Interspeech 2018
2-6 September 2018, Hyderabad, India

Efficient Voice Trigger Detection for Low Resource Hardware

Siddharth Sigtia, Rob Haynes, Hywel Richards, Erik Marchi, John Bridle

Siri Speech, Apple

{sidsigtia,r_haynes,h_richards,emarchi,john.bridle}@apple.com

Abstract

We describe the architecture of an always-on keyword spotting (KWS) system for *battery-powered* mobile devices used to initiate an interaction with the device. An always-available voice assistant needs a carefully designed voice keyword detector to satisfy the power and computational constraints of battery powered devices. We employ a multi-stage system that uses a low-power primary stage to decide when to run a more accurate (but more power-hungry) secondary detector. We describe a straightforward primary detector and explore variations that result in very useful reductions in computation (or increased accuracy for the same computation). By reducing the set of target labels from three to one per phone, and reducing the rate at which the acoustic model is operated, the compute rate can be reduced by a *factor of six* while maintaining the same accuracy.
Index Terms: Speech Recognition, Keyword Spotting, Deep Neural Networks, Acoustic Modelling, Embedded Devices

1. Introduction

There is increasing interest in hands free, voice-first interfaces to automated assistants such as Siri, Alexa and the Google Assistant. These systems rely on a voice trigger system to detect a key phrase (e.g. *Hey Siri* for Apple devices) that initiates an interaction with the device. The voice trigger detector is a kind of keyword spotter, that continuously receives a stream of audio from the microphone and listens only for the trigger phrase at the beginning of an utterance. A detection causes the device to change state and initiate a further interaction with the user. When the device is battery powered like the iPhone or the Apple Watch, it is imperative that the voice trigger detector consume as little power as possible while still maintaining sufficient accuracy. In recent iPhone designs this is achieved by running a primary detector on a low-power processor that runs even when the main processor is asleep [1]. This primary detector can decide to wake the main processor, where further checks are done (on the same waveform) before the main recognizer is applied and the identity of the speaker is confirmed [2]. This arrangement is efficient in battery use: power consumption depends on the continuing requirements of the primary detector plus the *false-wake* rate of the primary detector and the power cost of waking the main processor. It can produce good accuracy: the false-alarm rate of the system is determined not by the primary but by subsequent components. Furthermore, considerations of privacy mean that we want to reduce to a minimum the number of times that audio gets sent to a server in error.

Some older approaches to KWS adapted large vocabulary continuous speech recognition (LVSCR) systems for detecting keywords [3, 4], however these methods use large acoustic and language models which are unsuitable for on-device applications. In more recent approaches to KWS, by limiting the number of keywords that the system is trained to detect, it is possible to design systems that are small enough to be run efficiently on low-power mobile and embedded devices. Recent studies investigate several different approaches to this problem; some methods employ the conventional DNN-HMM approach for computing a detection score [5, 6], other feed-forward methods employ DNNs to compute frame-wise posteriors and then temporally smooth the posteriors to compute keyword detection scores [7, 8, 9], while some approaches use recurrent models [10, 11, 12, 13, 14]. In this paper we present a DNN-HMM model for detecting the trigger phrase *Hey Siri* specifically, however the methods developed are general.

In this paper we focus only on the primary detector which runs continuously on a low-power, low resource, always-on processor where computation and memory are the limiting factors. The rest of the paper is organised as follows: in Section 2, we outline the DNN-HMM KWS system, describe the structure of the acoustic model and provide details of the training and evaluation setup. In Section 3, we investigate the effect of changing the DNN targets and the effect of minimum durations of the HMM states on detection accuracy. In Section 4 we demonstrate that the detector DNNs can be run at lower frame-rates compared to the baseline without compromising accuracy. Section 5 provides a summary and conclusions from the results.

2. System Outline

Figure 1 provides an overview of the detector. The microphone receives a continuous stream of audio which is transformed into a stream of feature vectors by the front-end acoustic analysis. In order to limit the number of inputs to the DNN and therefore the size of the input layer, we compute 13-dimensional MFCCs using a window of 25ms at a rate of 100 frames per second (FPS). At a given time-step we supply 19 consecutive frames to the DNN which is trained to predict the label for the centre frame. The DNN comprises a stack of fully connected layers followed by a softmax layer [15] at the output. We use the same number of units in each of the hidden layers. Throughout the paper, the architecture of a DNN is denoted by $d \times w$, where d represents the number of hidden layers and w represents the number of units in each layer. We did not find a significant difference in accuracies between using sigmoid and ReLU activations for the hidden units. We therefore use sigmoid activations for the hidden units since the values are bounded between $(0, 1)$ which simplifies conversion of the weights to fixed-point values (see Section 2.2).

