutting-edge technique for separating the two classes. Major hyper parameters of SVM are regularization (C) , gamma and kernel.

A decision boundary is used by SVM to separate data points that belong to distinct classes. When establishing the decision border, a soft margin should be used. SVM (soft margin meaning allowing for misclassification of certain data points) attempts to solve an optimization problem with the following objectives:

- Increase the distance between classes and decision border initially (or support vectors)
- Secondly, maximize the number of correctly categorized points in the training set.

C penalizes the algorithm for each data point that is incorrectly categorized. When the C is small, the penalty for misclassified points is minimal, and a big margin is selected at the price of a higher number of misclassifications. With a high C, SVM aims to reduce the number of misclassified samples, resulting in a decision boundary with a narrower margin.

Gamma is a non-linear SVM hyperparameter. The radial basis function is a popular non-linear kernel (RBF). The RBF gamma parameter determines the effect of a single training point. Large similarity radius leads in more points being clustered together. For high gamma values, points must be very near together to be included in the same group (or class). Large gamma values overfit concepts.

For a linear kernel, merely optimize c. However, to employ an RBF kernel, both c and gamma must be optimized. If gamma is high, c has little influence. If gamma is modest, c has a linear effect on the model. The following are typical c and gamma values:$0.0001 < \text{gamma} < 10$ & $0.1 < c < 100$

The kernel is a collection of mathematical functions used in SVM algorithms. The kernel's job is to accept data and turn it into the desired form. They can be of several varieties. Some example kernels include linear, nonlinear, polynomial, RBF, and sigmoid.

Introduce kernel functions for vectors and sequence data. RBF is the most common kernel function. It has a limited reaction along the x-axis.

Inner product between two locations in a suitable feature space. By establishing a notion of resemblance, even in high-dimensional domains.

A nonlinear hyperplane is created using RBF and Polynomial function. For complicated applications, it is recommended to employ more powerful kernels to differentiate nonlinear classes.

3.1.2 K Nearest Neighbor (KNN)

KNN is a non-parametric and simplest machine learning algorithm originally proposed by Fix and Hodges. It is a supervised machine learning algorithm that relies on the idea that similar classes are close to each other in the feature space. When KNN is presented with an example it considers K close points to that example based on a distance function and makes a prediction.

KNN algorithm for classification:

1. For a test example 'i' computes the distance from 'i' to all the training examples.
2. Find the k-nearest training examples of 'i'
3. Apply majority voting or weighted voting to determine the label of test example 'i'.

There are three hyper parameters used in KNN, the number of neighbors to be considered, the distance function, and the weights of the nearest examples in influencing the classification.

The K number of neighbors could be any integer and is usually taken to be odd to not worry about tie breaking. The weights could be either uniform or distance, where in distance further data points from the example have less influence on the decision. Euclidean distance, Manhattan distance, Minkowski distance, and Chebyshev are the popular distance metric used in the KNN algorithm.

3.2 One-vs-Rest (OVR) Approach

In the OVR approach, for each class in the dataset, a binary classifier is constructed where that class itself is treated as the positive class and remaining all other classes are considered as negative class. Suppose if there are 5 classes in the data set, then 5 binary classifiers are trained for solving the classification problem. In this work, the 5 binary classifiers are, "Alexa vs Rest", "Google vs Rest", "Happy vs Rest", "Seven vs Rest", "Noise vs Rest". The overall implementation of the OVR approach is shown in figure 5.

