A comparative review of dynamic neural networks and hidden Markov model methods for mobile on-device speech recognition

MUSTAFA, Mohammed, ALLEN, Tony and APPIAH, Kofi <http://orcid.org/0000-0002-9480-0679>

Neural Comput & Applic
DOI 10.1007/s00521-017-3028-2

ORIGINAL ARTICLE

A comparative review of dynamic neural networks and hidden Markov model methods for mobile on-device speech recognition

Mohammed Kyari Mustafa[1] · **Tony Allen**[2] · **Kofi Appiah**[2]

Received: 9 June 2016 / Accepted: 19 April 2017

Abstract The adoption of high-accuracy speech recognition algorithms without an effective evaluation of their impact on the target computational resource is impractical for mobile and embedded systems. In this paper, techniques are adopted to minimise the required computational resource for an effective mobile-based speech recognition system. A Dynamic Multi-Layer Perceptron speech recognition technique, capable of running in real time on a state-of-the-art mobile device, has been introduced. Even though a conventional hidden Markov model when applied to the same dataset slightly outperformed our approach, its processing time is much higher. The Dynamic Multi-layer Perceptron presented here has an accuracy level of *96.94%* and runs significantly faster than similar techniques.

Keywords Discrete Fourier transform · Linear predictive cepstral coefficients · Mel-frequency cepstral coefficients · Speech recognition · Dynamic Multi-layer Perceptron · Hidden Markov models

✉ Mohammed Kyari Mustafa
mkmustafa@nda.edu.ng

Tony Allen
tony.allen@ntu.ac.uk

Kofi Appiah
kofi.appiah@ntu.ac.uk

[1] Department of Intelligence and Cyber Security, Nigerian Defence Academy, Kaduna, Nigeria

[2] School of Science and Technology, Nottingham Trent University, Nottingham NG11 8NS, UK

1 Introduction

Mobile speech recognition has become an everyday phenomenon. The predominant application of this technology can be found as Siri on the Apple IPhones, Cortana on Microsoft devices and Google talk on android devices. All three of these systems leverage a client-server approach to achieve recognition because the speech recognition process is computationally intensive. There are, however, limitations in the application of such technology. These limitations include, but are not limited to, the operational requirement for such systems to maintain a constant client-server connection and the need for efficient deployment of normalisation techniques to accommodate different speaker accents [1]. In addition, where the speech technology is used for security purposes, the need for the voice utterances to be transmitted over a network introduces security concerns. An alternative approach, where the complete speech recognition processing is performed entirely on the mobile device, is referred to as an on-device approach. This approach eliminates the connection-oriented problem and offers a more secure platform. An on-device system can also easily be adapted for embedded system use where sophisticated systems are required for disabled people.

The speech recognition process involves acoustic feature extraction and subsequent classification. The two main groups of classifiers used to achieve recognition are respectively statistical and probabilistic in nature.

The statistical method of interest in this paper is the hidden Markov model (HMM). HMMs are based on the concept of Markov chains which can represent any random sequential process that undergoes transitions from one state to another [2]. Due to its ability to model the phonetic transition of words, it has become widely accepted as the standard speech recognition technique in the speech recognition community. In this paper, the focus on HMMs is channelled towards their use for on-device speech recognition. The application of HMMs to speech recognition involves different training techniques. Conventional HMM methods apply K-means clustering to cluster observations into a set of transitions [3]. Further advancements involve the application of the Baum-Welch algorithm for better training [4]. Due to their acoustic modelling importance, support vector machines (SVMs) have also been applied [5]. HMMs equally evolved into continuous HMM, where the transitions between hidden states and the arrival of observations can occur at arbitrary times [6]. This gave rise to the application of different techniques in controlling unexpected transitions between two successive observations. These techniques include, but are not limited to, maximum likelihood estimation [7] and a semi-continuous approach, where the unexpected transitions are managed with the introduction of a language transition map [8]. The *Sphinx II* speech recognition system uses the latter approach and the *Pocketsphinx* speech recognition system, developed for use on some mobile devices, is an adaptation of this system [9].

