Int. j. inf. tecnol. (December 2022) 14(7):3647–3658
https://doi.org/10.1007/s41870-022-00992-z

ORIGINAL RESEARCH

# CNN based keyword spotting: An application for context based voiced Odia words

**Prithviraj Mohanty**[1] · **Ajit Kumar Nayak**[1]

Received: 19 February 2022 / Accepted: 28 April 2022 / Published online: 14 June 2022


**Abstract** In recent times, pre-trained convolutional neural networks (CNN) have outdid traditional automatic speech recognition systems based on Gaussian mixture model (GMM) on a variety of large vocabulary benchmarks. The proposed work presents, Odia keyword recognition using CNN approach. Different parts of a human body in spoken Odia language is considered as the keywords for recognition. MFCC feature extraction technique along with spectrogram representation of voiced keywords are considered as the input to the CNN. The projected CNN model is trained and implemented using a python frame work Keras with Tensorflow as backend. Various performance metrics are considered to compare the proposed model with a fully connected deep neural network model (DNN). A number of experiments are conducted with variation of epochs and split ratio (training-validation-testing) and results are obtained to show the accuracy, loss and other performance metrics of both the models. It is observed that the proposed model outperforms the DNN model. Further the average recognition accuracy of proposed CNN model is analyzed with other approaches like (hidden Markov model) HMM and SVM (support vector machine) model and manifests superior average recognition rate as opposed to considered state-of-the-art approaches.

**Keywords** CNN · GMM · MFCC · Spectrogram · DNN · HMM · SVM

✉ Prithviraj Mohanty
prithvirajmohanty@soa.ac.in

Ajit Kumar Nayak
ajitnayak@soa.ac.in

[1] Department of CSIT, ITER, S'O'A (Deemed to be) University, Bhubaneswar, Odisha, India

## 1 Introduction

In terms of rapid growth in technological development, researchers still lack behind to build a truly autonomous, intelligent artificial system that can interact with people in an honestly "human-like" manner. But, in numerous ways, research is being progressively more in the direction of the future consequence at a remarkably very fast stride, thanks to the ongoing advancement of ASR technology. With the rapid improvement of handheld devices, interacting with machines using speech-related technology becoming popular in numerous applications. Speech recognition has conquered the lives of human beings. It's there in smartphones, our game consoles and also in smartwatches. The Amazon Echo Dot which costs less than 53$, is a magic box that permits ordering pizza, listening to favourite music, getting a weather report or even buying a flight ticket just by talking to it with the voice. In today's world, speech-related products like: "Ok Google", Apple's "Siri", Amazon's "Alexa" and Microsoft's "Cortana" have been exploited with speech command technology which provides service for searching with voices, becoming more popular by end-users [1]. For almost all Android phones, fully hands-free services have been offered by Google [2]. In reality, keyword detection is a very powerful method to afford hands-free experience of getting data easily without explicit typing and is particularly helpful for mobile users. Speech command recognition which usually operates over tablets and smartphones, are much more essential for communicating with machine while driving or interacting with the components of a living room [26].

The research details included in this paper is structured into the following sections. Section 2 deliberates the related work considering with various approaches applied in the field of keyword spotting (KWS). A brief description of

dataset preparation and feature extraction technique used for context based voiced keyword recognition in Odia language is explained in Sect. 3. Section 4 represents the detail overview of two models used for voiced keyword recognition. Section 5 of the research work narrates the experimental setup, result analysis and discussion. The conclusion and future scope are conveyed in Sect. 6.

## 2 Related work

Recognition of voice instructions, also referred as KWS, is vital for an extensive choice of applications, from arranging voice corpuses and indexing keywords to consecutively utterance replicas closed in microcontrollers [29]. Machine learning has proven to be a powerful technique for classification problems particularly useful for KWS. Statistical HMM has been very popular among researchers to recognize voiced keywords [3–5]. In this method, HMM models are created for each of the keywords to be recognized and then a decoding algorithm is used to generate maximum likelihood matching of the keywords [6]. This technique requires each keyword is to be initialized and trained efficiently which is very computationally expensive. More recent works present some precise models that depend on large margin construction [7, 8] or application of recurrent neural networks [9]. These methods display some enhancement over the statistical HMM approach, resulting in long-latency, because they have to be processed over the entire speech for spotting the keywords [10]. The advancement of neural attention models amplified the concert on numerous tasks, specifically those connected to extended sequence to sequence models [30, 31]. These models are enormously dominant ways to appreciate what portions of the contribution are being utilized by the model to forecast the outputs. Researchers also worked on sequence discriminative frameworks and acoustic models with Connectionist Temporal Classification (CTC) which shows better performance for voice to text conversion in Mandarin and English language [32]. In [33, 34],authors have been explored the voice command identification using deep residual networks. Considering with restricted choice architectures, the superlative accuracy attained in specific tasks with spectrogram approaches was over 95% [35]. The DNN technique applied by Google [2], for keyword recognition, performs better compared to the statistical HMM system as it is simple and needs comparatively lesser computation. Considering a variant of smaller and larger vocabulary sizes, CNN models [11, 12] outperforms DNN models and fully connected single-layer neural network model [13]. The reason for CNNs perform better compared to DNN for keyword recognition is that, CNN requires fewer parameters in comparison with DNN, hence reduces memory and computational cost. Also, spectrum presentation of an audio signal which correlates with frequency and time has a better local association with CNNs, thus excepted to outperform better performance than DNNs. Therefore, CNNs result an enhanced performance and compact model size as oppose to DNNs and is said to be the contemporary method for the keyword recognition task.

As the voice command recognition method is commonly applicable on smartphones or tablets and other PDAs, hence it should have low-latency with small memory requirements and less computational complexity. Thus the inspiration of the present work is to construct a context dependent keyword recognition system that needs to be proficient in identifying predefined keywords and benefits machines to act together over the asked commands. The voice keywords are considered are ten important human body parts spoken in the Odia language. The proposed system is used to classify the predefined one-second voiced signal corresponding to its labels. A fully connected DNN model and a proposed CNN model are used to compute the possibility that the input voiced keywords fit to individual labels and finally yields the expected labels that the machine can trust for the input audio signal. The presented work is very useful and can be further engineered to run over an Android application.

## 3 Context based voiced Odia keyword recognition

In this section, we briefly discuss the data set preparation and feature extraction technique used for recognition of context based voiced keywords in Odia language.