The output softmax layer contains 20 units; the phrase *Hey Siri* contains 6 phones, each of which are further divided into 3 states. The 2 remaining output units correspond to silence and a background/filler state. At every frame, the DNN outputs are divided by state priors to yield scaled likelihoods $\hat{y}_n$ [16]. A Dynamic Programming process accumulates scores for the best way to get to each state by multiplying state scores with the probability of staying or transitioning to a state and taking the maximum over the 2 ways of arriving at a state. The observation probabilities are then added to the state scores. Finally for every

10.21437/Interspeech.2018-2204

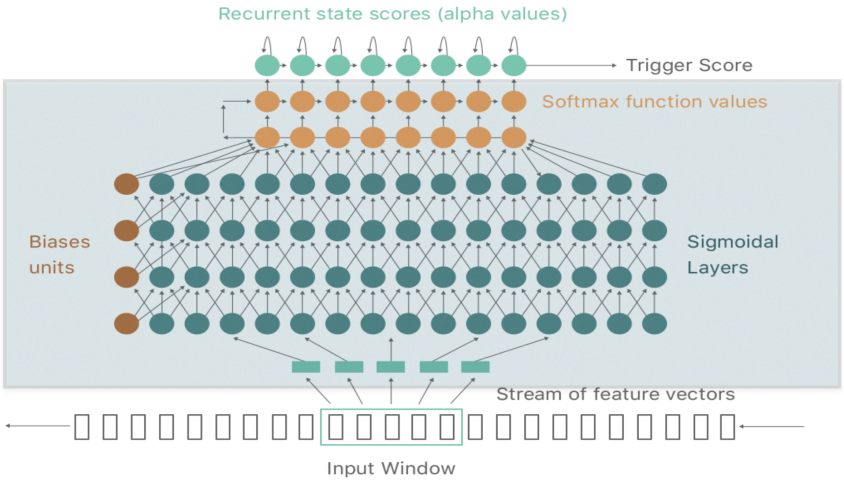

Figure 1: *An overview of the DNN-HMM KWS system. A window of frames from a continuous stream of features is input to the DNN at every time step. The DNN comprises a stack of full connected sigmoidal layers (some connections have been omitted for clarity) and a softmax output. The posteriors are then used to compute α values for the HMM states and finally a score R_t is computed at each frame.*

frame, we calculate the ratio of the log-likelihood of the score for the final state of the keyword HMM and the filler HMM:

$$R_t = \log \frac{P(\mathbf{x}_t|\theta_{KW})}{P(\mathbf{x}_t|\theta_F)},$$

where $\mathbf{x}_t$ is a stream of input features and θ_{KW}, θ_F are the parameters of the keyword and filler HMMs, respectively. A detection is made if the score R_t is above a set threshold.

2.1. Model Training

The DNN acoustic model (AM) is trained by minimising the framewise cross-entropy loss (or maximizing the conditional likelihood of the training labels):

$$C = -\frac{1}{N}\sum_{n=1}^{N}\left[\log \hat{y}_{nc_n}\right],$$

where c_n is the target label for input x_n, $\hat{y}_{nc_n}$ is the model output for the nth frame and c_nth class and N is the total number of examples in a mini-batch. The DNN weights are initialised randomly by sampling from a uniform distribution according to the procedure outlined in [17]. The model parameters are optimised using stochastic gradient descent (SGD). We use randomly sampled minibatches of 128 examples and training is initialised with a learning rate of 0.01. One epoch is defined as 10,000 gradient updates and the learning rate is decayed by a factor 0.96 at the end of every epoch. We use a constant momentum rate of 0.9 and the models are trained until convergence. We monitor the progress of training on a held-out fraction of the training set.

2.2. Quantisation

The DNNs AMs are trained on one or more GPUs and the weights and activations of the networks are represented by 32-bit floating point numbers. However when deployed on low-power, low-memory mobile devices, storing floating point weights can take up a lot of memory while floating point arithmetic operations consume a lot of power. Quantising the weights and activations to 8-bit fixed point integers reduces model size by 75% and reduces power consumption. Additionally, parallel 8-bit operations are available for certain processors. Therefore after training, we quantise the model weights to 8-bit fixed point integer values for inference on the device. We use a similar scheme as in [18], but with a floating-point sigmoid. We also quantise the first layer as it comprises a large proportion of the whole network, in this case adding an elementwise pre-sigmoid scaling to help accommodate the wider range of weights there.

