Journal of Scientific Research and Development (2023) Vol. 22 (1) 115 - 125

A bi-annual journal published by the Faculty of Science, University of Lagos, Nigeria
http://jsrd.unilag.edu.ng/index.php/jsrd

A hybrid CNN-LSTM model for speaker independent command word recognition

Ebun Phillip Fasina*, Babatunde Alade Sawyerr, Chibuzor Nwalor, Ogban Ugot
Department of Computer Sciences, University of Lagos, Akoka. Nigeria
*Corresponding author: efasina@unilag.edu.ng

(Received 02 May 2023/Revised 24 June 2023/Accepted 25 June 2023)

Abstract

Automatic speech keyword recognition is an important subset of general speech recognition. It is especially relevant in situations with limited computational resources, such as voice command recognition in low-power/low-memory device and robot interaction. This paper introduces a method for performing efficient Speaker Independent Real Time Command Word Recognition using a hybrid Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) network with only 9.8K trainable parameters. CNN extracts short-term spatial features from Mel Frequency Cepstral Coefficients of command words arranged into an image-like format. LSTM learns extracted spatial features as long-term dependences. The model is trained and evaluated on the Google Speech Commands dataset on which it achieved an accuracy of 83%, a memory requirement that is 2-5% of state-of-the-art models and a faster response time when compared to off-the-shelf models.

Keywords: Command Word Recognition, Convolutional Neural Network, Long Short-Term Memory, Deep Learning, Recurrent Neural Network, Natural Language Processing

Introduction

Speech recognition is the process of converting human speech into text by a computer and it is the intersection of linguistics and computer science. Speech recognition systems today are capable of recognizing thousands of words independent of any particular speaker by using powerful neural networks often run in the cloud, this means that while General Speech Recognition is extremely versatile, it is impractical for applications that require voice input typically restricted to a handful of command words from users as it is resource-intensive, relatively slow and subject to privacy breaches if an always-listening system streams audio to a remote device for processing (De Andrade et al. 2018).

This work introduces a hybrid Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) network architecture designed to recognize command words in real time and capable of running locally on low-end devices. The main contribution of this work is the development of a speaker independent command word recognition system using a hybrid CNN-LSTM model that is trained on MFCC of command words. The model achieves an accuracy of 83% with extremely low trainable parameters of 9.8K (2-5% of state-of-art models) and very fast response that is 4 times faster than off-the-shelf models. The CNN-LSTM meets the requirements of being suitable for low-resource devices and robots.

Literature Review

After many decades of research and development the accuracy of speech recognition systems has begun to improve considerably with the advent of deep learning architectures so that they can now be compared to that of humans. Traditionally, speech recognition has been confronted with challenges such as variations and context in speech, speaker independence and noise in the environment. To tackle these challenges researchers in speech recognition have focused on understanding various classes of speech, speech representation, feature extraction techniques, speech classifiers, and performance evaluation.

Based on recent advances in statistical modeling and deep learning speech recognition systems have found application in human machine interfaces, query-based information systems, online stock price quotations, automated weather report, data entry, avionics, speech transcription, commands and assistance to the handicap, supermarkets, robotics, reservation systems etc. Figure 1 shows the basic model of speech recognition systems which can be mathematically represented as four components: acoustic front end, acoustic, language model and search unit (Anusuya and Katti, 2009).

The traditional approach to continuous speech recognition is to assume that speech is a specified word sequence X that produces an observed acoustics sequence Y with probability $P(X,Y)$. The goal of speech recognition systems is to find the correct sequence of words from the acoustic sequence by using an acoustic model $P(A|X)$ that maximizes with *a posterior probability* $P(X|A)$ that the spoken word sequence is the decoded word string.

$$P(X|A) = \arg max_X P(X|A) \qquad (1)$$

Applying Bayes rule to equation 1, gives

$$P(X|A) = \frac{P(A|X)P(X)}{P(A)} \qquad (2)$$

Since $P(A)$ is independent of X, the maximum a posterior probability decoding rule in equation 1 is

$$X = arg\, max_X P(A|X)P(X) \qquad (3)$$

$P(X)$ in equation 3 is the language model that defines the probability associated with a postulated sequence of words.

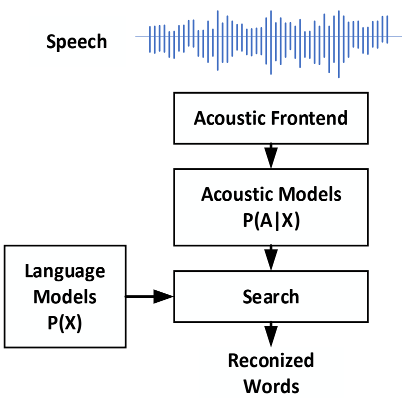

Figure 1: Continuous Speech Recognition System (Anusuya and Katti, 2009)