### 3.1 Data set preparation

For the proposed context-based Odia keyword recognition using convolution neural network model (CBOKR-ConvNet), the data set has been prepared by considering Odia voices for the different parts of a human body. Ten important parts of a human body as shown in Table 1, is taken as the keywords for recognition. The data set is considered in the same way as the data set is used in Google's Tensor Flow and AIY team for keyword spotting [14]. More than 10,000 wav audio files of different speakers saying 10 important body parts as the keyword in Odia language. Each audio signal has a duration of one second and contains a single keyword. The auditory files are stored in directories based on the content and the data set is intended to help for training and testing simple machine learning models. The data is captured in a variety of forms like using recording software *audacity* or through recording apps used in smart mobile phones. Then, it was

**Table 1** Human body parts as keywords

| Keyword (Odia) | Keyword (English) | Keyword (IPA) | Keyword (Roman) | #Files |
|---|---|---|---|---|
| ମୁଣ୍ଡ | Head | muɳɖɔ | muṇḍa | 1000 |
| ବାଳ | Hair | baːlɔ | bāḷa | 1021 |
| ଆଖି | Eye | aːkʰi | ākhi | 1026 |
| ନାକ | Nose | naːkɔ | nāka | 1043 |
| କାନ | Ear | kaːnɔ | kāna | 1014 |
| ପାଟି | Mouth | paːʈi | pāṭi | 1063 |
| ହାତ | Hand | haːtɔ | hāta | 1043 |
| ଗାଧୋ | leg | goɖɔ | goḍa | 1060 |
| ପେଟ | Stomach | peʈɔ | peṭa | 1011 |
| ପିଠି | Back | piʈʰi | piṭhi | 1054 |

converted to a 16 bit little-endian PCM encoded wave files with 16K sampling rate. All these wave files are processed with *audacity* to remove noise and continuous silence parts. Mainly persons within the age group 18 to 60 years are considered to speak single keywords, rather than conventional sentences. So, all are encouraged for individual words throughout a 2-minute session. Each speaker is asked to speak 10 such keywords, with most speakers saying each of them 5 times. All together 10,335 wave files were developed for the proposed work. Table 1 shows the numbers of audio files (# Files) for each Odia keyword with equivalent English words, IPA ( International Phonetic Alphabet) and Roman words.

### 3.2 Feature extraction with MFCC

The utmost leading and prominent method used for the extraction of spectral features is MFCC. It extracts parameters from the voice just like the way humans are listening to the speech and at the same time, it deemphasizes other vital data. Also, MFCC provides a discrete cosine transform (DCT) which is a real logarithm of short-term energy being reflected on the Mel frequency scale [15]. MFCC technique has already been applied for a wide range of signal analysis tasks and is considered to be performing well in contrast to other feature extraction methods. In this work, MFCC parameter extraction technique is used to extract speech parameters for each sound file. Grounded on humanoid perceptron tests, Mel-Frequency exploration is engaged to re-compute dimension of frequency and increase more perceptually-related illustration of voice signal [16]. Fig.1 depicts the block diagram of MFCC derivation process.

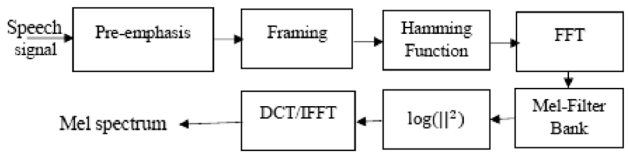


**Fig. 1** Block diagram of MFCC derivation process

To make the data ready for training using the models, the first and the foremost approach is extracting the spectrogram from the voiced keyword signals and saving them into appropriate Numpy files. Finally, the files are organized to match with corresponding classes in a CSV. Using Fourier transform, a voice signal is broken into its corresponding waves at different frequencies. The spectrogram of the voice signal represents the intensity of each frequency to time. The sampling rate of all the audio signals is considered as 16000 samples per second. It resemblances that these audio files are captured up to a frequency of 8000 Hz. MFCC are so far additional alterations on spectrograms and are intended to capture humanoid voice in a better manner. For each input frame of a signal, MFCC evaluates the cepstral coefficient (delta transformation), delta cepstral energy (delta-delta transformation) and power spectrum deviation. There is not much variance with spectrograms which in fact sphere more data in comparison to MFCC. Hence, the proposed system uses spectrogram representation. From the spectrograms, the Numpy array whose shape is (28X28) (time, frame) is resulted that describes the strength of a certain frequency from a time stamp. Now, audio files can be really treated as images and a convolutional based model that works fine over images can be applied over here. Fig. 2 shows the spectrogram representation of the sample audio signal of the voiced keyword "ମୁଣ୍ଡ"(head).

## 4 Methods for recognition of voiced Odia keywords

In this work, two deep learning methods are implemented to recognize voiced Odia keywords. These are CNN model and DNN model.

### 4.1 Convolutional neural network model

The most popular alternatives of deep learning tools extensively implemented in STT systems are CNNs. The CNN model uses weight sharing, filters and pooling which

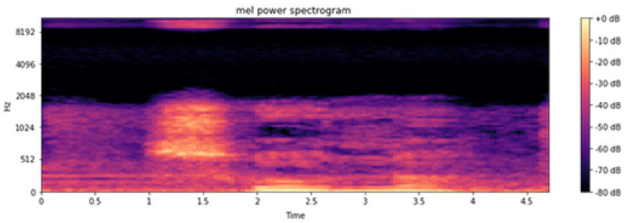


**Fig. 2** Spectrogram of the audio signal "ମୁଣ୍ଡ"(head)

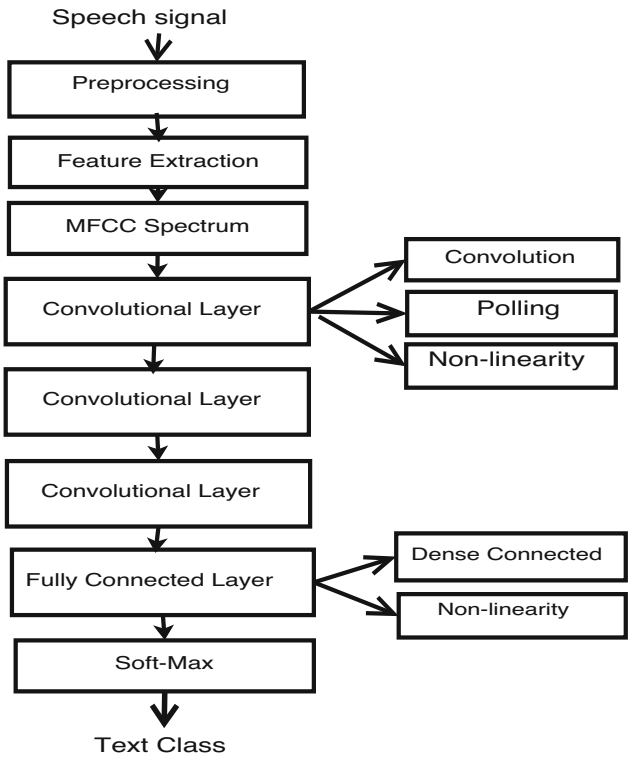


**Fig. 3** Block diagram of CNN model

makes a remarkable performance enhancement over STT systems [13]. The block diagram of the CNN model is depicted in Fig. 3. Initially, the preprocessing, feature extraction and spectrogram representation of each audio signal is considered and the Numpy files are generated. Then a sequence of convolutional layers is considered. There are three components of each convolutional layer. These are convolution, pooling and nonlinearity [17]. After convolution, it follows a fully connected layer with dense connected with some dropout. Finally, a softmax layer is considered to match with the number of classes to be generated. With enhanced architectures and improved training, CNNs now achieving high performance comparable to the performance of humans [18]. In order to process the data at a low level, CNNs use the nonlinear function. The core of the CNNs is pooling, which decreases the dimensions of the parameter map and *maxout* a widely used nonlinear function that exhibits high performance particularly for ASR tasks [19, 20]. Polling is an important technique that keeps more useful data and eliminates less significant data from the joint feature representation. It also reduces the spectral variance existing in the input voiced signal and controls the overfitting [27]. CNNs are always efficient for spoken keyword recognition task [13]. The projected CNN model architecture (CBOKR-ConvNet) has been employed with three two dimensional convolutional layers. The channel size is considered as one, hence all the