In the case of probabilistic classifiers for speech recognition, different forms of artificial neural networks have been adopted. These range from standard feedforward neural networks to more complex recurrent neural networks [10–13]. The predominant feedforward neural network structure is the Multi-layer Perceptron (MLP) neural network. MLPs are fully connected networks with every node in the previous layer being connected to every node in the succeeding layer [10]. MLPs are also, by nature, feedforward networks where the direction of the connections of the respective units moves in one direction, from the input to the output, with no connections flowing backwards. As such, this renders MLPs to be static classifiers, which reflects how they are applied to speech recognition tasks. The adoption of recurrent neural networks allows the stochastic nature of speech signals to be automatically considered, something which static MLPs cannot model without the adoption of a segmentation algorithm. Speech recognition MLPs have been applied directly to control a mobile robot [10] and MLPs have been used to generate sets of features which are then used as inputs to other classifiers [12]. In [11], MLP is used together with a Self-Organising Map (SOM). However, in this case, the SOM is used to generate phoneme clusters as input features for the MLP. More recently, deep neural networks (DNNs) have been shown to present an improvement over classic MLPs [14].

This paper does not present a general comparison of HMM and NN techniques for speech recognition but instead compares the adaptation of these technologies for on-device mobile use or for embedded systems (with little or no internet connection). The speech recognition MLP, presented in this work, adopts a dynamic input structure to

- Automatically handle the issue of speech length variability
- Conserve processing time on the mobile device

The baseline HMM used as a comparison was also tuned for on-device embedded use by adopting positive integer calculations with fixed point notation.

2 Feature extraction

For every speech recognition task, the first step is the extraction of relevant speech features from the speech signal. In order to extract these speech features, the speech signal is first broken down into frames by applying a suitable window, such as the Hamming window [15]. The window is used to break the respective audio samples into frames of 16 ms each, this being within the 10–30 ms interval speculated as the time frame within which the vocal tract maintains a fixed set of characteristics [16]. There are no overlapping samples in the windows used in this work.

For comparison purposes, linear predictive cepstral coefficients (LPC) and Mel-frequency cepstral coefficients (MFCC) are extracted from the respective windowed speech samples. The extraction of these features is explained below:

1. LPC is a feature extraction technique within which a speech sample is approximated as a linear combination of past speech samples [17]. Linear prediction is mostly used in low bit rate transmission or storage of speech [16]. The LPC features are obtained by minimising the sum of squared differences between the actual speech samples and the linearly predicted ones. The LPC process has become the predominant technique for estimating basic speech parameters such as formants, spectra and vocal tract area functions [15]. The prediction of a speech sample $x(n)$ is given by Eq. 1:

$$x(n) = \sum_{i=1}^{p} a_i x(n-1) \tag{1}$$

where $x(n)$ is the sample being predicted, $x(n-i)$ is the previous sample, P is the order and α_i is the prediction coefficient.

2. MFCCs are the coefficients of a Mel-frequency cepstrum (MFC) that can be computed from any audio signal. Based on the Mel scale, the MFCCs are computed by applying a set of triangular band pass filters to the discrete Fourier transform (DFT) of a windowed sample and then taking a discrete cosine transform of the resulting logarithmic power spectrum [18]. Many speech recognition systems use the Mel-frequency cepstral coefficients and their first and sometimes second derivatives, as the input feature vectors of the speech signal. This helps to better reflect the dynamic changes of the speech [19]. The equation for converting a speech sample frequency component to their Mel scale equivalents is given by:

$$\mathrm{Mel}(f) = 1127\log\left(1+\frac{f}{700}\right) \quad (2)$$

where f is the frequency component to be converted to its corresponding Mel value.

2.1 Experimentation data

There are two databases used for this work and both contain strings of the spoken digits 1–9. The databases have different sampling rates and details of these two databases are as given below:

1. Center for Spoken Language Understanding (CSLU2002) database; a total of 20 speakers (11 females and 9 males) are used from this database. Each speaker utters the digits 1–9 for a total of 16 times. This database has a sampling rate of 8 kHz and 16 bit encoding. More details about this can be found in [20].
2. Texas Instruments and Massachusetts Institute of Technology Digits database (TIDIGITS); a total of 326 speakers (111 male, 114 females, 50 boys and 51 girls), divided equally into training and testing sets, were used from this dataset. The sampling rate for this database is 20 kHz with 16 bit encoding. More details about this can be found in [21].

3 Automatic speech recognition classifiers

There are two ways of presenting a speech signal to a recogniser. These two approaches respectively use either a phonetic or whole word approach. Both approaches are investigated in this work in order to address one of the fundamental characteristics of speech signals, namely, the issue of speech length variability. Speech length variability is the difference in the length of a speech signal with respect to spoken words [22]. Speech signal lengths are dependent on the individual uttering those words, and even the same person uttering the same word more than once does not guarantee that the duration of the speech will be constant for a given word. The Dynamic Multi-layer Perceptron Neural Network uses a whole word approach that is modified to take account of the variability in the length of the speech signals. The hidden Markov model approach combines both the word and phonetic approach, as it is designed based on a phonetic approach and this pays attention to the phonemes within the words.