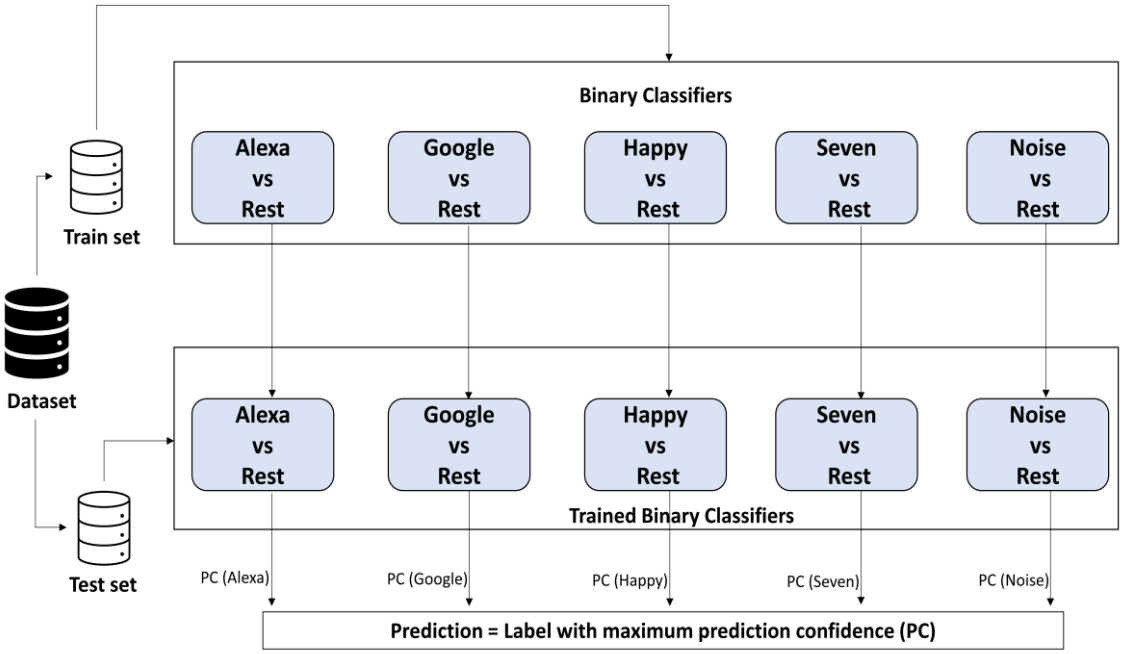

Fig 5 One vs Rest (OVR) classification approach

Initially, 5 binary classifiers (“Alexa vs Rest”, “Google vs Rest”, “Happy vs Rest”, “Seven vs Rest”, “Noise vs Rest) are constructed and trained (after tunning the hyperparameters)on the train split of the dataset. During testing, each classifier is used to predict the label of the test data and prediction confidence obtained for positive class in each classifier is recorded. The maximum value of prediction confidence is computed from it and its corresponding label is considered as the final prediction of the model. Here the dataset is the flattened arrays of spectrograms and their class label.

3.3 One-vs-One (OVO) Approach

In the OVO approach, one binary classifier is trained for each unique pair of classes in the dataset. Suppose there are 5 classes in a dataset, then in the OVO approach, 10 binary classifiers are trained where each classifier is trained on a unique pair of classes from the 5 classes. In this work, the 10 binary classifiers ‘Alexa vs Google’, ‘Alexa vs Happy’, ‘Alexa vs Seven’, ‘Alexa vs Noise’, ‘Google vs Happy’, ‘Google vs Seven’, ‘Google vs Noise’, ‘Happy vs Seven’, ‘Happy vs Noise’, and ‘Seven vs Noise’ are trained. The overall implementation of the OVO method is shown in figure 6.

Initially, the 10 classifiers are trained on the train split of the data from their respective pair of classes. During testing 10 classifiers are used to predict the label of the test data independently. After that among the predicted 10 labels, the majority one / most occurring label is selected as the final predicted label. Here the dataset is the flattened arrays of spectrograms and their class label.