**Table 2** Architecture summary of proposed CBOKR-ConvNet model

| Layer (Category) | Output form | #Parameters |
|---|---|---|
| conv_1(ConvNet2D) | (No, 27, 27, 32) | 160 |
| conv_2(ConvNet2D) | (No, 26, 26, 48) | 6192 |
| conv_3(ConvNet2D) | (No, 25, 25, 120) | 23160 |
| max_pooling_1(MP2D) | (No, 12, 12, 120) | 0 |
| drop_1 (Dropout) | (No, 12, 12, 120) | 0 |
| flatt_1(Flatten) | (No, 17280) | 0 |
| den_1(Dense) | (No, 128) | 2211968 |
| drop_2(Dropout) | (No, 128) | 0 |
| den_2(Dense) | (No, 64) | 8256 |
| dropout_3(Dropout) | (No, 64) | 0 |
| dense_3(Dense) | (No, 10) | 650 |

layers are characterized with two dimensions (height and width) except *flatten* which is taken as only one dimension. The architecture summary of the proposed ConvNet model is described in Table 2. In that table, the cascading of layers is shown in first column and each layer's out shape obtained is presented in the second column. Convolutional layers (Conv2-D) are considered with filter sizes. Since the batch size is unchanged during the course of training and testing, the output form reflects as "No". The parameters are computed after each cascading layer and the number is represented in the third column. Finally, the total trainable parameters obtained from the model summary was found to be 2,250,386.

### 4.2 Deep neural network model

The second model implemented is a regular feed-forward fully connected neural network (FC-DNN) with 3 hidden layers and 512 hidden nodes per each hidden layer, as shown in Fig. 4. The experiment result shows that 3 hidden

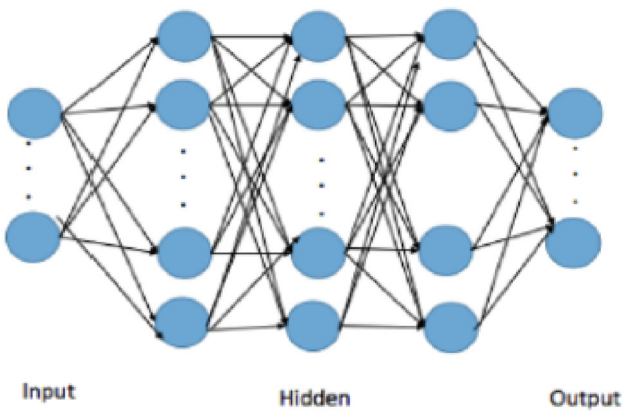


**Fig. 4** Fully-connected DNN architecture

**Table 3** Architecture summary of FC-CNN model

| Layer (Category) | Output form | #Parameters |
|---|---|---|
| den_1(Dense) | (No, 512) | 401920 |
| acti_1(Activation) | (No, 512) | 0 |
| drop_1(Dropout) | (No, 512) | 0 |
| den_2(Dense) | (No, 512) | 262656 |
| acti_2(Activation) | (No, 512) | 0 |
| drop_2(Dropout) | (No, 512) | 0 |
| den_3(Dense) | (No, 10) | 5130 |
| acti_3(Activation) | (No, 10) | 0 |

layers outperform to DNNs with 1 or 2 hidden layers. But increase in the number of hidden layers results in an increasing number of the trainable parameters which in turn consumes more computation time. Another experimental experience suggests an increase in the number of hidden nodes per layer may achieve a better result but may lead to overfitting [21]. Dropout layers with some probability are used to overcome the problem of overfitting. The rectified linear unit (ReLU) as activation function is used to reduce the weighted sum that is received from the previous layer. The architecture summary of FC-DNN is depicted in Table 3. The major layers are *dense*, *activation* and *dropout* which are cascaded sequentially. The input dimension to the first layer is 28X28. The output shape of first two dense layers are 512 (dense neurons) whereas the output form of third dense layer is 10 (dense neurons). In the first dense layer, 512 dense neurons are applied over 28X28 position with one channel, so it results in 28*28*512 (weights) +512 (bias) = 401920 parameters. Similarly the 512 neurons in the second dense layer results 512*512 (weights) + 512 (bias)= 262656 parameters. The first column shows the sequence of layers used, the second column shows the output form of each layer and the third column represents the number of parameters to be computed after each layer. Thus, the total number of trainable parameters are obtained as 669,706.

## 5 Experimental results and analysis

In this section, we briefly explain the training environment setup with different parameter settings and various performance metrics used for two models along with detail result analysis and discussion.

### 5.1 Training environment setup

For this research, the number of trainable parameters, used in both the networks are not huge enough. So, the training and testing have set up using a laptop (i5, 2.7GHz processor with 8GB RAM) enabled with Ubuntu 16.04 operating system, Python 3.6 and *Keras* as framework with TensorFlow 1.4.0 as backend. The batch size for both networks is the same and is considered to be 30. In each epoch, randomly 30 training samples will be chosen, which generally breaks down the correlation between the samples, hence helping the network to learn efficiently. For initialization of weights, randomly weights are adjusted from a normal distribution with a mean value of zero. The weight values are considered as close to zero, but not exactly zero. The learning rate is the most vital hyperparameter while forming the neural network. For both the models, the learning rate is considered to be 0.001. Cross-entropy is usually applied in machine learning as a loss function. It is a logarithmic function applied during training to eliminate small errors. With some optimization, the value can rise to 0.001. Dropout method is applied as the regularization technique, which reduces the overfitting of the network models by avoiding composite co-adaptations on training data [22]. The computational cost of dropout regularization is less in the case of a CNN. It works by eliminating or dropping out some nodes in a probabilistic manner from the input layer. Using dropout technique, a huge number of networks through dissimilar network structures can be simulated which makes the model more robust for the inputs [28]. In the model implementation, the drop probability is considered as 0.5, so that it can maximize more network structure.

### 5.2 Performance metrics

The next phase after executing a machine learning algorithm is to observe how operative is the model centred on metrics and datasets. Diverse performance metrics are applied to analyse the performance of the machine learning models. For this research, accuracy, confusion matrix, precision, recall, F-score and Cohen's kappa are considered as the performance metrics. 10 important body parts of a human being are considered as the recognition of 10 labels or classes.