Speech is the most crucial and essential method of communication among human beings. It is a natural and fundamental way of communicating for most humans. The quest to develop frameworks for recognition of human speech has long been an intriguing issue for the scientific and computing world, which has yielded many results. Speech recognition, also known as Automatic Speech Recognition (ASR), involves automatically converting an acoustic or audio waveform into text by a computer (IBM Cloud Education, 2020). Over the years, various approaches to solving speech recognition problems have been explored. One of the early algorithms, invented by Soviet researchers in 1970, the Dynamic Time Warping (DTW) algorithm is still a popularly used algorithm for speech recognition (Çiçek, 2020). It calculates the optimal warping path, which is the shortest distance, between two data from sound, and produces an output which is the path warping values and the distance between the two data. Thus, it is used to find an optimal alignment between two given time-dependent sequences under certain restrictions. The system works in such a way that the smaller the warping path that is produced, the greater similarity can be found in the two patterns (Permanasari, 2019). DTW has been used to compare different speech patterns in automatic speech recognition. DTW becomes relevant in speech recognition because different recordings of the same words may include similar sounds in the same order, but the precise durations of each sub-word within the word may be incongruent, or not match. DTW recognizes words by matching them to templates with temporal alignment.

Hidden Markov Models (HMMs) are black boxes, where the sequence of output symbols generated over time is observable, but the sequence of states visited over time is hidden from the observer. While a regular Markov chain model is useful for clearer events, such as text inputs, HMMs allow for less obvious events, such as mapping part-of-speech tags into probabilistic models. Thus, it allows labels to be assigned to each unit, i.e., words, syllables, sentences, etc., in the sequence. These labels are mapped with the provided input, allowing it to determine the most appropriate label sequence (Trivedi, 2014).

With the advent of Artificial Neural Networks and Long Short-Term Memory Networks which provide a powerful way to model sequential data with long term dependencies, end to end speech recognition became possible (Graves, 2014). Central to this advancement is Connectionist Temporal

Classification (CTC), which allows Recurrent Neural Networks to be trained for sequence transcription tasks in which the alignment at each timestep between the input and target sequences is unknown (Graves, 2014). This made it feasible to calculate the loss at every timestep of an LSTM network learning speech recognition as CTC provided the alignment between the network's predictions at each timestep and the actual word.

The CTC network will strip the output of the repeated characters generated at successive timesteps and be decoded i.e., finding the most probable output transcription using a beam search algorithm and integrating a language model. One such model is the Deep Speech end to end speech recognition system composed of a 5-layer model and trained using CTC loss. In this model a spectrogram frame is passed in with left and right context as input with the first 3 layers were non recurrent with the 4th being a bidirectional RNN and the 5th layer also being non recurrent but taking inputs from both the forward and backward recurrent units, and the output layer is a softmax which outputs the probability of a character occurring. The model was trained using synthesized speech to inject noise into prerecorded labelled clean speech in order to simulate real life environments and provide robustness to noise and other audio effects without need of hand-crafted mechanisms to filter noise (Hannun et al., 2014).

(Abdel-Hamid et al., 2014) used the Mel-Frequency Spectral Coefficients (MFSC) to train CNN for speech recognition. They demonstrated that over the TIMIT phone recognition and voice search vocabulary it outperformed systems using Dense Neural Networks (DNN) by as much as 6-10%. Identification of speech commands, also known as keyword spotting, is important from an engineering perspective for a wide range of applications, from indexing audio databases and indexing keywords (Tabibian et al 2013) to running speech models locally in microcontrollers (Zhang et al, 2017).

(Chen et al., 2015) explored keyword spotting using a combination of Long Short Term Memory Networks and DTW. This approach involved generating a fixed length representation from variable length audio input and comparing the representation against averaged template samples using Cosine similarity to determine if a keyword was spoken. In this architecture, preprocessing steps involved using a voice-activity detection system to limit running of the Keyword Spotting System to voice regions and log-filter bank energies were computed for the voice regions and passed to the LSTM directly (Chen et al, 2015). (Gouda et al, 2018) in an attempt to represent audio samples as images and passed them to CNNs for perform keyword spotting. This architecture worked by converting audio into spectrograms which are two-dimensional structures suitable for processing by a CNN. The CNN explored here consists of two convolutional layers and two max pooling layers followed by a densely connected layer and a softmax output layer. This approach involved listening to the whole audio and converting it into a spectrogram thereby treating keyword spotting as an image recognition problem.

In recent years, hybrid CNN-LSTM architectures have come to the fore with the CNN part of the model typically used for local feature extraction and the LSTM part for learning long term dependencies. (De Andrade et al, 2018) introduced a CNN-LSTM model with attention where audio samples were converted into spectrograms and passed as input to a 2D CNN which performed local feature extraction and passed the result to a bidirectional LSTM. Similar to previous works using CNNs, this architecture required the whole audio sample before prediction. (Li and Zhou, 2019) developed three command word recognition systems using three machine learning algorithms, Vanilla Single-Layer softmax model, DNN and CNN using the MFCC features of command words. CNN had the best performance with an accuracy of 95.1% for six labels. (Waqar, 2021) developed a MFCC-CNN four speech command system with an accuracy of 96.5%. (Wubet, 2021) developed three models, CNN, SVM and CNN-SVM for keyword recognition and found that the CNN-SVM model outperformed the CNN and SVM models. The models were training the spectrograms of key words. (Yang et al. 2020) trained CNN, DNN, and LSTM using MFCC for command word recognition and found that CNN outperformed the other machine learning algorithms.