3.1 Standard Multi-layer perceptron

MLPs are feedforward neural network structures that provide full connectivity between nodes in a previous layer and the nodes in the succeeding layer. Weights are used for the connection between the nodes of a previous layer and those of a succeeding layer (Fig. 1).

The calculations of the outputs of each node within the respective layers are the same. Equation 3 is used to calculate the output.

$$\mathrm{Net}_i = W_{i_0} + \sum_{j\in \mathrm{Pred}(i)} \left(W_{ij}a_j\right) \quad (3)$$

where a_j is the output of the preceding layer neuron, W_{ij} is the weight connecting the two neurons and W_{i0} is the bias of the neuron.

Backpropagation is one of the classic techniques used to train a MLP network. The backpropagation algorithm initially calculates the error at the output of the network using Eq. 4 and then propagates the error backwards to the hidden nodes where the errors at those nodes are calculated using Eq. 5. Equation 6 is then applied to update all the weights in the network.

$$\mathrm{error}_o = \mathrm{out}_o(1-\mathrm{out}_o)(\mathrm{do}_o-\mathrm{out}_o) \quad (4)$$

$$\mathrm{error}_j = \mathrm{out}_j\left(1-\mathrm{out}_j\right)\sum \mathrm{error}_o W(t)_{jo} \quad (5)$$

$$w(t+1)_{jo} = W(t)_{jo} + \beta \ \mathrm{error}_o \ \mathrm{out}_j + \tau\left[W(t)_{jo}-W(t-1)_{jo}\right] \quad (6)$$

where the subscript ***o*** is applicable to the output nodes and ***j*** is applicable to the hidden layer, error is the error at a particular node, *out* is the output node, *W*(*t*) is the weight, τ is the momentum factor and β is the learning rate. However, as explained earlier, momentum is not applied in our implementation.

3.2 Proposed Dynamic Multi-layer Perceptron implementation

Dynamism is the concept proposed to be applied to the standard MLP. Using a standard MLP would require a static number of input neurons that are then connected to the respective units of the network. For word-based speech recognition, all the MFCC feature vector frames, calculated for a given word, are simultaneously presented to the networks input layer. Generally, the input layer number is chosen, large enough to accommodate the largest set of input vectors that the network is expected to process. For smaller sized input vectors, the

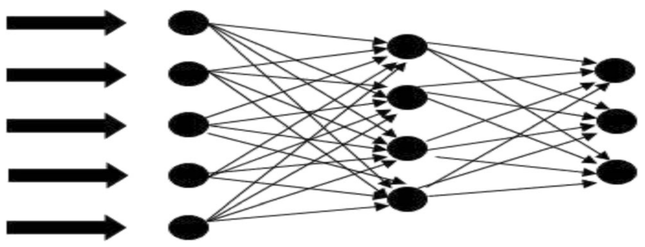


Fig. 1 Standard MLP structure

unused input neurons are binned by assigning 0 to them. In this way, these unused neurons do not affect the calculations within the network. However, whilst adopting such an approach standardises the inputs seen by the MLP networks, the additional calculations required for those unused inputs increase the processing time unnecessarily.

The idea behind the Dynamic MLP is to create a set of networks that can cater for inputs of variable length. By doing so, only the active neurons required for any calculation within the network are used. In this way, speech length variability is automatically accommodated and computational effect is minimised. This latter point is particularly important for on-device mobile deployment.

All calculations within these networks are the same for a standard MLP. Conventional backpropagation was used to train the network with the exception that only the weights processed during a given computation of network output were updated. This yielded faster calculations in terms of training the network.

3.3 Dynamic MLP experimentation

The highest number of frames encountered in the TIDIGITS database samples was 172 whilst for the CSLU database, the highest number of frames is 62. Consequently, the network was designed with a default input size of 172 frames resulting in an internal input size of 2064 (172 frames * 12 features per frame).

The pseudocode in Table 1 is used to pass a digit through the network. Word length variability is included in the network by allowing the entire digit to pass to the hidden units of the network using only the input to hidden weights corresponding to the size of the input pattern, i.e. each digit only

Table 1 Pseudocode of a forward pass through the Dynamic MLP

Dynamic MLP algorithm forward Pass

```
For every D with $F_N$ frames
    For the $N^{th}$F of D
    Arrange elements in ascending numbers of IU
    End
    Pass IU to X
    Propagate X to H using only connected $W^{in}$
    For every $h_n \in H$
    $Net1^i = f(h_n)$ as calculated using Eq. 6
    End
    Propagate $Net1^i$ to O using $W^{out}$
    For every $o^n \in O$
    Calculate $Net2^i$ using Eq. 6
    End
End
```

D digit, *F* frames, *IU* input unit, *X* input, *H* hidden layer, *Win* input weights, *Net1* network output at hidden layer, *O* output, W^{out} output weights, *Net2* network output at output layer

makes use of the number of nodes that reflect its length in the input layer. This adaptation significantly reduces the network calculation processing when compared to padding the input with zeros and still having to calculate for the nodes which are of no relevance to the required calculation.