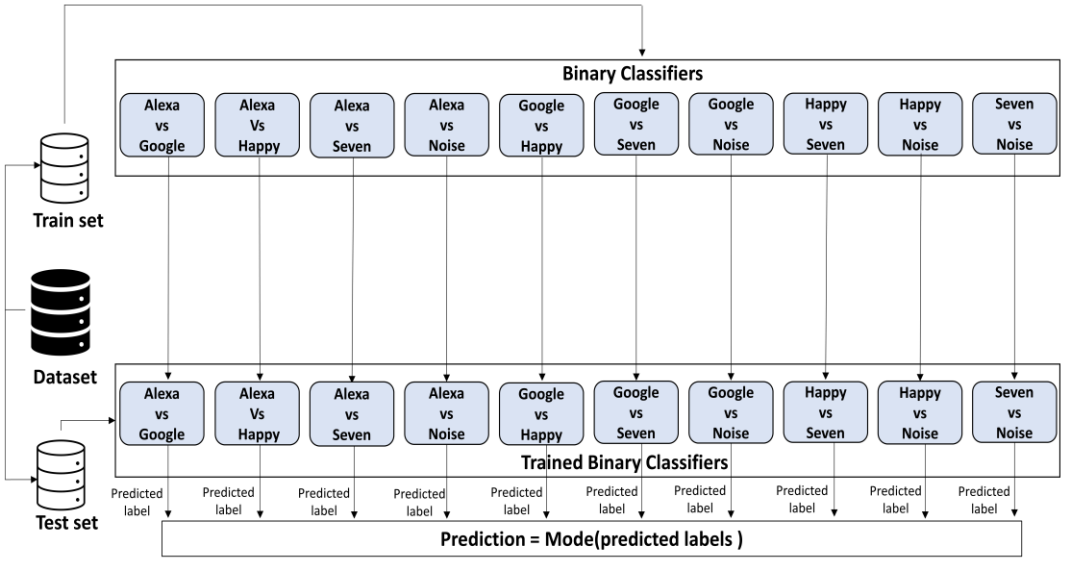

Fig 6 One vs One (OVO) classification approach

4. Implementation details and Results

This section briefly explains the implementation procedure followed in this work and discusses the obtained results. The proposed work is implemented using python language in a machine equipped with Intel Core i7 -6700 CPU @ 2.60GHZ processor, 24 GB RAM. Multiple libraries were used for the implementation. An open-source machine learning library “Scikit-learn” is used for implementing the SVM and KNN binary classifiers for the OVO approach and OVR approach. After training the binary classifiers, the OVO method and OVR method are implemented from the scratch.

For processing the audio signals (voice recordings), 'Librosa' (version = 0.8.1) is used. The spectrogram of the audio signals is generated using the Librosa library. While creating the spectrograms, the following configuration is used; Hope length = 512, length of FFT window= 2048, sampling rate = 16 Khz. The generated spectrograms are in 'RGBA' format. Before feeding them into the classifiers they are converted to grayscale format and resized into the dimension 64 × 64. The above-mentioned preprocessing is employed to reduce the computational time of training and testing of classifiers to a great extent. After converting the spectrograms to a dimension of 64 × 64 × 1 (1 represents the number of channels, here the spectrograms are in grayscale), each one is flattened to form a feature vector of size 1 × 4096. And the feature vector of size 1 × 4096 is used for training and testing the classifiers.

Four evaluation metrics: accuracy, precision, recall, f1-score are utilized to compare the performance of the classifiers. Following equations are used to calculate the accuracy, precision, recall, and f1-score.

$$Accuracy = \frac{TP+TN}{TP+FP+FN+TN} \times 100 \quad (1)$$

Accuracy is calculated as the total number of correct predictions made by the model against total predictions. Though accuracy is simple and commonly used, it is not a good measure when the data is imbalanced.

$$Precision = \frac{TP}{TP+FP} \quad (2)$$

$$Recall = \frac{TP}{TP+FN} \quad (3)$$

F1 score is the harmonic mean of recall and precision.

$$F1\ score = \frac{2*(Recall * Precision)}{(Recall + Precision)} \quad (4)$$

Here, TP: number of examples that are truly classified as positives, TN: number of examples that are truly classified as negatives, FP: number of examples that are falsely classified as positives, and FN: number of examples that are falsely classified as negative. Moreover, training time and testing time are also computed to evaluate the computational complexity.

4.1 Hyperparameters Tunning

To tune the hyperparameters, initially 80% of the dataset is taken and 10 fold stratified cross-validation with the grid search approach is implemented. The validation accuracy value is considered for ranking the hyperparameter combinations. The hyperparameter combination with maximum average validation accuracy is selected as the best hyperparameter for each classifier.

4.1.1 One vs Rest method

In the SVM-OVR method, 5 binary classifiers are implemented. Following values are considered for hyperparameters during the grid search; 'C':[0.1,1, 10, 100], 'gamma': [0.1,0.01,0.001],'kernel': ['rbf', 'poly', 'sigmoid'] .36 combinations of hyperparameters are considered during grid search with 10 fold stratified cross-validation. The results of the grid search method for the SVM-OVR binary classifier with 10 fold stratified cross-validation are shown in figure 7. The validation accuracy of each hyperparameter combination shown in figure 7 is the average of validation accuracy obtained for the respective hyperparameter combination in each fold. In figure 7, each row shows the validation accuracy of each binary classifier with different hyper-parameter combinations.