Methodology

The functional requirement of this work is to develop a speaker independent command word recognition system (SICWR) for low-resource devices and robots capable of recognizing a limited set of command words spoken in the English language. The non-functional requirements of SICWR are that a) the trained command word recognition model should be less that 5MB; b) the audio preprocessing should take less than 50 milliseconds per frame and c) the model should be at least 80% accurate. Given the low memory budget and response time of SICWR, it was decided that a hybrid CNN-LSTM be developed that recognized command words preprocessed into an MFCC vector format into 2D MFCC image. MFCCs are a more compact representation of speech signals than spectrograms. In earlier work (Abdel-Hamid et al., 2014) avoided using MFCC to train CNN for speech recognition because according to them the Discrete Cosine Transform (DCT) that generates MFCCs project spectral energies into a new basis that may not maintain locality. Instead, they used Mel-Frequency Spectral Coefficients (MFSC). In this work the non-locality of MFCCs is not a drawback it is learned by LSTM and used for word classification. The rest of this section is outlined in the following subsections: a) the SICWR system architecture b) the audio preprocessor module c) CNN, d) LSTM, e) the CNN-LSTM hybrid model implementation, and f) the Labeller.

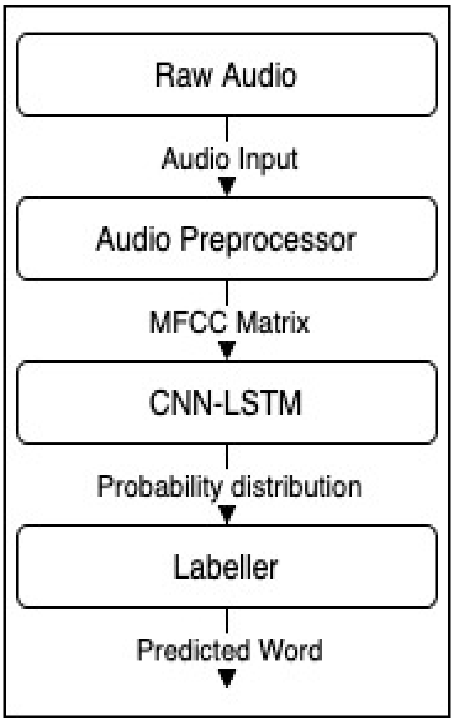

Figure 2: CNN-LSTM SICWR architecture

SICWR System Architecture

The SICWR system architecture is shown in Figure 2. It starts by taking in a raw audio file as input and passes this file to the Audio Preprocessor which will perform the MFCC extraction with a frame size of 25ms, an overlap of 15ms, 512 discrete Fourier transform points, and a 26-filter Mel-filterbank. Mean normalization was then applied to help improve the Signal to Noise Ratio (SNR) of the audio signal (Subramanian and Chong, 2019). Figure 3 shows the visualization of the mean normalized MFCC of typical audio signal of a command word.

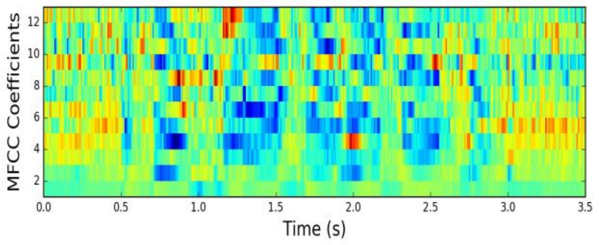

Figure 3: Mean Normalized MFCCs (Fayek, 2016)

After the MFCC feature extraction, only 12 coefficients (from the 2nd to the 13th) will be kept. The coefficients are then stored as one row in the input feature matrix and subsequent rows are filled up via the same process. The result of this is a matrix containing rows of MFCC coefficients stacked on top of each other to form an ‘image’ of coefficients over a period of 162.5ms. The 12×12 matrix is then passed as input to the 2D CNN for local feature extraction. The 2D CNN consists of a 4×4 convolutional layer and a 2×2 max pooling layer followed by a 44×1 fully connected layer. The 44×1 vector is passed to a single layer unidirectional LSTM with a 64×1 hidden state size. The LSTM’s output is passed to 2 fully connected layers with dimensions 32×1 and $N_c \times 1$ where N_c is the number of command words. The network makes use of the softmax activation function to convert the $N_c \times 1$ vector into a probability distribution of the uttered command word. Table 1 summarizes the CNN-LSTM architecture. Figures 6 and 7 show the schematic of the hybrid model architecture and the system’s architecture.

The system architecture consists of an Audio preprocessor, the CNN-LSTM SICWR, and a Labeller. The function performed by the Audio preprocessor and Labeller are what makes the SICWR usable.

Audio Preprocessor

Raw audio streams are not directly usable by neural networks, so some preprocessing will have to be done to convert the audio format then clean it up to make training the model easier. The audio preprocessor handles the key process of converting a raw audio stream of input vectors usable by the neural network. This class is instantiated with an audio stream as well as the values for preprocessing like the frame size, frame overlap and windowing function. The output is in the form of a matrix representing a collection of processed audio frames. The following preprocessing steps are implemented by the Audio preprocessor:

Framing: Here the audio is split into a sequence of overlapping frames. This is important as it is required for the Fourier transform of the audio signal. Since short-term frequencies of signals are stationary, they can be captured effectively this way.