The Dynamic MLP thus allows a digit to propagate from the input of the network all the way to the output of the network using only the number of connections it needs to get to the output. This concept makes the input layer dynamic and, at the same time, the weights connecting the input nodes to the hidden nodes of the network dynamic. However, the output nodes of the network remain the same because there are nine digits in the training data which the network is being trained to recognise. The output of the network thus represents the nine digits. Both the hidden and output layers have activation functions. The activation functions adopted for these layers are the hyperbolic tangent and linear function, respectively.

The network was trained using the backpropagation algorithm but without any momentum factor due to the dynamic nature of the input and hidden units. A fixed learning rate was also adopted to ensure uniformity in the calculation and to detect any problems that might arise from the network becoming unstable. The input, hidden and output layers were completely flushed out when a new digit was passed into the network. This ensures that a previous larger sample does not affect the calculation of a new and shorter sample.

In the DMLP, the backpropagation algorithm is modified so that only weights used by the network when a digit passed through are modified using the backpropagation algorithm. It can thus be said that dynamism is also implemented in the training procedure. This is also one of the reasons for adopting a fixed learning rate for the network. The parameters adopted for the network are also involved in the stopping conditions for the network. The maximum number of iterations used for the network is 3000 iterations and the minimum error used was 0.00001. This ensured a good balance between training and error calculations.

3.4 Hidden Markov model implementation

The algorithm used for the hidden Markov model is based on a readily available java library [23]. The HMM implementation initially uses K-means clustering algorithm to compute a HMM that does not depend on the temporal dependence of the observations. This is used as an initialiser. The Baum–Welch algorithm is then used in a loop to better train a HMM that models the speech signal. A total of nine HMMs are built for each set of utterances. During testing, the system is designed to calculate the Kullback-Leibler distance between the unseen observation and the trained set of HMMs. The HMM with the closest probability to the observation presented is chosen as the candidate HMM. This in turn reflects the digit.

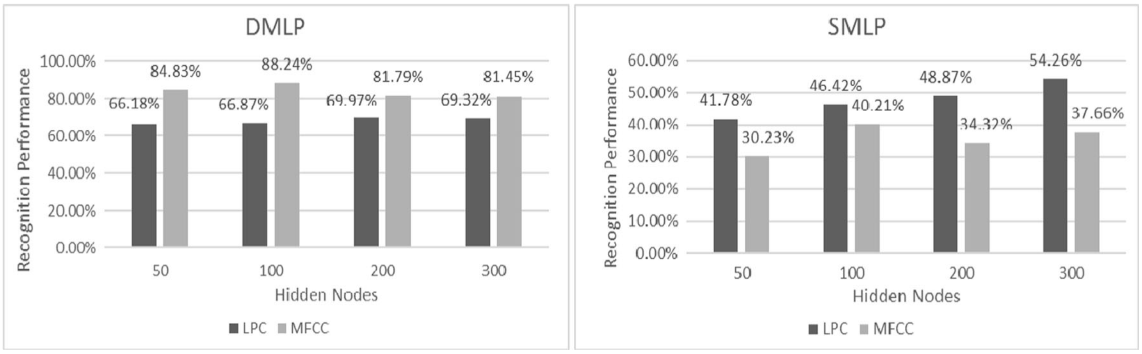


Fig. 2 Dynamic MLP vs standard MLP

4 Results

4.1 Dynamic Multi-layer Perceptron results

The Dynamic MLP was compared with a standard static input MLP for evaluation and the results of these are presented in Fig. 2.

The results presented here are for the TIDIGITS database because it is the only database without an overlap of speakers in the training and test files. These results show a better performance for the dynamic approach and, as such, further tests were carried out using the Dynamic MLP only. The results presented in Fig. 3 show a further set of tests on the Dynamic MLP with samples from both databases. The label 1 or 2 attached to the feature name is used to identify how the training data was presented to the network.