***Accuracy and loss***

Loss is well-defined as the change between the predicted value and the actual true value by the model. Accuracy is also a metric used to compute the performance of the model. It can be defined as:

$$Accuracy = \frac{\#Correct\ Prediction}{\#Total\ Predictions} \quad (1)$$

Most of the time it has been perceived that as accuracy rises with a loss in reduction. Loss (cross-entropy) and accuracy are used to measure two dissimilar things. Cross-entropy loss grants inferior loss to guesses which are nearer

**Table 4** Comparison between testing accuracy

| #Epochs | Split Ratio(%) | **Accuracy(%) CBOKR-ConvNet** | FC-DNN |
|---|---|---|---|
| 20 | 70-15-15 | 88.03 | 83.19 |
| | 80-10-10 | 88.72 | 83.79 |
| | 90-05-05 | 88.78 | 83.46 |
| 50 | 70-15-15 | 89.61 | 85.03 |
| | 80-10-10 | 89.59 | 85.82 |
| | 90-05-05 | 89.16 | 85.59 |
| 100 | 70-15-15 | 89.48 | 84.39 |
| | 80-10-10 | 88.77 | 84.52 |
| | 90-05-05 | 89.01 | 85.20 |
| **200** | 70-15-15 | 89.37 | 85.18 |
| | **80-10-10** | **91.23** | **87.15** |
| | 90-05-05 | 90.13 | 86.36 |

The bold values signifies that in epoch size of 200 with split ratio 80-10-10, accuracy of both the models are better as compared to other epochs and split ratios.

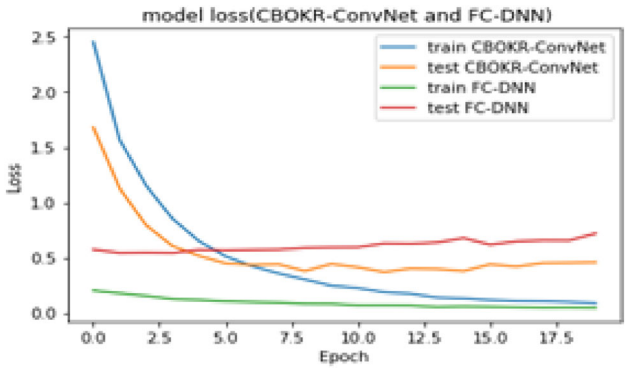


**Fig. 5** Model loss with epoch size 20

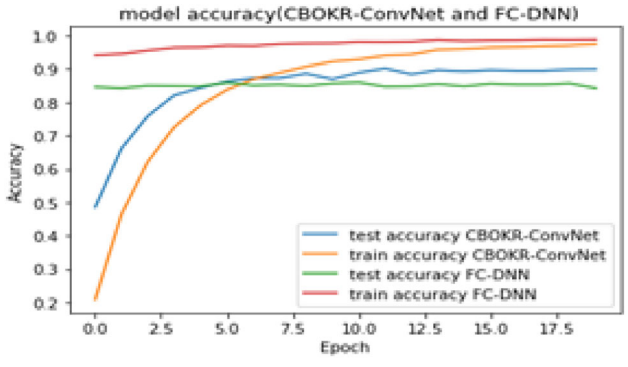


**Fig. 6** Model accuracy with epoch size 20

to the class label. Accuracy, on the other side, is a binary 1 or 0 for a specific sample. So, a loss is an uninterrupted variable i.e. it is finest when predictions are nearer to 1 (for true labels) and close to 0 (for false ones). The accuracy is computed over test-set and plotted the accuracy trend (fitting) of 2 different models considering different epochs with variation of split ratio.

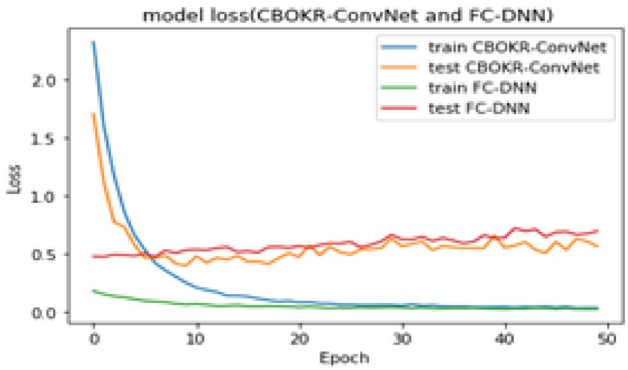


**Fig. 7** Model loss with epoch size 50

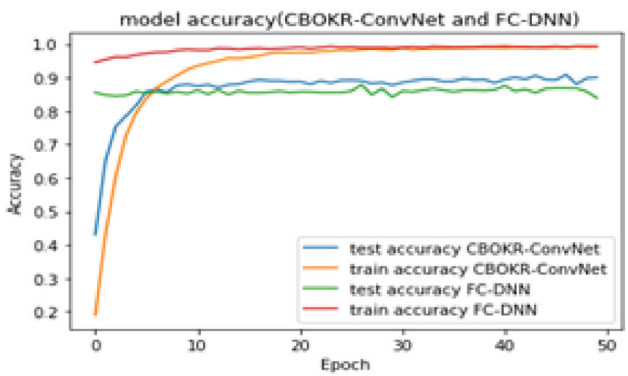


**Fig. 8** Model accuracy with epoch size 50

### *Confusion matrix*

It is a matrix that reflects the performance summary of a classification problem (binary or multiclass). Classification accuracy only can be confusing if an unequal number of

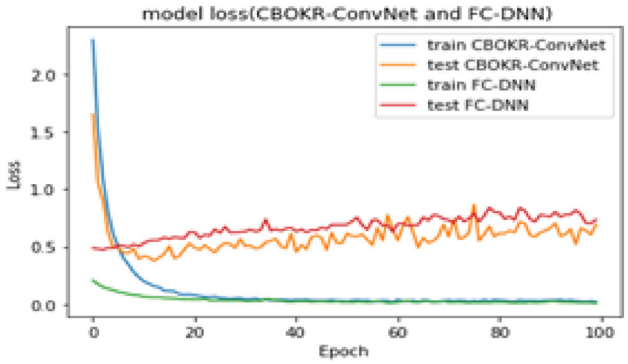


**Fig. 9** Model loss with epoch size 100

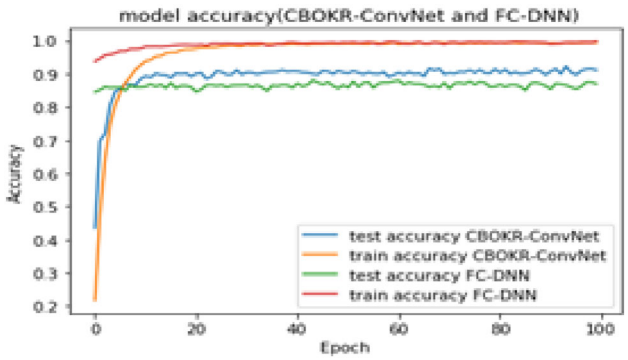


**Fig. 10** Model accuracy with epoch size 100

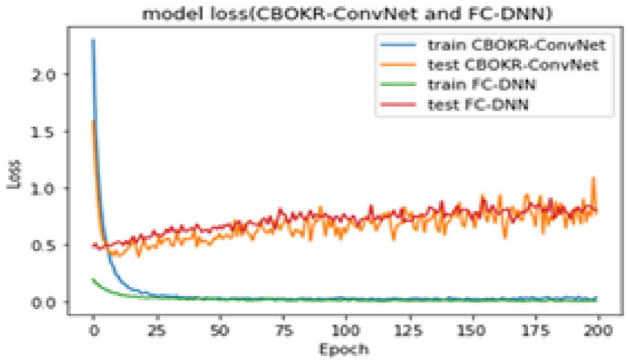


**Fig. 11** Model loss with epoch size 200

observations is present in each class or have more than two classes in the dataset. Computing a confusion matrix can give a better idea of what the classification model is getting right and what types of errors it is making.