Windowing: After the audio is sliced into frames, the Hamming window function is applied to each frame to reduce spectral leakage.

Short Time Fourier transform: Next, a fast Fourier Transform is done on each frame to get the frequency spectrum and then compute the power spectrum.

Mel Frequency Cepstral Coefficients: The sound of speech is governed by the shape of the human vocal tract which manifests as an envelope of the short time power spectrum obtained in the previous step. MFCCs give an accurate and compact representation of this envelope.

Mean Normalization: After the input features are generated, they are scaled to balance the spectrum and SNR

Grouping: After other preprocessing steps, the output vector is stacked in batches of 12 to form the MFCC image.

Convolutional Neural Networks

Convolutional Neural Networks are a class of ANNs that use convolution operations instead of general matrix multiplication in at least one of their layers. This gives CNN its unique form of *sparse matrix* and *shared weight* regularization (Bezdan, and Džakula, 2019; Taye, 2023). On the other hand, Multilayer Perceptrons (MLPs) are fully connected networks, where each neuron in one layer is connected to all neurons in the next layer making these networks prone to overfitting data (LeCun, 1989). The configuration of one of the early CNNs, "LeNet" (LeCun, 1989) is shown in Figure 4 is described to explain of operation of CNNs. LeNet has 7 layers with trainable parameters or weights. The input is a gray scale 32×32 pixel image. The main operations performed by CNNs are convolution and pooling. Layer C1 is a convolution layer with 6 neurons, each with inputs from 5×5 kernels and output to 28×28 feature maps. C1 has 156 parameters or weights and contains 122,304 connections. Convolution operations extract high-level features from input data by performing a matrix multiplication between a filter and its receptive field or kernel, i.e., a restricted area of a previous layer typically a square as opposed to fully connected layers whose receptive field is the whole previous layer.

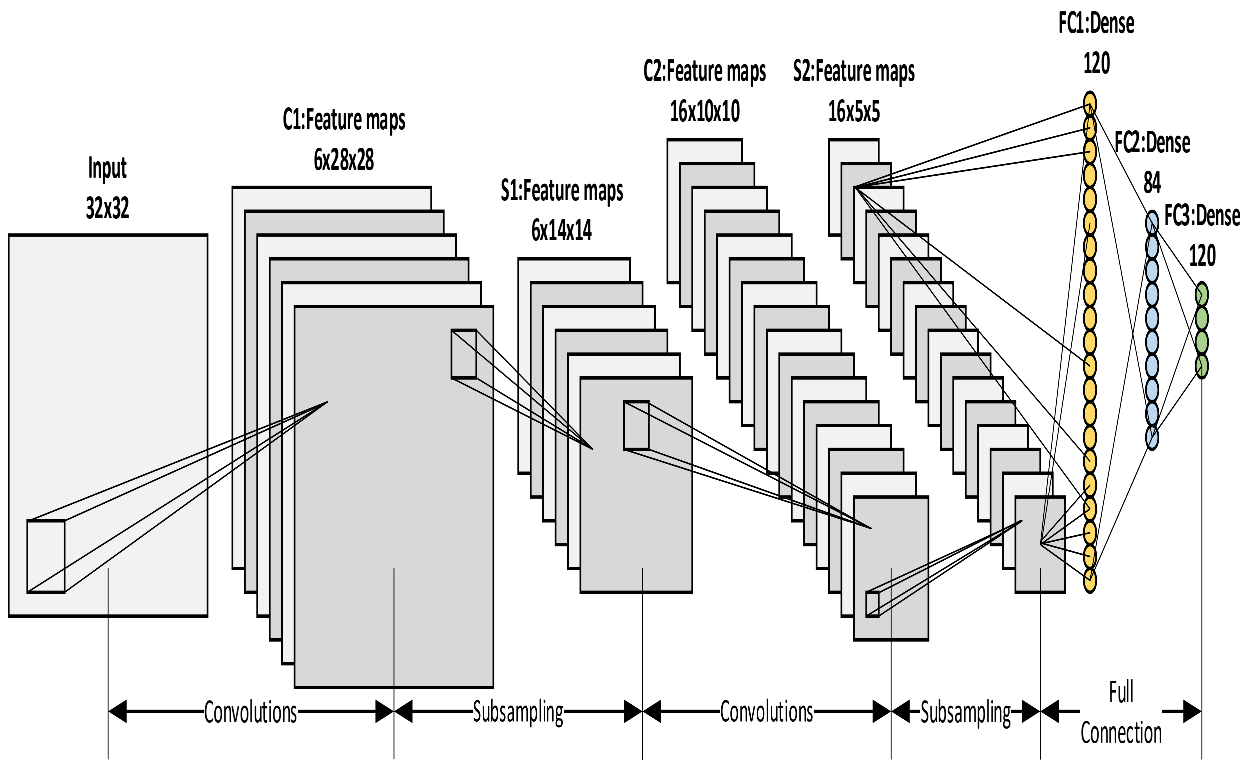

Figure 4: LeNet CNN architecture (LeCun, 1989)

Starting from the beginning of the input, the filter moves to the right with a certain stride until it reaches the end of a row and then goes down to the next row and repeats this process until the whole input has been convolved. Figure 5 gives an exploded view of the first convolution and subsampling layers. The equation for a convolution operation with a 2D filter is given by equations 4, 5, and 6 (Zhang, 2016).

$$\sigma(x) = \frac{1}{1 + exp^{-x}} \tag{4}$$

$$C_p^1 = \sigma(I * k_{1,p}^1 + b_p^1) \tag{5}$$

$$C_p^1(i,j) = \sigma\left(\sum_{u=-2}^{2}\sum_{v=-2}^{2} I(i-u, j - v) k_{1,p}^1(u,v) + b_p^1\right) \tag{6}$$

Equation 4 is the activation function of each neuron and equation 5 is a convolution operation ($*$ is the convolution operator). Equation 6 describes the complete convolution over the image I with output to the feature map C_p^1 at location i (row) and j (column).