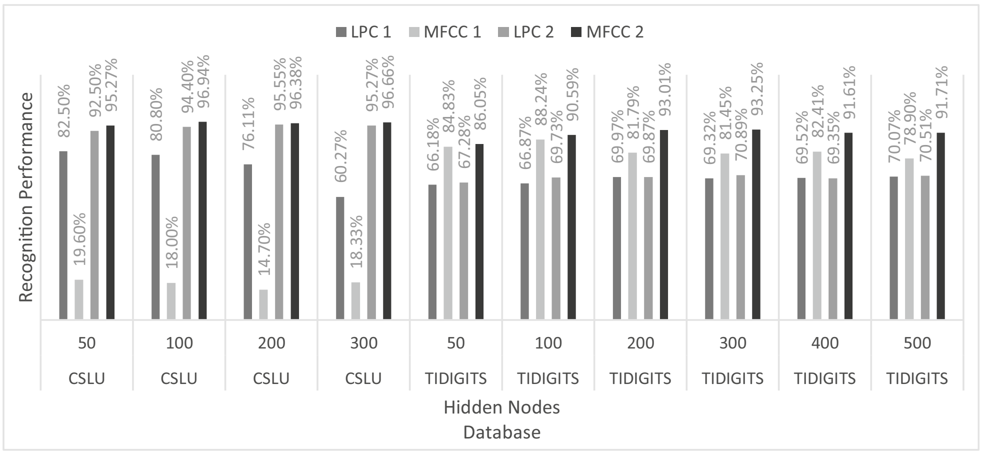
For label 1 samples, the training data was arranged in ascending digit number only, where all the feature vectors of the digit 1 are presented before those of 2, 3 and so on until digit 9. This ascension was defined by the digit number and not the number of frames contained in the digit. The number of frames therefore varies quite randomly during training.

In the second format, which is labelled 2 next to the feature name, the features for the training data are extracted and the digits are arranged in order of the ascending number of frames. For example, the digit with the lowest number of frames comes first, the digit with the second lowest number of frames second, all the way up to the digit with the largest number of frames. This arrangement does not take into consideration the position of the speaker in the database but just the number of frames per digit within the sample. Consequently, the arrangements of the speakers (male, female, boy and girl) are quite random because the first digit could have come from anyone of these speakers. This is not just limited to the speaker; it is also the same for the digit position. For example, in the training data of the TIDIGITS database, one sample representing the digit nine was the one having the lowest number of frames and as such it was the first one in the arrangement of the training data.

The results presented show that the second implementation favours the training of the network. This implementation appears to allow the network to learn the training data slowly over time as opposed to the much more random learning curve of the first implementation, where the first digit could be a digit with a higher or a lower number of frames. The second implementation guarantees that the training data is applied smoothly in such a way that no preceding digit in the training data set is larger in the number of frames than the succeeding digit. This can be seen in the results for the various formulations of the hidden units.

The best performances were for the MFCC 2 features with *96.94%* (CSLU100) and *93.25%* (TIDIGITS300), respectively. The best performance for the LPC 2 features was ***95.55%*** (CSLU200) and *70.89%* (TIDIGITS300). The relatively

Fig. 3 Results of the final set of experiments conducted using the Dynamic MLP



poorer performance of the TIDIGITS database in comparison to the CSLU database is probably due to two reasons:

1. There are no speaker overlaps in the TIDIGITS database in respect of the testing and training datasets whereas there are a few overlaps with regard to the CSLU database. The CSLU testing results therefore contain an element of previously seen samples. Secondly, the TIDIGITS database samples used for training and testing was not pre-processed to extract only the speech sections of the audio samples. Consequently, the TIDIGIT audio samples contained a lot of background silence. This adversely affected the training performance of the DMLP where the whole word, including background silence sections, is used to train the network.
2. The MFCC feature set performs better than the LPC feature set because the LPC features are created on the assumption that the vocal tract is a linear speech production mechanism. For the TIDIGIT samples, this effect is in addition to the LPC algorithm's difficulty with modelling segments of pure background silence.

4.2 HMM results

Figure 4 shows the hidden Markov model results for the same two databases. It can be seen that the best hidden Markov model outperforms the DMLP network results seen in Figs. 2 and 3. However, there is a crucial advantage to all the HMM systems which is not available to the DMLP. Within a HMM system, separate HMMs are created for each digit. This gives the HMM method a unique advantage over the neural network methods because a misclassification or wrong labelling by one HMM model need not necessarily affect the overall recognition performance of the HMM system. This is not the case with the neural network implementations where all the training sequences are used to train a single network. Any misclassification in this one network will result in a misclassification overall.