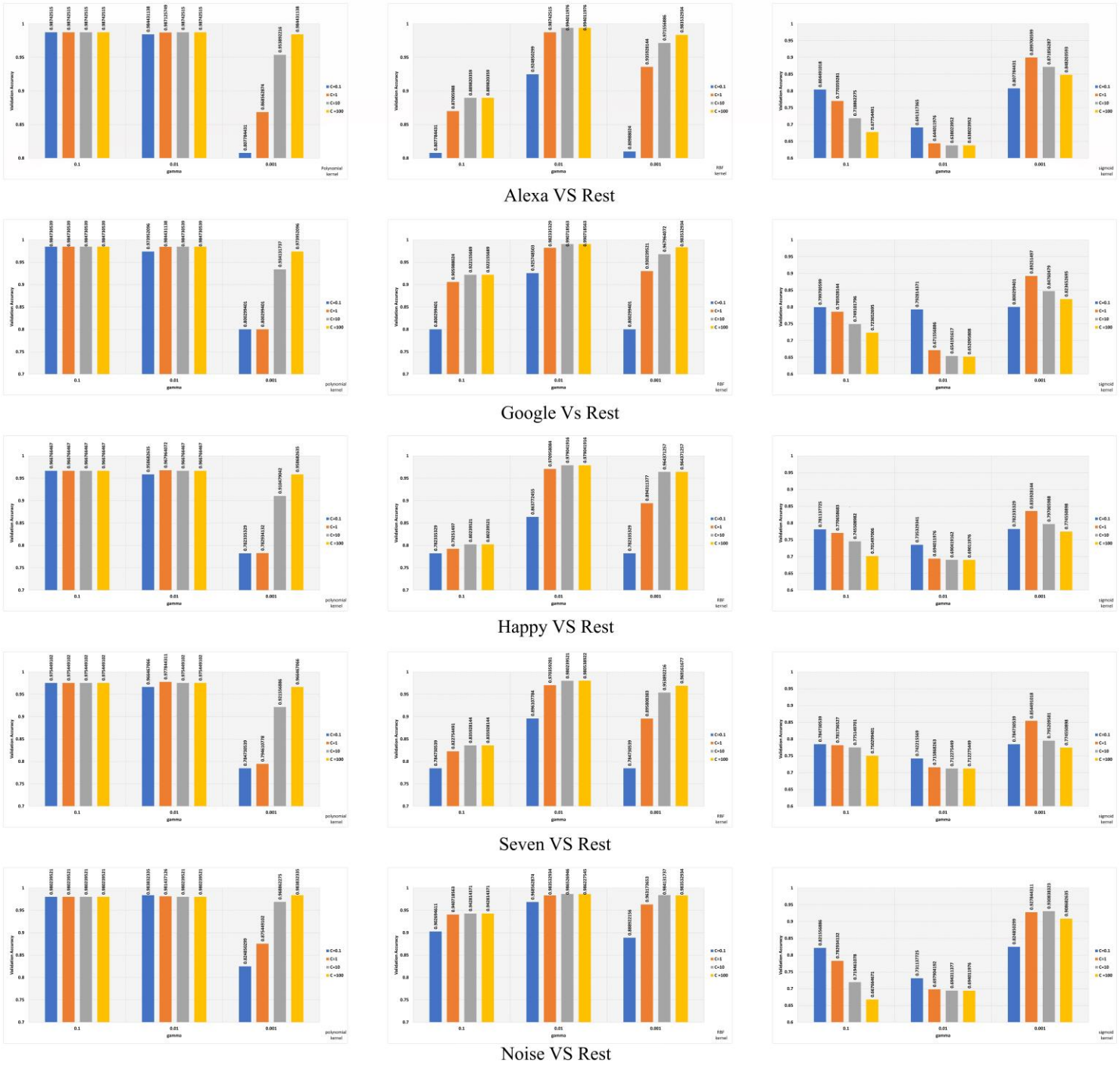

Fig 7 Results of the grid search method for the SVM-OVR binary classifier with 10 fold stratified cross-validation

The results in the first column in the figure are for the polynomial kernel SVM with different values of C and gamma. Similarly, the second column is for SVM with RBF kernel and the third column is for SVM with the sigmoid kernel. The best hyperparameters obtained in each SVM binary classifier are given in table 2.

Similarly, in the KNN-OVR method, five binary classifiers are implemented. Following values are considered for hyperparameters during the grid search; nearest neighbors = [1 to 30, odd number only], distance metric: [Euclidean, Chebyshev, Manhattan]. 45 combinations of hyperparameters are considered during grid search with 10 fold stratified cross-validation.

Table 2 Best hyper parameters obtained for binary classifiers in SVM-OVR approach

Classifier	Kernel	gamma	C
Alexa vs Rest	RBF	0.01	10
Google vs Rest	RBF	0.01	10
Happy vs Rest	RBF	0.01	10
Seven vs Rest	RBF	0.01	100
Noise vs Rest	RBF	0.01	10

The results of the grid search method for the KNN-OVR binary classifiers with 10 fold stratified cross-validation are shown in figure 8.