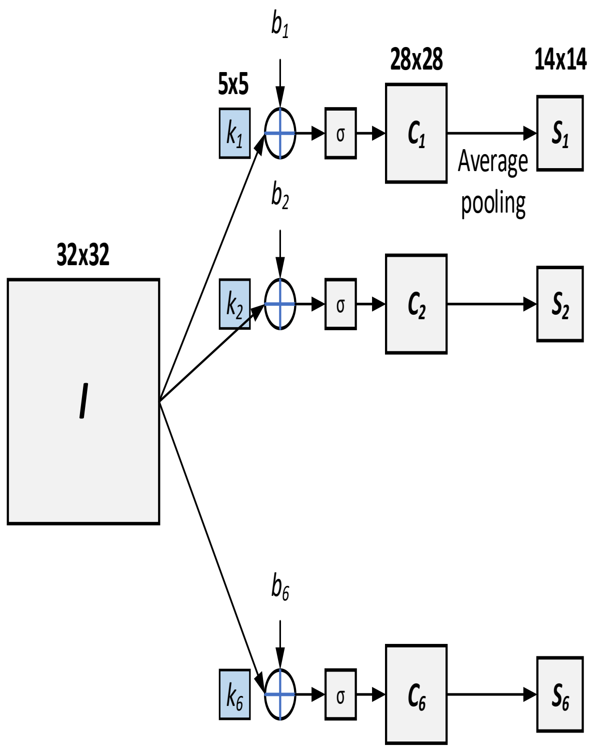

Figure 5: First convolutional layer of LeNet.

Layer S1 is a subsampling layer with 6 feature maps of size 14×14 that are each connected to pooling unit with 2×2 nonoverlapping receptive field on feature maps C_p^1. The Pooling operation is used for reducing the size of features produced by the convolutional layers. This decreases the computational power required to process the data via dimensionality reduction and extracts dominant features which are rotational and positional invariant, thus maintaining the process of effectively training the model. The two types of pooling are max pooling which returns the maximum value in the receptive field and average pooling which returns the average of all the values in the receptive field. Max pooling discards noisy values and so it performs denoising along with dimensionality reduction, compared to average pooling which averages noise with actual data. The equation for average pooling at the first subsampling layer is given in (7).

$$S_p^1(i,j) = \frac{1}{4}\sum_{u=0}^{1}\sum_{v=0}^{1} C_p^1(2i-u, 2j-v) \tag{7}$$

where $i,j = 1,2,\ldots,14$. Equations 8, 9 and 10 describe the convolution at layer C2 and average pooling at layer S2 respectively.

$$C_q^2 = \sigma\left(\sum_{p=1}^{6} S_p^1 * k_{p.q}^2 + b_q^2\right) \tag{8}$$

$$C_p^2(i,j) = \sigma\left(\sum_{p=1}^{6}\sum_{u=-2}^{2}\sum_{v=-2}^{2} S_p^1(i-u, j - v) k_{1,p}^2(u,v) + b_p^2\right) \tag{9}$$

$$S_q^2(i,j) = \frac{1}{4}\sum_{u=0}^{1}\sum_{v=0}^{1} C_q^2(2i-u, 2j-v) \tag{10}$$

$q = 1,2,\ldots,16$ and $i,j = 1,2,\ldots,10$. Units in the fully connected layers are classical neurons that compute the dot product of the input vector with their weight vector W and add the bias. The weighted sum

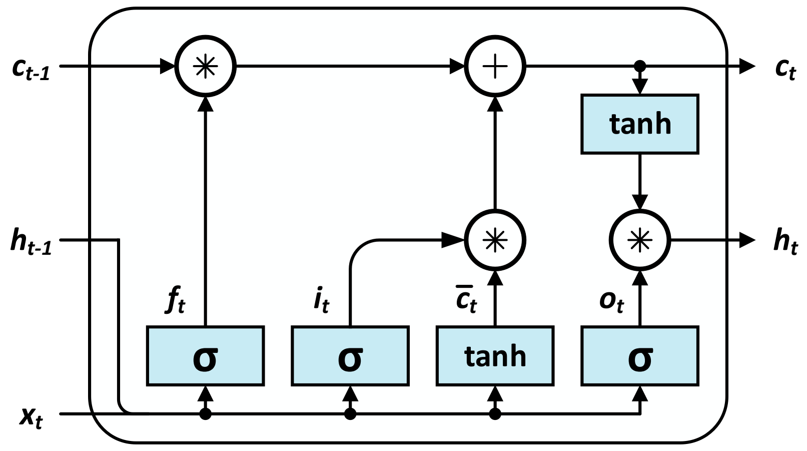

Figure 6: Network structure of an LSTM cell

is then passed through the sigmoid activation function to produce the final output. Neurons at the fully connected layer compute the final output using equation 11 below:

$$y = \sigma(W \times f + b) \qquad (11)$$

where f is the flattened vector at the output of the second pooling layer, b is bias and σ is the activation or sigmoid function. Convolutional neural networks are trained by backpropagation. Errors at the output of the dense or fully connected layer are backpropagated, a process that adjusts the network's weights to minimize the output error or loss.