***Precision, recall, F-score and Cohen's kappa***

***Precision*** is also called as positive predictive value. It can be computed as available of all the positive classes that have anticipated correctly, how many are actually positive (True Positive). It should ideally be 1(high) for a good classifier. It is 1 when there is no FP (False Positive).

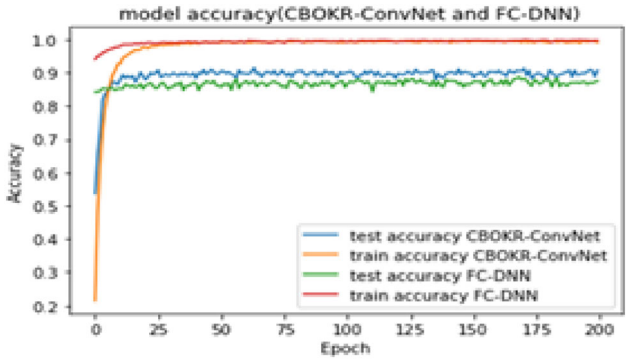


**Fig. 12** Model accuracy with epoch size 200

$$Precision = TP/(TP + FP) \tag{2}$$

***Recall*** is also called as sensitivity or true positive rate. For a better classifier, it should be 1. It can be computed as out of all the positive classes' i.e. addition of True Positive and False Negative (FN), how much we predicted correctly.

$$Recall = TP/(TP + FN) \tag{3}$$

It is hard to relate two models with less precision value and more recall value or vice versa. In order to compare two models, ***F-score*** value also used. F-score supports to compute recall and precision together. It usages harmonic mean instead of arithmetic mean by exhausting the extreme values more. F score value will be 1 when both precision and recall values are 1.

$$F - score = 2 * \frac{(Precision * Recall)}{(Precision + Recall)} \tag{4}$$

***Cohen's Kappa*** measurement is a very valuable performance metric. In a multi-class classification problem, performance metrics such as accuracy, or precision/recall do not deliver the whole representation of the performance of the proposed classifier. This statistic is a very decent measure that can grip very well in both multi-class and imbalanced class problems. It is represented in as:

$$C_k = \frac{\rho_o - \rho_e}{1 - \rho_e} = 1 - \frac{1 - \rho_o}{1 - \rho_e} \tag{5}$$

where $\rho_o$ is the obtained agreement, and $\rho_e$ is the expected agreement. This essentially tells about how ample better the proposed classifier is performing over the performance of a classifier that predicts at random rendering to the occurrence of each class. This value is always $\leq 1$. Values of $\leq 0$ identify that the classifier is unusable. No concrete way is there to understand its importance. [23] afford a method to describe the values. Adhering to their system a value $< 0$ is representing no covenant, 0.0–0.2 as minor,

**Table 5** Confusion matrix obtained from CBOKR-ConvNet model

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 90 | 1 | 2 | 1 | 0 | 2 | 0 | 1 | 1 | 0 |
| 0 | 114 | 0 | 1 | 1 | 3 | 2 | 0 | 0 | 1 |
| 3 | 0 | 95 | 0 | 0 | 0 | 2 | 1 | 1 | 1 |
| 3 | 1 | 1 | 94 | 0 | 0 | 0 | 2 | 1 | 5 |
| 2 | 1 | 2 | 1 | 89 | 0 | 3 | 1 | 1 | 1 |
| 5 | 2 | 2 | 2 | 1 | 80 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 1 | 0 | 0 | 100 | 0 | 1 | 0 |
| 2 | 0 | 0 | 3 | 0 | 0 | 0 | 78 | 0 | 2 |
| 1 | 0 | 0 | 0 | 0 | 1 | 2 | 3 | 89 | 0 |
| 0 | 0 | 6 | 3 | 0 | 2 | 0 | 1 | 0 | 103 |

**Table 6** Confusion matrix obtained from FC-DNN Model

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 84 | 0 | 3 | 4 | 0 | 2 | 0 | 2 | 2 | 1 |
| 0 | 103 | 0 | 2 | 0 | 4 | 3 | 2 | 3 | 5 |
| 2 | 3 | 81 | 7 | 2 | 2 | 1 | 1 | 0 | 14 |
| 5 | 0 | 3 | 88 | 1 | 0 | 0 | 2 | 0 | 4 |
| 2 | 1 | 1 | 0 | 90 | 1 | 1 | 2 | 0 | 3 |
| 8 | 1 | 2 | 1 | 1 | 71 | 5 | 0 | 4 | 2 |
| 0 | 0 | 0 | 0 | 0 | 2 | 101 | 0 | 1 | 0 |
| 2 | 1 | 0 | 1 | 0 | 0 | 0 | 79 | 0 | 2 |
| 2 | 1 | 2 | 2 | 1 | 1 | 4 | 5 | 79 | 1 |
| 1 | 0 | 11 | 6 | 1 | 3 | 0 | 1 | 0 | 92 |

0.21–0.4 as reasonable, 0.41–0.6 as modest, 0.61–0.8 as considerable, and 0.81–1.0 as nearly faultless agreement.

***ROC and AUC***

Receiver operating characteristic curve (ROC) is a realistic plot that demonstrates the analytical capability of both binary and multi-class classification systems as its perception inception is different. One-vs- all approach is used to find the ROC of a multi-label classifier. The area under the curve (AUC) is the proportion of the area that is under the ROC curve, ranging between 0 1. The ROC approach is used to visualize the performance of a classifier, and AUC is the alone number that is used to encapsulate a classifier's performance by measuring the ranking between two class labels. The higher is the value, the better is the classifier. Using the predictable likelihoods and ground correct labels, two pairs of data arrays are necessary to figure the ROC curve. These are false positive rates and true positive rates for each possible threshold value. The ROC curve pictures the excellence of the ranker or probabilistic model on a test data set, without constraining to a classification threshold.

### 5.3 Result analysis and discussion

We have considered a total of 10,335 *.wav* samples for experimentation over two models CBOKR-ConvNet and FC-DNN. Along with, multiple epoch sizes with the variation of training, validation and testing samples are considered with k-fold cross validation, where k=3. After training, a single prediction on the test set is preferred to obtain the test accuracy. Table 4 represents the test accuracy of both the models with different epochs along with the variation on split ratio over the data set. With an epoch size of 200 and a split ratio (%) of 80-10-10, the proposed model obtains a higher testing accuracy of 91.23% compared to the FC-CNN model whose accuracy is 87.15%. Also, it has been observed that the CBOKR-ConvNet model has better accuracy compared to the FC-DNN model with different epochs with the variation of split ratio. Figs. 5, 6, 7, 8, 9, 10, 11, and 12 shows the model loss and accuracy both in the training and testing phase for the two models with epoch size of 20,50,100 and 200 respectively with training, validation and testing ratio of (80-10-10)%. It has observed from the figures that loss and accuracy in training for both the models are almost the same but the CBOKR-ConvNet model has better test accuracy compared to the FC-DNN model.