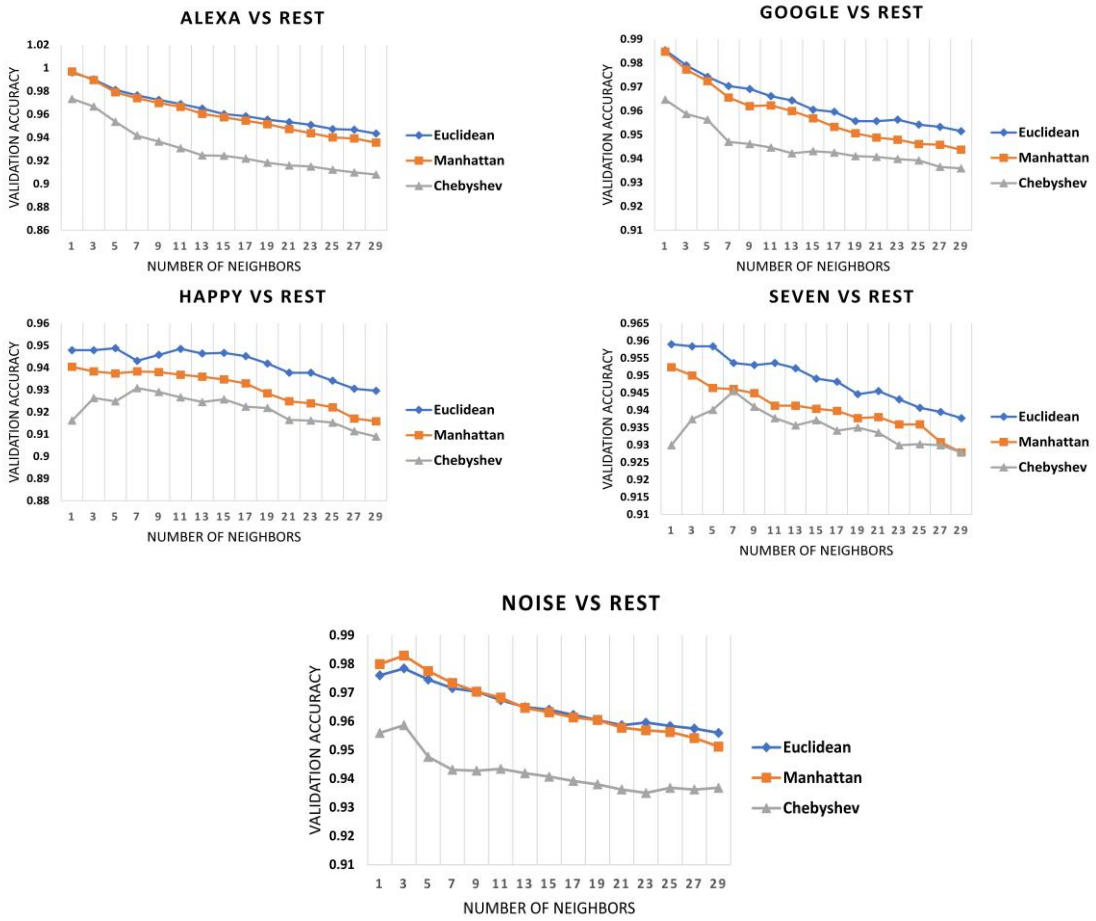

Fig 8 Results of the grid search method for the KNN-OVR binary classifiers with 10 fold stratified cross-validation

The best hyperparameter obtained in each KNN binary classifier in the OVR method is given in table 3.

Table 3 Best hyper parameters obtained for binary classifiers in KNN-OVR approach

Classifier	Number of Neighbours	Distance Metric
Alexa vs Rest	1	Manhattan
Google vs Rest	1	Euclidean
Happy vs Rest	5	Euclidean
Seven vs Rest	5	Euclidean
Noise vs Rest	3	Euclidean

4.1.2 One vs One method

In the KNN-OVO method, ten binary classifiers are implemented. Following values are considered for hyperparameters during the grid search; nearest neighbors = [1 to 30, odd number only], distance metric: [Euclidean, Chebyshev, Manhattan]. 45 combinations of hyperparameters are considered during grid search with 10 fold stratified cross-validation. The results of the grid search method for the KNN-OVO binary classifier with 10 fold stratified cross-validation are shown in figure 9 (next page). The best hyperparameter obtained in each KNN binary classifier in the OVO method is given in table 4.

Table 4 Best hyperparameters obtained for binary classifiers in the KNN-OVO approach

Classifier	Number of Neighbours	Distance Metric
Alexa vs Google	1	Manhattan
Alexa vs Happy	1	Manhattan
Alexa vs Seven	1	Euclidean
Alexa vs Noise	1	Manhattan
Google vs Happy	1	Euclidean
Google vs Seven	1	Euclidean
Google vs Noise	3	Manhattan
Seven vs Happy	11	Euclidean
Seven vs Noise	5	Manhattan
Happy vs Noise	3	Manhattan

Similarly, in the SVM-OVO method, 10 binary classifiers are implemented. Following values are considered for hyperparameters during the grid search; 'C':[0.1,1, 10, 100], 'gamma': [0.1,0.01,0.001],'kernel': ['rbf', 'poly', 'sigmoid'] .36 combinations of hyperparameters are considered during grid search with 10 fold stratified cross-validation. The best hyperparameter obtained for each binary classifier is given in table 5. The results of the grid search for SVM-OVO are not included due to space limitations.