Long Short-Term Memory (LSTM)

LSTM networks are a kind of Recurrent Neural Network (RNN) architecture. RNNs are a class of artificial neural networks primarily used for processing sequential data and derive their recurrent nature by using the same set of neuron weights for every data point in a sequence while making the output of the current input depend on the results of past computations. RNNs are trained using backpropagation through time which works by unrolling all input timesteps and executing the original backpropagation algorithm. After unrolling for a number of time steps, errors are then calculated and accumulated for each timestep. The network is rolled back up and the weights are updated with accumulated error using backpropagation (Werbos, 1990). In 'deep' recurrent neural networks i.e., where there are thousands of timesteps, thousands of derivatives will be required for a single weight update, and this can cause gradients to sometimes vanish or explode. This limits the ability of an RNN to effectively learn long term dependencies leading to short term memory (Hochreiter, 1998). LSTMs get around this by making use of internal mechanisms called gates to regulate the internal memory of the network, namely the input, output and forget gates. As part of the training process, LSTMs learn what parameters to assign to each gate and so learn what parts of data in a sequence are important to keep and what parts to discard (Hochreiter, 1997).

As can be seen from Figure 6 and equations (12) to (17) listed below each gate has a unique function as well as weights and bias parameters. For every input x_t generate a hidden activation h_t, the forget gate f_t, an update gate i_t, an output gate o_t and updates the memory cell c_t. In the forward-pass of the LSTM, the input gate given the previous hidden state and current input decides what portion of the state will be updated. The forget gate given the same inputs, decides what portion of the state should be discarded. Both gates produce values between 0 and 1 and so the state is updated by multiplying the forget gate's value with the previous state, multiplying the candidate state by the input gate's value and summing the results. Similarly, the output gate decides the value of the new hidden state given the new cell state.

$$\square_{\square} = \square(\square_{\square h} h_{\square-1} + \square_{\square\square}\square_{\square} + \square_{\square}) \qquad (12)$$

$$\square_{\square} = \square(\square_{\square h} h_{\square-1} + \square_{\square\square}\square_{\square} + \square_{\square}) \qquad (13)$$

$$\tilde{\square}_{\square} = \square\square\square h(\square_{\tilde{\square} h} h_{\square-1} + \square_{\tilde{\square}\square}\square_{\square} + \square_{\tilde{\square}}) \qquad (14)$$

$$\square_{\square} = \square_{\square} * \square_{\square-1} + \square_{\square} * \tilde{\square}_{\square} \qquad (15)$$

$$\square = \square(\square_{\square h} h_{\square-1} + \square_{\square\square}\square_{\square} + \square_{\square}) \qquad (16)$$

$$h_{\square} = \square_{\square} * \square\square\square h(\square_{\square}) \qquad (17)$$

CNN-LSTM Hybrid Model Implementation

The hybrid CNN LSTM model (see Table 1) was implemented using Keras. The MFCC feature extraction (Muda et al., 2010) was done using the Librosa library (McFee et al, 2015). Inputs to the system are raw audio streams converted to stacked *numpy* arrays of Mel Frequency Cepstral Coefficients (MFCC). The model is arranged in this way to allow CNN to extract local features across a relatively short time frame from an 'image' of cepstral coefficients. The image of cepstral coefficients contains the left and right context for every MFCC frame. CNN passes extracted short-term features to the unidirectional LSTM to learn longer term dependencies and output predictions in real time.

Table 1: Summary of CNN-LSTM model

Layer	Output size	Operation
Convolution	12×12	Conv (4×4) Stride = 1
Pooling	6×6	Maxpool (4 × 4); Stride = 2
Flattening	36×1	
Dense	44×1	Batch Norm + ReLU
LSTM	64×1	
Dense	32×1	ReLU
Softmax	$(N_c \times 1) \times 1$	

Labeller

The Labeller serves as the final output layer of the SICWR, as it handles the mapping of the networks output (in form of a probability distribution).to the detected command word's text, or null value if no command word was detected.

Results and Discussion

The model was trained on the 12-Command Google Speech Commands dataset containing the target speech commands as well as negative samples. The dataset contains 100,503 samples of 1-second-long audio containing either the command word or a negative sample along with a label corresponding to the target class. The dataset was split into an 85-5-10 train-test-validation dataset. The model was trained for 11 epochs using the Adam optimizer (Kingma, 2014) with a learning rate of 0.001. Plots of the model's accuracy and loss with increasing training epochs a shown in Figures 7 and 8.