Tables 5 and 6 shows the confusion matrix obtained from the model CBOKR-ConvNet and FC-DNN considering epoch size= 200, batch size=30 with a split ratio of %=80-10-10. It has perceived that, the diagonal values present in Table 5 are greater than the diagonal values of Table 6. More diagonal values in Table 5 suggest that CBOKR-ConvNet has better classification performance compared to the FC-DNN model. Table 7 represents the comparison of different performance metrics for both the models with different epochs with variation in split ratio considered for the dataset. It has observed from the table that, the proposed CBOKR-ConvNet model has greater or equal values compared to the FC-CNN model with different variations of epochs and split ratio considering the performance metrics like precision, recall, F-score and Cohen's Kappa.

The precision recall curve for both the models with average precision (AP) score value is represented in Fig.13 and Fig. 14 (Epoch=200 and split ratio=80-10-10). It is understood that, the proposed CBOKR-ConvNet outperforms better (higher AP value) compared to the FC-DNN model. Fig. 15 shows the average ROC curve with AUC value of the proposed CBOKR-ConvNet model on epoch =200 with split ratio=80-10-10. Fig. 16 depicts the ROC curve for each of the labels represented as one class with corresponding AUC values. More extension of the AUC values of the multiclass is represented in Fig. 17. Similarly Fig. 18 shows the average ROC curve with AUC value for FC-DNN model. Also for FC-DNN, Fig. 19 and Fig. 20 represents the ROC curve with equivalent AUC value for each of the class and its extension respectively. A comparison of the ROC curve for both models is depicted in Fig. 21. Individual AUC values obtained for each of the

**Table 7** Comparison of different performance metrics for both the models

| #Epochs | Split Ratio(%) | **Precision(%)** | | **Recall(%)** | | **F-score(%)** | | **Cohen's Kappa(%)** | |
|---|---|---|---|---|---|---|---|---|---|
| | | CBOKR-ConvNet | FC-DNN | CBOKR-ConvNet | FC-DNN | CBOKR-ConvNet | FC-DNN | CBOKR-ConvNet | FC-DNN |
| 20 | 70-15-15 | 88.61 | 83.42 | 83.19 | 83.19 | 88.06 | 83.22 | 81.33 | 81.32 |
| | 80-10-10 | 88.90 | 84.12 | 83.79 | 83.79 | 88.76 | 83.83 | 81.98 | 81.98 |
| | 90-05-05 | 89.29 | 84.07 | 83.47 | 83.46 | 88.80 | 83.45 | 81.06 | 81.46 |
| 50 | 70-15-15 | 89.69 | 85.19 | 85.03 | 85.03 | 89.61 | 85.07 | 83.37 | 83.37 |
| | 80-10-10 | 89.67 | 85.98 | 85.83 | 85.82 | 89.60 | 85.85 | 84.24 | 84.24 |
| | 90-05-05 | 89.44 | 85.89 | 85.59 | 85.59 | 89.22 | 85.64 | 83.97 | 93.97 |
| 100 | 70-15-15 | 89.63 | 84.62 | 84.39 | 84.39 | 89.49 | 84.37 | 82.65 | 82.65 |
| | 80-10-10 | 89.11 | 85.07 | 84.51 | 84.52 | 88.84 | 84.65 | 82.79 | 82.79 |
| | 90-05-05 | 91.30 | 85.40 | 85.20 | 85.20 | 91.05 | 85.19 | 83.53 | 83.53 |
| **200** | 70-15-15 | 90.38 | 86.45 | 87.45 | 87.42 | 90.36 | 90.04 | 84.19 | 84.18 |
| | **80-10-10** | **91.08** | **87.06** | **88.12** | **87.34** | **91.03** | **87.86** | **85.87** | **85.87** |
| | 90-05-05 | 90.28 | 86.51 | 86.36 | 86.36 | 90.15 | 86.37 | 84.82 | 84.83 |

The bold values signifies that in epoch size of 200 with split ratio 80-10-10, the performance metrics (precision, recall, F-score and Cohen's kappa) of both the models have better results as compared to other epochs and split ratios.

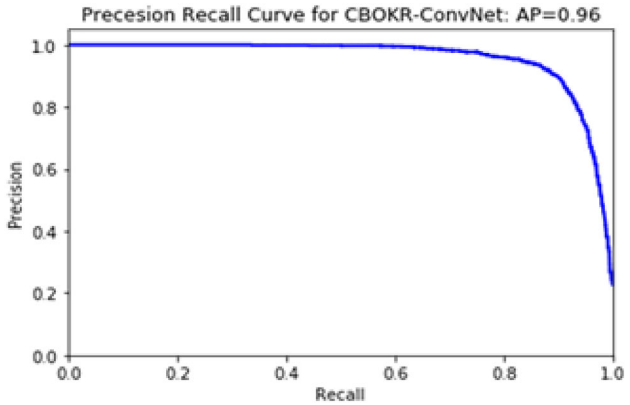


**Fig. 13** Precision recall curve with AP values for CBOKR-ConvNet Model

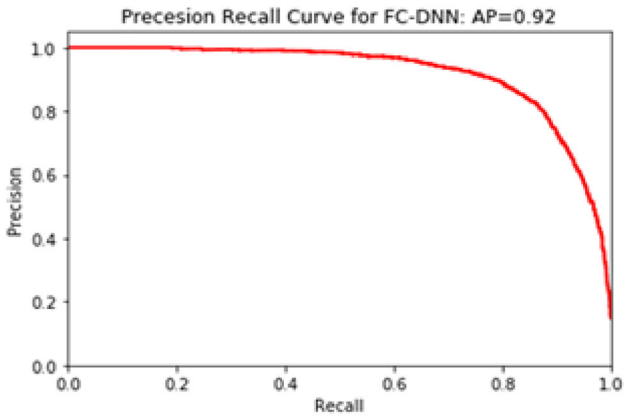


**Fig. 14** Precision recall curve with AP values for CBOKR-ConvNet Model

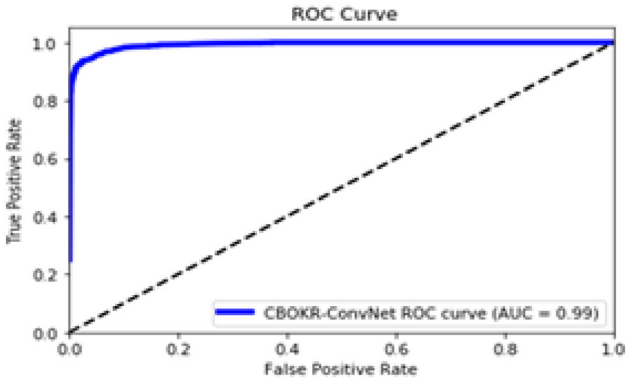


**Fig. 15** ROC curve of CBOKR-ConvNet model

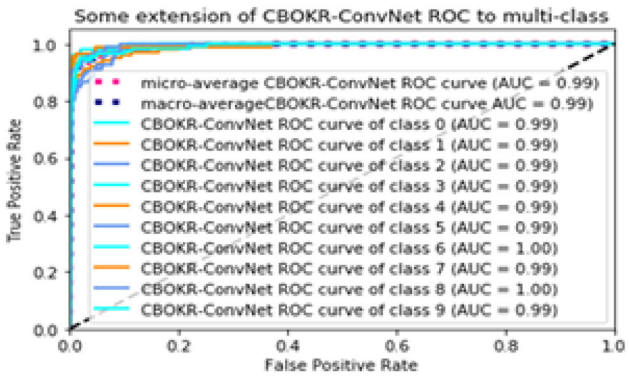


**Fig. 16** Extension of ROC curve to multi-class for CBOKR-ConvNet Model

classes in the proposed CBOKR-ConvNet have greater or equal to the individual AUC values obtained for each of the classes in the FN-DNN model. Also, the average AUC