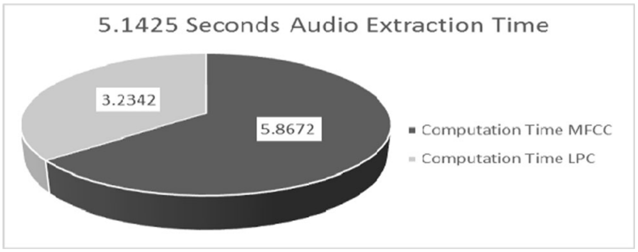


Fig. 5 Average computation time for 5.54125 audio sample

4.3 Memory and time comparison between DMLP and HMM

In this work, recognition performance is not the only parameter of interest. Speed of processing on the mobile device is equally important. It is worthy of note that the extraction of the respective features to be used for the automatic speech recognition classifier incurs their own computation time. For example, on the Samsung Note 3 [24], the following average feature extraction computation times were experimentally determined.

It is clear from Fig. 5 that it takes approximately a second to extract the MFCC features from a single second of speech data and about half a second to extract the LPC features from the same second of speech.

In addition to the feature extraction processing times, the results shown in Table 2 compare the respective computation processing times for the two classification methods. This computation time is taken after loading the respective audio feature vector data unto the mobile devices. This is to ensure that the test is a comparative test of the respective classification techniques alone.

Each test was run five times and an average taken. This is to ensure some degree of consistency in the computation times on the mobile devices. It should be noted that the respective mobile devices were not programmed to isolate other activities running on them. As such, the computation times shown include time taken by other background applications whilst the mobile device were performing the test.

Fig. 4 Hidden Markov model results for MFCC-based word recognition

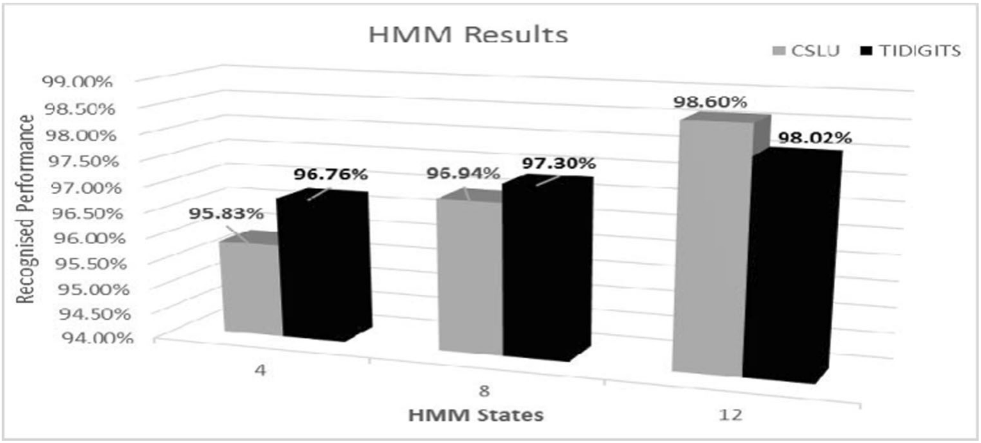


Table 2 Comparison of computation time in seconds for a total of 143.4 s of audio data for a range of mobile phones

Device	Processor	RAM	HMM 4	HMM 8	HMM 12	DMLP 50	DMLP 100	DMLP 200	DMLP 300
Galaxy S3	1.4 GHz (quad-core)	1 GB	66.1204	139.3366	201.893	0.5664	1.2288	2.3012	3.7752
Galaxy S4	1.9 GHz (quad-core)	2 GB	19.9746	38.8962	59.53	0.3326	0.6814	1.989	5.1172
Galaxy S5	2.5 GHz (quad-core)	2 GB	12.895	26.1608	39.593	0.1512	0.3376	0.6988	1.3744
Note 2	1.6 GHz (quad-core)	2 GB	84.6602	106.9112	221.2478	0.3426	0.7134	1.4506	2.3222
Note 3	2.3 GHz (quad-core)	3 GB	12.1884	26.7236	35.592	0.207	0.441	1.2524	2.3436

Table 3 Memory size for the different applications on the mobile device

HMM 4	HMM 8	HMM 12	DMLP 50	DMLP 100	DMLP 200	DMLP 300
752 KB	805 KB	861 KB	3.25 MB	4.25 MB	6.28 MB	8.34 MB

The mobile devices used are good representatives of the range of readily available common mobile devices, currently in circulation, that are based on the android platform. The numbers succeeding the HMM label represent the number of states and the numbers succeeding the DMLP label represent the number of hidden units.