Table 2: Accuracy results on 12-commands from Google Speech Command Dataset

Model	Accuracy (%)	Trainable Parameters
Attention RNN v2	96.9	202 K
ConvNet on raw WAV	89.4	700 K
DS-CNN	95.4	498 K
res 8	94.1	110 K
res15	95.8	238 K
res26	95.2	438 K
CNN-LSTM	83.0	9.8 K

The performance of the CNN-LSTM model is compared with other state-of-the-art models (see Table 2), e.g., res15, res26, and Attention RNN v2 (De Andrade et al., 2018). The results obtained from the CNN-LSTM are significantly less accurate than the all the models except ConvNet, it meets our benchmark of 80% with the tradeoff of being around 2-5% the size of state-of-the-art models. The runtime performance is compared with the two versions of DenseNet-121. The DenseNet-121 model A (see Table 3) is without pretraining and multiscale. DenseNet-121 model B is pretrained on UltraSound8K without multiscale (UltraSound8K, 2023).

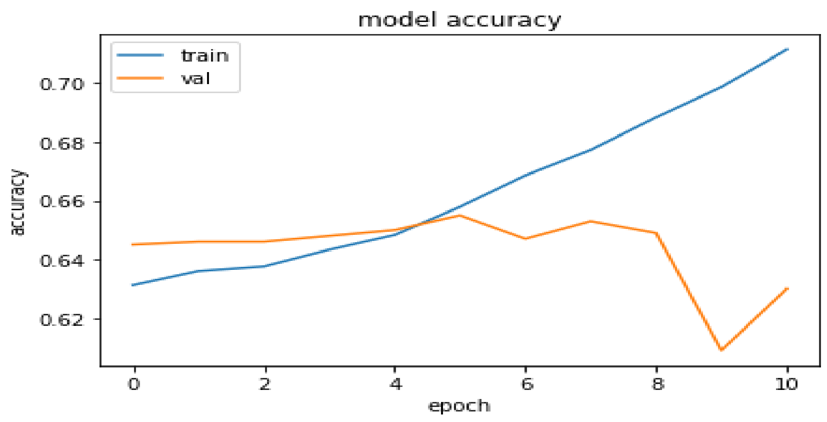

Figure 7: Plot of the CNN-LSTM model accuracy with increasing training epochs.

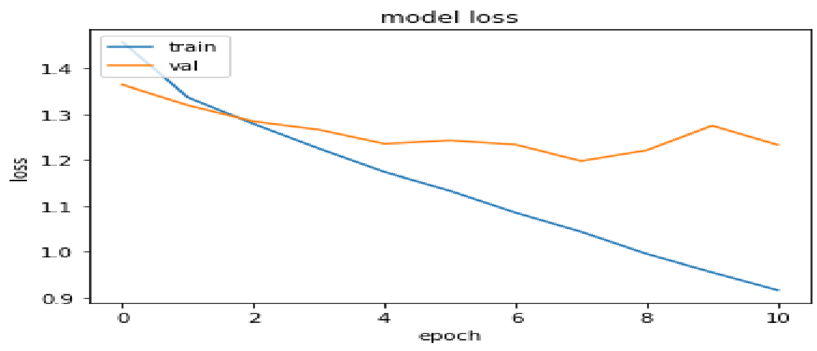

Figure 8: Plot of model's loss with increasing training epochs

It can be observed that the CNN-LSTM command word recognition system outperforms the two DenseNet-121 models, both in terms of memory requirements and prediction delay. There is almost fourfold improvement in both average memory and prediction delay over the two models of DenseNet-121.

Table 3: Performance metrics

Model	Average CPU Usage (%)	Average Memory Usage (MB)	Prediction Delay (secs)
DenseNet-121 A	40	1500	2
DenseNet-121 B	60	2000	1.8
CNN-LSTM	20	500	0.5

Conclusion and Recommendations

Speech command recognition is an important component of HCI systems and models capable of running locally with a small footprint while maintaining a reasonable accuracy are a crucial requirement. In this work, we introduced an CNN-LSTM architecture that achieves this benchmark.

The recognition system takes raw WAV files as inputs, extracts its features in the form of MFCC, arranges them into an image of coefficients and passes them to the hybrid CNN-LSTM Model which outputs a probability distribution that is converted into the text of the spoken word if a command word was detected.

The Google Speech Commands dataset was used to train and evaluate the effectiveness of our model. The model achieved a 83% accuracy while keeping average CPU usage at 5%, average memory usage at 50MB and an extremely small size of 9.8k trainable parameters. This model is therefore well suited to resource constrained environments with the tradeoff being lower accuracy.

Future work will involve exploring different recurrent architectures with the aim of reducing complexity even further.

References

Abdel-Hamid, O., Mohamed, A. R., Jiang, H., Deng, L., Penn, G., & Yu, D. (2014). Convolutional Neural Networks for Speech Recognition. IEEE/ACM Transactions on Audio, Speech, And Language Processing, 22(10), 1533-1545.

Anusuya, M. A., & Katti, S. K. (2010). Speech Recognition by Machine, A Review. *arXiv preprint arXiv:1001.2267*.

Bezdan, T., & Džakula, N. B. (2019, January). Convolutional neural network layers and architectures. In *Data Science and Digital Broadcasting Systems, International Scientific Conference On Information Technology And Data Related Research*, (Vol. 10).