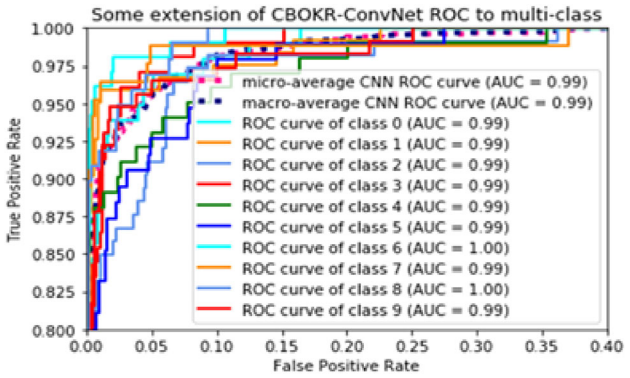


**Fig. 17** More Extension of ROC curve to multi-class for CBOKR-ConvNet Model

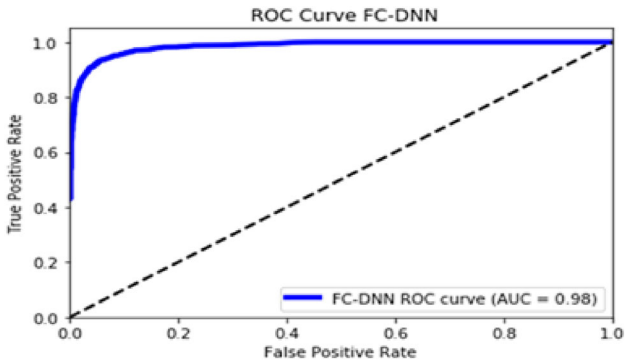


**Fig. 18** ROC curve of FC-DNN Model

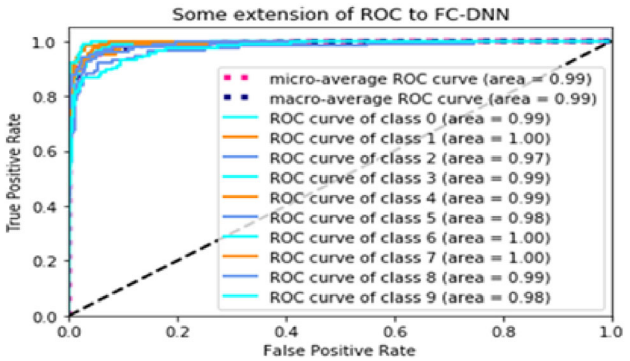


**Fig. 19** Extension of ROC curve to multi-class for FC-DNN Model

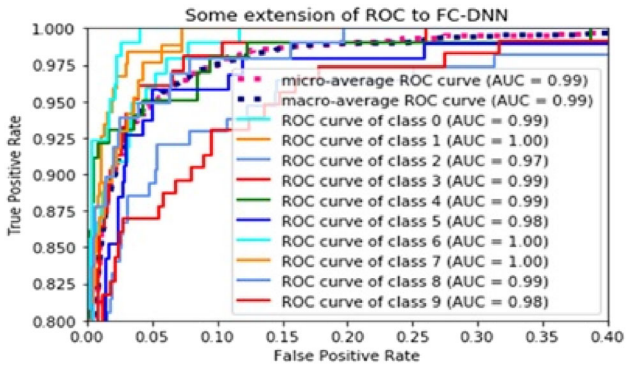


**Fig. 20** More Extension of ROC curve to multi-class for FC-DNN Model

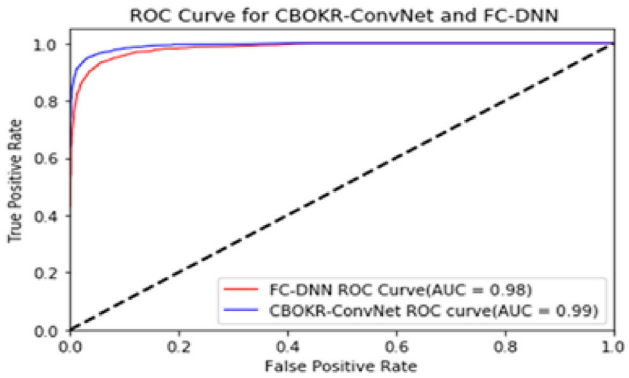


**Fig. 21** Comparison of ROC curve for both the Models

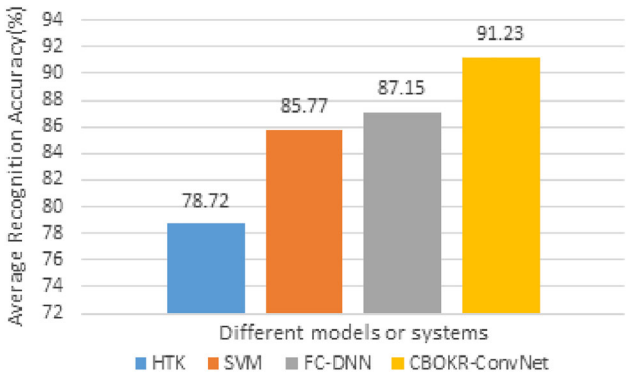


**Fig. 22** Comparison of average accuracy of different models/systems

score for the CBOKR-ConvNet model is 0.99 (Fig.15) which is more than the average AUC score of the FC-DNN model which is 0.98 (Fig. 18). Hence, from different performance metrics and the average AUC values obtained from both the models, It has observed that, the proposed CBOKR-ConvNet achieves better classification performance compared to the FC-DNN model.

Further, the dataset is being tested with other models or systems using HTK[24] and SVM [25]. Fig. 22 shows the graphical representation of the average recognition accuracy of different models used in the research work. It has been noticed that CBOKR-ConvNet outperforms compared to other models or systems.

## 6 Conclusion and future work

In the presented work, a convolutional neural network model for context based Odia keyword recognition (CBOKR -ConvNet) has proposed. The model is customized to recognize 10 keywords which are considered as

10 important body parts of a human being, voiced using Odia language. At the same time, the model is also compared with another deep neural network model (FC-DNN). Numerous experiments have assessed with varying the split ratio (training- validation-testing) % with different epochs. The experimental analysis reveals that the proposed model outperforms over FC-DNN and deliberates better testing accuracy of 91.23% with the split ratio of 80-10-10 and epoch value as 200. The number of parameters is more in CBOKR-ConvNet in comparison to FC-DNN, which also generates better outcomes in terms of various performance metrics. Also, the proposed model provides better AUC values for each of the classes in comparison to the FC-DNN model. Hence the ROC curve of the CBOKR-ConvNet model is superior to the other considered model. Finally considering the current data set, average recognition accuracy for other models using HTK and SVM are computed. Analyzing the reported result, It is clear that CBOKR-ConvNet unveils better average recognition accuracy in comparison with other models or systems.