Table 5 Best hyperparameters obtained for binary classifiers in the SVM-OVO approach

Classifier	Kernel	Gamma	C
Alexa vs Google	RBF	0.1	10
Alexa vs Happy	RBF	0.1	10
Alexa vs Seven	RBF	0.1	10
Alexa vs Noise	RBF	0.1	10
Google vs Happy	RBF	0.1	10
Google vs Seven	RBF	0.1	10
Google vs Noise	RBF	0.1	10
Seven vs Happy	RBF	0.1	10
Seven vs Noise	RBF	0.1	10
Happy vs Noise	RBF	0.1	10

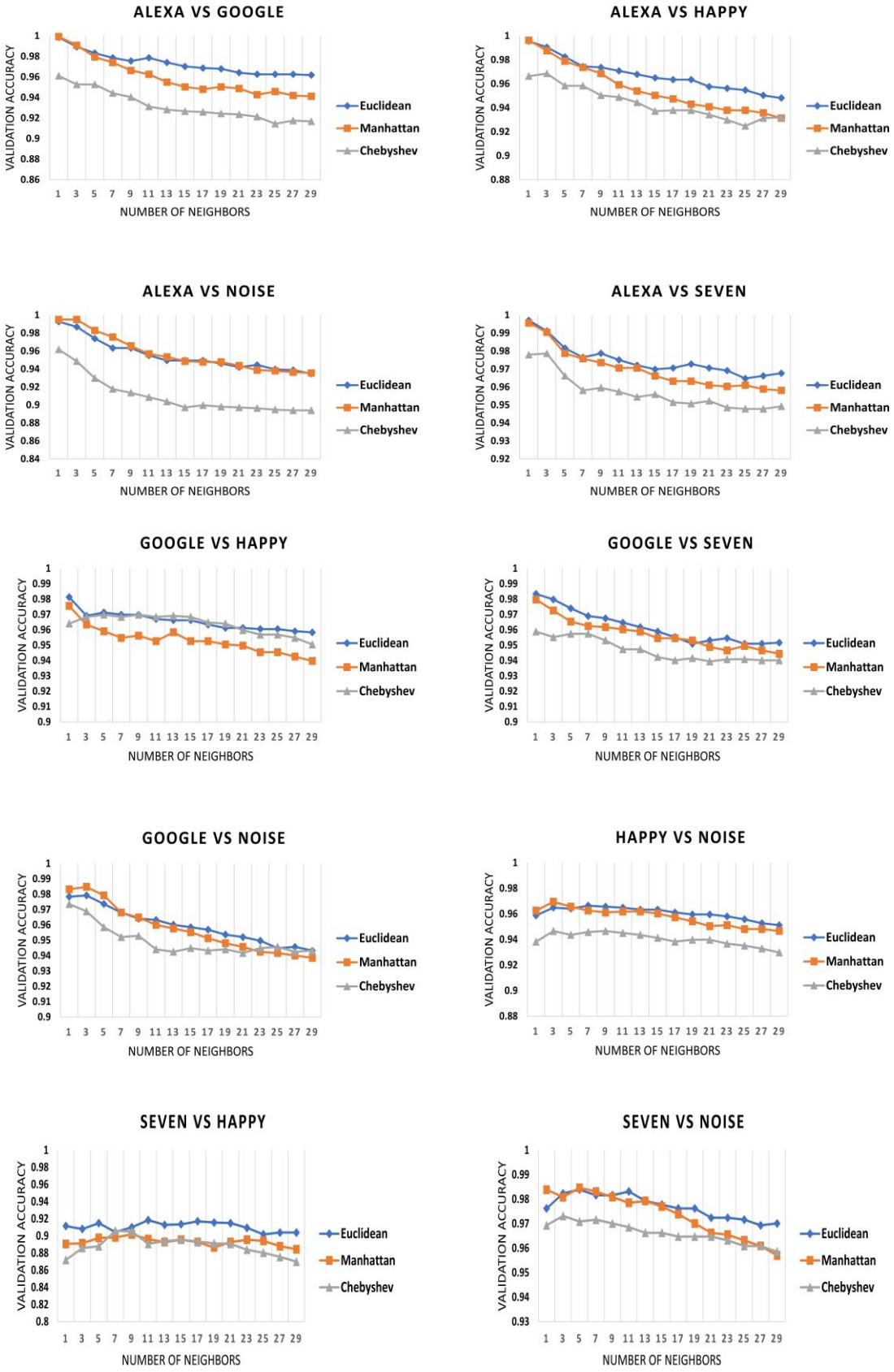

Fig 9 Results of the grid search method for the KNN-OVO binary classifiers with 10 fold stratified cross-validation