Chen, G., Parada, C., & Sainath, T. N. (2015). Query-By-Example Keyword Spotting using Long Short-Term Memory Networks. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* (pp. 5236-5240). IEEE.

Çiçek, B.G. (2020, July) *How to do Speech Recognition with a Dynamic Time Warping Algorithm: Understanding a dynamic time warping algorithm,* https://betterprogramming.pub/how-to-do-speech-recognition-with-a-dynamic-time-warping-algorithm-159c2a1bb83c

De Andrade, D. C., Leo, S., Viana, M. L. D. S., and Bernkopf, C. (2018). A Neural Attention Model for Speech Command Recognition. *arXiv preprint arXiv:1808.08929*.

Fayek, H. (2016) *Speech Processing for Machine Learning: Filter banks, Mel-Frequency Cepstral Coefficients (MFCCs) and What's In-Between* https://haythamfayek.com/2016/04/21/speech-processing-for-machine-learning.html

Gouda, S. K., Kanetkar, S., Harrison, D., and Warmuth, M. K. (2018). Speech Recognition: Keyword Spotting Through Image Recognition. *arXiv preprint arXiv:1803.03759.*

Graves, A., and Jaitly, N. (2014, June). Towards End-to-End Speech Recognition with Recurrent Neural Networks. In *International Conference on Machine Learning* (pp. 1764-1772). PMLR.

Hannun, A., Case, C., Casper, J., Catanzaro, B., Diamos, G., Elsen, E., Prenger, R., Satheesh, S., Sengupta, S., Coates, A., and Ng, A. Y. (2014) Deep Speech: Scaling Up End-To-End Speech Recognition. *arXiv preprint arXiv:1412.5567, 12*.

Hochreiter, S. (1998). The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02), 107-116.

Hochreiter, S., and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8), 1735-1780.
https://www.ibm.com/cloud/learn/speech-recognition

IBM Cloud Education (2020, September) *What is Speech Recognition?*

Li, X., and Zhou, Z. (2019). Speech Command Recognition with Convolutional Neural Network. Stanford CS 229 Projects. [Online]. Available: http://cs229.stanford.edu/proj2017/final-reports/5244201.pdf

McFee, B., Raffel, C., Liang, D., Ellis, D. P., McVicar, M., Battenberg, E., and Nieto, O. (2015, July). Librosa: Audio and Music Signal Analysis in Python. In *Proceedings of the 14th python in science conference* (Vol. 8, pp. 18-25).

Muda, L., Begam, M., and Elamvazuthi, I. (2010). Voice Recognition Algorithms using Mel Frequency Cepstral Coefficient (MFCC) and Dynamic Time Warping (DTW) Techniques. *arXiv preprint arXiv:1003.4083.*

Permanasari, Y., Harahap, E. H., and Ali, E. P. (2019, November). Speech recognition using dynamic time warping (DTW). In *Journal of Physics: Conference Series*. vol. 1366, no. 1, p. 012091. IOP Publishing.

Subramanian, A. K., and Chong, N. Y. (2019, August). Mean Spectral Normalization of Deep Neural Networks For Embedded Automation. In *IEEE 15th International Conference on Automation Science and Engineering (CASE)* (pp. 249-256). IEEE.

Tabibian, S., Akbari, A., and Nasersharif, B. (2013). Keyword spotting using an evolutionary-based classifier and discriminative features. *Engineering Applications of Artificial Intelligence*, 26(7), 1660-1670.

Taye, M. M. (2023). Theoretical Understanding of Convolutional Neural Network: Concepts, Architectures, Applications, Future Directions. Computation, 11(3), 52.

Trivedi, P. A. (2014). Introduction To Various Algorithms of Speech Recognition: Hidden Markov Model, Dynamic Time Warping and Artificial Neural Networks. *International Journal of Engineering Development and Research*, 2(4), 3590-3596.

UltraSound8K (2023) UltraSound8K Dataset. https://urbansounddataset.weebly.com/urbansound8k.html

Waqar, D. M., Gunawan, T. S., Kartiwi, M., & Ahmad, R. (2021). Real-Time Voice-Controlled Game Interaction using Convolutional Neural Networks. In *IEEE 7th International Conference on Smart Instrumentation, Measurement and Applications (ICSIMA)* (pp. 76-81). IEEE.

Werbos, P. J. (1990). Backpropagation through Time: What It Does and How to Do It. *Proceedings of the IEEE*, 78(10), 1550-1560.

Wubet, Y. A., & Lian, K. Y. (2021). A Hybrid Model Of CNN-SVM For Speakers' Gender and Accent Recognition Using English Keywords. In *IEEE International Conference on Consumer Electronics-Taiwan (ICCE-TW)* (pp. 1-2). IEEE.

Yang, X., Yu, H., & Jia, L. (2020). Speech Recognition of Command Words Based on Convolutional Neural Network. In *International Conference on Computer Information and Big Data Applications (CIBDA)* (pp. 465-469). IEEE.

Zhang, Y., Suda, N., Lai, L., and Chandra, V. (2017). Hello Edge: Keyword Spotting on Microcontrollers. *arXiv preprint arXiv:1711.07128*.

Zhang, Z. (2016). Derivation Of Backpropagation in Convolutional Neural Network (CNN). *University of Tennessee, Knoxville, TN, 22, 23*.