In future, the study can be extended for designing a better CNN model that will improve the recognition accuracy with a large corpus. One limitation of the proposed CNN model is a large number of parameters, which takes more time for training and computing the results. Considering with less number of parameters, a new CNN model can be aimed for making it possible to work on some power-constrained machineries.

## References


1. Schalkwyk J, Beeferman D, Beaufays F, Byrne B, Chelba C, Cohen M, Kamvar M, Strope B (2010) Your word is my command: Google search by voice: a case study. Adv Speech Recogn, 61-90
2. Guoguo C, Parada C, Heigold G (2014) Small-footprint keyword spotting using deep neural networks.IEEE Int Conf Acoust Speech Signal Proces (ICASSP), Florence, 4087-4091
3. Rohlicek JR, Russell W, Roukos S, Gish H (1990) Continuous hidden Markov modeling for speaker-independent wordspotting. In: Proc Int Conf Acoust Speech Signal Proces (ICASSP). IEEE, 627–630
4. Rose RC, Paul DB (1990). A hidden Markov model based keyword recognition system. In: Proc Int Conf Acoust Speech Signal Proces (ICASSP). IEEE, 129-132
5. Wilpon JG, Miller LG, Modi P (1991) Improvements and applications for key word recognition using hidden Markov modeling techniques. In: Proc Int Conf Acoust Speech Signal Processing (ICASSP). IEEE, 309-312
6. Silaghi MC (2005) Spotting subsequences matching an HMM using the average observation probability criteria with application to keyword spotting. AAAI Vol. 3, 1118-1123
7. Grangier D, Keshet J, Bengio S (2009) Discriminative keyword spotting. Automatic speech and speaker recognition: large margin and kernel methods,Vol. 51,175-194
8. Tabibian S, Akbari A, Nasersharif B (2011) An evolutionary based discriminative system for keyword spotting. In 2011 International Symposium on Artificial Intelligence and Signal Processing (AISP), IEEE,83-88
9. Li KP, Naylor JA, Rossen ML (1992) A whole word recurrent neural network for keyword spotting. In [Proceedings] ICASSP-92: 1992 IEEE International Conference on Acoustics, Speech, and Signal Processing,IEEE, 2:81-84
10. Fernández S, Graves A, Schmidhuber J (2007) An application of recurrent neural networks to discriminative keyword spotting. In International Conference on Artificial Neural Networks, Springer, Berlin, 220-229
11. Tóth L (2014) Combining time-and frequency-domain convolution in convolutional neural network-based phone recognition. In 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE,190-194
12. Sainath TN, Mohamed AR, Kingsbury B, Ramabhadran B (2013) Deep convolutional neural networks for LVCSR. In 2013 IEEE international conference on acoustics, speech and signal processing, IEEE, 8614-8618
13. Li X, Zhou Z (2017) Speech Command Recognition with Convolutional Neural Network. CS229 Stanford education
14. Warden P (2018) Speech commands: a dataset for limited-vocabulary speech recognition. arXiv preprint arXiv:1804.03209
15. Jufarsky D, Martin JH, Jufarsky D, Martin JH (2000) Speech and language processing: an introduction to natural language processing, computational linguistics, and speech recognition, Prentice Hall series in artificial intelligence.
16. Dave N (2013) Feature extraction methods LPC, PLP and MFCC in speech recognition. Int J Adv Res Eng Technol 1(6):1–4
17. LeCun Y, Bengio Y (1998) Convolutional networks for images, speech, and time series, The handbook of brain theory and neural networks
18. Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems, p.1097-1105
19. Zhang X, Trmal J, Povey D, Khudanpur S (2014) Improving deep neural network acoustic models using generalized maxout networks.In 2014 IEEE international conference on acoustics, speech and signal processing (ICASSP), IEEE, p.215-219
20. Zhang Y, Pezeshki M, Brakel P, Zhang S, Bengio CLY, Courville A (2017) Towards end-to-end speech recognition with deep convolutional neural networks. arXiv preprint arXiv:1701.02720
21. Sainath TN, Parada C (2015) Convolutional neural networks for small-footprint keyword spotting. In Sixteenth Annual Conference of the International Speech Communication Association
22. Hinton GE, Srivastava N, Krizhevsky A, Sutskever I, Salakhutdinov RR (2012) Improving neural networks by preventing co-adaptation of feature detectors. arXiv preprint arXiv:1207.0580
23. Landis JR, Koch GG (1977) The measurement of observer agreement for categorical data. Biometrics, 33(1): 159-74. PMID: 843571159-174
24. Mohanty P, Nayak AK (2018) Isolated Odia digit recognition using HTK: an implementation view. In 2018 2nd International Conference on Data Science and Business Analytics (ICDSBA), IEEE, 30-35
25. Mohanty P, Nayak AK (2019) Multi-class support vector machine based continuous voiced Odia numerals recognition. Int J Sci Technol Res 8(10):2754–2764
26. Mohapatra H, Panda Rath AK, N, (2022) IoT infrastructure for the accident avoidance: an approach of smart transportation. Int J Inf Tecnol. https://doi.org/10.1007/s41870-022-00872-6
27. Mohanty P, Sahoo JP, Nayak AK (2022) Voiced Odia digit recognition using convolutional neural network. Advances in distributed computing and machine learning. Lecture Notes in

Networks and Systems, vol 302. Springer, Singapore. https://doi.org/10.1007/978-981-16-4807-6_16
28. Rusia MK, Singh DK (2021) An efficient CNN approach for facial expression recognition with some measures of overfitting. Int J Inf Tecnol 13:2419–2430. https://doi.org/10.1007/s41870-021-00803-x
29. de Coimbra AD, Sabato L, Viana Martin Loesener Da S, Christoph B (2018) A neural attention model for speech command recognition, arXiv preprint arXiv:1808.08929
30. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser Ł, Polosukhin I (2017) Attention is all you need. Adv Neural Inform Proces Syst,Vol. 30
31. Bahdanau D, Cho K, Bengio Y (2014) Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473
32. Amodei D, Ananthanarayanan S, Anubhai R, Bai J, Battenberg E, Case C, Casper J, Catanzaro B, Cheng Q, Chen G, Chen J (2016) PMLR. End-to-end speech recognition in English and Mandarin, In International conference on machine learning, pp 173–182
33. Tang Raphael, Lin Jimmy (2018) Deep residual learning for small-footprint keyword spotting, In 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), p.5484-5488
34. Arik S, Kliegl M, Child R, Hestness J, Gibiansky A, Fougner C, Prenger R, Coates A (2020) Convolutional recurrent neural networks for small-footprint keyword spotting. U.S. Patent 10,540,961, Baidu USA LLC
35. Zhang Y, Suda N, Lai L, Chandra V (2017) Hello edge: Kkeyword spotting on microcontrollers, arXiv preprint arXiv:1711.07128