4.2 Evaluation of classifiers

After tuning the hyperparameters, the classifiers are trained with the obtained best hyperparameters in the train set and evaluated in the unseen data to analyze their performance. For this process, the whole data set is considered, and stratified 5 fold cross-validation is employed during the training and testing process. It means the whole dataset is split into 5 parts and in each iteration, 4 folds will be used for training the classifier, and the remaining fold will be used to evaluate the classifier. The classification performance of four multi-class classifiers is provided in figure 10. The precision, recall, and F1 score for the classifiers are computed by the macro averaging method.

It is clear from the figure that the SVM_OVR, support vector machine classifier implemented in one vs rest approach has achieved the best classification performance among all the classifiers.

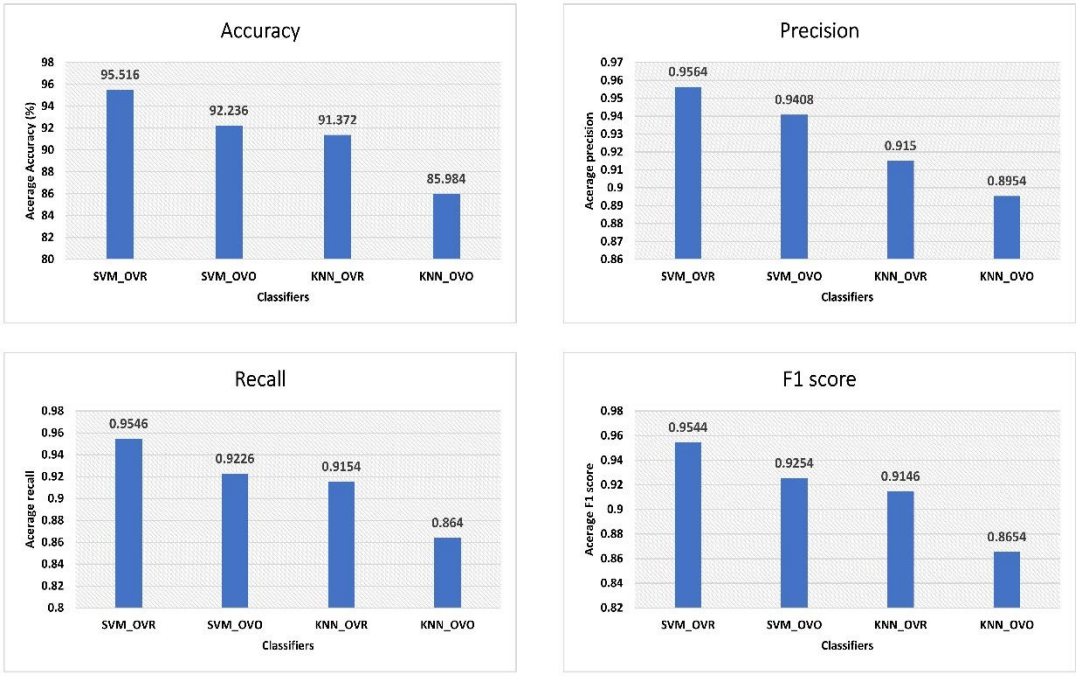

Fig 10 Evaluation performance of four multi-class classifiers in the unseen data (the accuracy, precision, recall, and f1 score are the average of values obtained in each fold)

SVM_OVR method achieved an average classification accuracy of 95.516 % (standard deviation: 1.76). The second best method, SVM_OVO, the support vector machine classifier implemented in one vs one approach achieved an average classification accuracy of 92.236 % (standard deviation: 1.64). The least average classification accuracy was obtained in KNN_OVO, the KNN classifier implemented in one vs one approach. The obtained value of accuracy in KNN_OVO is 85.984 % (standard deviation: 2.11). KNN_OVR approach achieved an average classification accuracy of 91.372% (standard deviation: 1.79). By considering the classification accuracy alone, we can conclude that, in this dataset, the SVM classifier is better compared to the KNN algorithm. Furthermore, the OVO approach is better compared to the OVR approach in this dataset.