In Table 2, it can clearly be seen that the four-state HMM system is the closest in time performance to the DMLP system. However, whilst the best performing HMM system (HMM12) may have a superior recognition performance to the best performing DMLP system (DMLP 100), the time involved in computing the 12-state HMM output can be seen to be one to two orders of magnitude greater than the DMLP computation times. On mobile devices, this processing time differential is a cardinal advantage for the DMLP method. Indeed, none of the DMLP architecture performs slower than any of the HMM architectures.

In Table 3, the sizes of memory used by the different techniques are given. For the HMM, this includes the trained HMMs for the respective digits. On the other hand, the DMLP includes the trained weights of the network. The highest memory used is the DMLP with 300 hidden units. However, this is 8.34 MB in total; this is not very much in terms of current mobile devices and especially the mobile devices used for this experiment.

4.4 Deep neural networks

In order to try and improve the performance of the DMLP network, particularly on the TIDIGIT database, further experiments were conducted using a Deep Neural Network approach. However, these experiments were conducted on a computer, rather than mobile phone, as initial test runs of the deep neural network experiments were found to be computationally intensive even on a computer level.

Although the results shown in Fig. 6 do show a slight improvement over the DMLP LPC2 results shown in Figs. 2 and 3, the substantially longer processing times again make them impractical for implementation on a mobile phone.

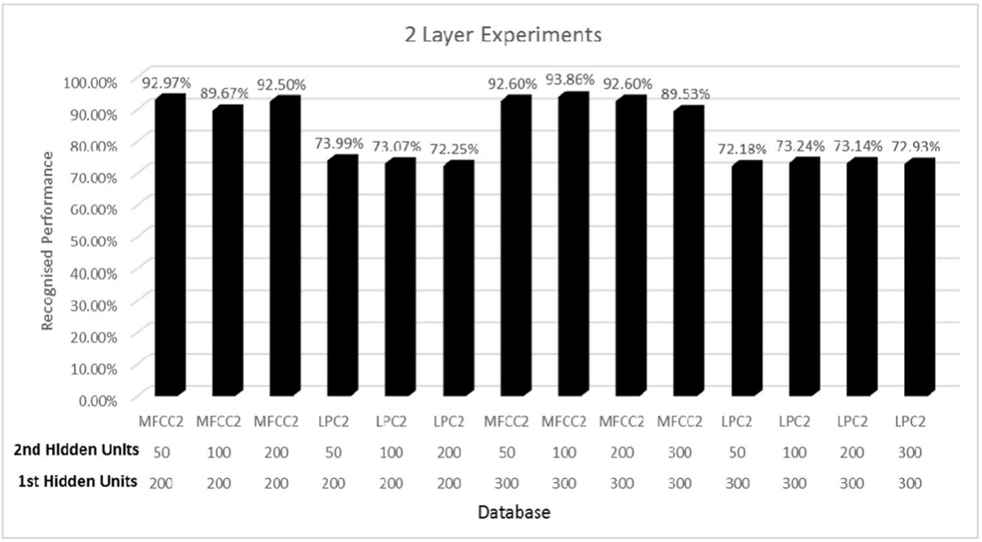


Fig. 6 Further DMLP experiments using two hidden layers

5 Conclusion

Two automatic speech recognition approaches adapted for use on mobile devices are presented. The Dynamic MLP shows a novel approach to speech recognition by adapting a conventional feedforward neural network architecture to the speech recognition process as opposed to vice versa. This method can accommodate as input, the variable length output of voice activity detection (VAD) algorithms and classify them without any modifications to the input length.

Although the HMM methods do outperform the Dynamic MLP in terms of absolute recognition performance, it can be seen from Table 3 that all the HMMs take substantially longer to compute than the DMLP system. Using the time for the best performing network structure of the Dynamic MLP, it would be safe to say that the marginal recognition performance advantage of *1.66%* for the HMM is compromised by the far longer processing time required to achieve this improvement. Indeed, the Dynamic MLP computation time is *8935.4%* faster than the HMM for each second of the test data. This is much too large a computational time advantage to ignore for the sake of a *1.66%* performance advantage.