However, for a problem like spoken keyword spotting, apart from accuracy other metrics such as precision and recall should consider for the evaluation. Because while implementing a real-

time system the recall and precision of the classifier are also important. For keyword spotting problems, recall and precision are important. Low precision indicates the number of false positives is higher and low recall indicates that the number of false negatives is higher. For keyword spotting, the recall value should be very high, otherwise, it will affect the usability of the system. Because the user has to repeat the same word multiple times until the system recognizes it correctly. On the other hand, the classifier/system should achieve a decent or good precision value. Otherwise, the system will have a high false-positive rate and the system will detect background sounds or similar sounds as the positive class. By considering these factors we believe that recall value is a little more important than the precision value in keyword spotting problems.

The recall and precision values of the SVM_OVR method are 0.9546 and 0.9564 which are almost similar and acceptable values for the given problem. Because the maximum value of precision and recall is 1 and the obtained value is more than 0.95 which shows the efficacy of the SVM_OVR classifier for recognizing the spoken keywords. Furthermore, the F1 score of the SVM_OVR classifier, 0.954 justifies the keyword recognition performance of the classifier. The precision, recall, f1 score obtained in SVM_OVO classifiers are 0.9408, 0.9226, and 0.9254 respectively. The KNN_OVO method achieved the least precision, recall, and f1 score among all the classifiers. So while analyzing the macro average precision value, we can conclude that the SVM classifier (while comparing with KNN), OVR approach (while comparing with the OVO approach) are better. The precision, recall, and f1 score value displayed in figure 10 are the average f macro average precision, macro average recall, and macro average f1 scores obtained for each classifier in each fold. To get a clearer picture, the average precision, average recall, and average f1 score value concerning each class in each classifier are given in table 6.

Table 6 precision, average recall, and average f1 score value w.r.t each class in each classifier

Classifiers	Class	Precision	Recall	F1 score
SVM_OVR	Alexa	0.971	0.988	0.979
	Google	0.970	0.965	0.967
	Seven	0.959	0.929	0.943
	Happy	0.925	0.957	0.940
	Noise	0.957	0.933	0.942
SVM_OVO	Alexa	0.733	0.995	0.844
	Google	0.991	0.941	0.965
	Seven	0.998	0.881	0.936
	Happy	0.989	0.881	0.931
	Noise	0.992	0.917	0.951
KNN_OVR	Alexa	0.973	0.992	0.982
	Google	0.931	0.981	0.955
	Seven	0.890	0.813	0.849
	Happy	0.846	0.854	0.850
	Noise	0.935	0.940	0.936
KNN_OVO	Alexa	0.637	0.977	0.771
	Google	0.937	0.919	0.928
	Seven	0.975	0.796	0.876
	Happy	0.970	0.696	0.810
	Noise	0.957	0.932	0.943

It is clear from table 6, that except SVM_OVR classifier, the rest of the classifiers is not able to achieve precision and recall values greater than 0.9 for all classes. In SVM_OVO, there are many false positives while classifying the class Alexa. So the precision value w.r.t to class Alexa has dropped to 0.733. Further in SVM_OVO, the classifier is not able to accurately classify more than 10 % examples from class seven and happy accurately. The KNN_OVR is not able to accurately classify 19% of Seven class examples and15 % of Happy class examples.

To evaluate the computational complexity of each classifier, training time and predicting time are analyzed. For prediction time, the number of predictions per second is considered.

Table 7 Computational complexity of the classifier

Classifiers	Average training time (seconds)	Average Predictions per second.
SVM_OVR	186	68.622
SVM_OVO	84.32	61.57
KNN_OVR	0.254	23.37
KNN_OVO	0.446	18.33

The average training time was very high in the SVM_OVR classifier. Compared to SVM_OVO number of binary classifiers (five in SVM_OVR, 10 in SVM_OVO) trained in SVM_OVR are less. However, the number of examples considered while training binary classifiers is different. While training binary classifiers in the SVM_OVR approach, each binary classifier has to see/train with all the data in the dataset. But in the SVM_OVO approach, each binary classifier has to see only the data from its respective pair of classes. It means even though SVM_OVO has to fit more binary classifiers, the amount of data processed by each binary classifier will be less than half of the datasets. In SVM, the training time increases with the number of examples in the train set. But when it comes to prediction, the SVM_OVR is best among all and able to predict about 68.622 examples in a second. On the other hand, KNN methods are very fast in training, because there is no actual training process happening in KNN. But during testing, it is slow and the OVR approach can predict only 23.37 examples per second, and the OVO approach can predict only 18.33 examples per second. Considering the predicting time SVM_OVR is the best classifier for real-time applications.