The computation time for the HMM is also likely to grow exponentially over more sets of target data because HMMs are designed and implemented for isolated recognition in such a manner that a HMM representation of every word is created. To test a HMM for recognition of any unknown word is to test the word against each HMM implementation of the respective training data. This incurs an additional computation time which would grow exponentially. Although the computation time of the neural network would also increase as the network size increases, this increase is expected to be significantly less for the equivalent HMM system as the same single network is used to process all digits. However, future work to qualify this assertion would be necessary.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Rose RC, Arizmendi I (2006) Efficient client–server based implementations of mobile speech recognition services. Speech Commun 48:1573–1589
2. Rabiner LR (1989) A tutorial on hidden Markov models and selected applications in speech recognition. Proc IEEE 77: 257–286
3. Juang B-H, Rabiner LR (1990) The segmental K-means algorithm for estimating parameters of hidden Markov models. Acoustics, Speech and Signal Processing, IEEE Transactions on 38(9):1639–1641
4. Audhkhasi Kartik, Osoba Osonde, Kosko Bart (2013) Noisy hidden Markov models for speech recognition. In: Neural Networks (IJCNN), The 2013 International Joint Conference on, IEEE, p 1–6
5. Zarrouk E, Ayed YB, Gargouri F (2014) Hybrid continuous speech recognition systems by HMM, MLP and SVM: a comparative study. International Journal of Speech Technology 17(3):223–233
6. Liu Yu-Ying, Li Shuang, Li Fuxin, Song Le, Rehg James M. (2015) Efficient learning of continuous-time hidden markov models for disease progression. In: Advances in neural information processing systems, pp 3600–3608
7. Bietti Alberto, Francis Bach, and Arshia Cont (2015). An online EM algorithm in hidden (semi-) Markov models for audio segmentation and clustering. In: 2015 I.E. International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, p 1881–1885
8. Huang X, Alleva F, Hwang M-Y, Rosenfeld R (1993) An overview of the SPHINX-II speech recognition system. In: Proceedings of the workshop on Human Language Technology. Association for Computational Linguistics, Stroudsburg, pp 81–86
9. Huggins-Daines David, Kumar Mohit, Chan Arthur, Black Alan W., Ravishankar Mosur, Rudnicky Alexander I. (2006) Pocketsphinx: a free, real-time continuous speech recognition system for hand-held devices. In: 2006 I.E. International Conference on Acoustics Speech and Signal Processing Proceedings, vol. 1, IEEE, p I-I
10. Wijoyo S. (2011) Speech recognition using linear predictive coding and artificial neural network for controlling movement of mobile robot. In: Proceedings of 2011 International Conference on Information and Electronics Engineering (ICIEE 2011), p 28–29
11. Eng Goh Kia, Ahmad Abdul Manan (2005). Malay speech recognition using self-organizing map and multilayer perceptron. Proceedings of the Postgraduate Annual Research Seminar
12. Park Junho, et al (2009) Training and adapting MLP features for Arabic speech recognition. Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on. IEEE
13. Hmad N, Allen T (2013) Echo State Networks for Arabic Phoneme Recognition. World Acad Sci Eng Tech Int J Comput Electr Autom Control Inform Eng 7(7):904-910
14. Seltzer Michael L, Yu Dong, Wang Yongqiang (2013) An investigation of deep neural networks for noise robust speech recognition. In: 2013 I.E. International Conference on Acoustics, Speech and Signal Processing, IEEE, p 7398–7402
15. Hai J, Joo EM (2003) Improved linear predictive coding method for speech recognition: information, communications and signal processing, 2003 and Fourth Pacific Rim Conference on Multimedia. Proceedings of the 2003 Joint Conference of the Fourth International Conference on, vol. 3, IEEE, p 1614–1618
16. Rabiner LR, Schafer RW (1979) Digital processing of speech signals. Institution of Engineering and Technology
17. Vaidyanathan P (2007) The theory of linear prediction. Synthesis Lectures on Signal Processing 2:1–184
18. Kamm T, Hermansky H, Andreou AG (1997) Learning the Mel-scale and optimal VTN mapping: Center for language and speech processing, workshop (WS 1997). Johns Hopkins University, Citeseer
19. Plannerer Bernd (2013) An introduction to speech recognition. 28 March (2005). http://spoken-number-recognition.googlecode.com/svn/trunk/docs/introSR.pdf. Accessed 17 Jun 2013
20. CSLU (2002) database Available online at https://www.cslu.ogi.edu/corpora/spkrec/
21. TIDIGITS (1993) database details Available online at https://catalog.ldc.upenn.edu/topten

22. Benzeghiba M et al (2007) Automatic speech recognition and speech variability: a review. Speech Commun. 49:763–786
23. Francois JM (2016) Jahmm-hidden Markov model (HMM): an implementation in Java. http://jahmm.googlecode.comAccessed on 12th January
24. Samsung Note 3 Specifications. Available Online at http://www.gsmarena.com/samsung_galaxy_note_3-5665.php . Accessed 12th Jan. 2016