2.3. Dataset

The training dataset contains approximately a million utterances with the trigger phrase *Hey Siri*. The training utterances are augmented with reverberations and varying noise levels to simulate a range of real world conditions. The evaluation dataset contains 4000 utterances with the trigger phrase and 2000 hours of speech data in a range of noise conditions without the trigger phrase. During training we use 1000 utterances from the training set as a validation set and the remaining utterances for parameter estimation. Frame-wise labels for all the utterances are generated via forced alignment using an LVCSR system. The context-dependent phonestate labels are mapped to phrase-specific labels, silence and a filler label for the remainder.

2.4. Evaluation

Results for the baseline DNN-HMM KWS system are presented in the form of modified detection error trade-off (DET) curves, where the Y-axis represents false reject (FR) rate and the X-axis represents number of false accepts (FAs) per hour (lower is better). Points on the curve correspond to different threshold values and the curve is used to set an operating point for a given device and resource limitations. The accuracies of models are compared at the desired FA/hour value. We train DNNs with $d = 5$ layers and a range of hidden units in each layer, $w \in 32, 48, 128, 192$.

From Figure 2 we note that the accuracies of the detectors improve as the number of parameters in the model are increased. Although we observe a significant improvement in accuracy with larger models (e.g. the 5x192 DNN), they contain an order

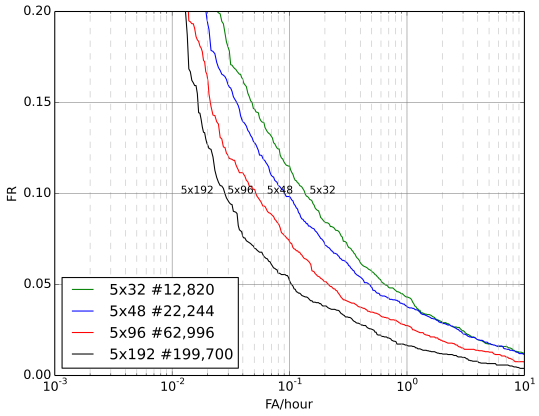

Figure 2: *Modified DET curves over a range of DNN architectures and parameter counts.*

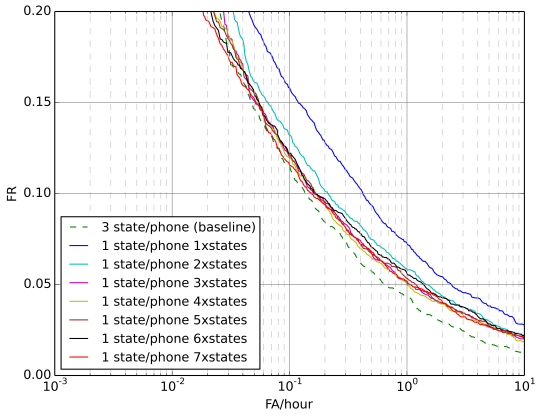

Figure 3: *DET curves for 5x32 whole phone DNN models with varying minimum duration constrains.*

of magnitude more parameters compared to the smaller models. These large DNNs cannot be continuously run within the power and compute budget of battery powered devices like the iPhone and the Apple Watch. Therefore for the first pass we typically use very small DNNs with a total number of parameters $< 15k$. Note that for smaller models, the input layer dominates computation. For example for the 5x32 DNN, the input layer accounts for roughly half the parameters and computation in the model.

3. Whole Phone Models

The KWS system described above follows the conventional DNN-HMM approach where the front-end calculates MFCCs every 10 ms at 100 FPS and the target labels for each phone are divided into 3 states for the beginning, middle and end of the segment. The scaled conditional probabilities from the AM for each frame are then multiplied (or rather their log-values are added). At each frame the system reports a score for the labelling of the preceding frames that ends with the states of the trigger phrase in their sequence, arranged so as to maximise the score. More recently, it has been demonstrated that LVCSR systems trained to predict whole phone labels (single label per phone) can achieve accuracies similar to conventional systems with 3 labels per phone. This result has been demonstrated for both LSTM RNNs trained with cross-entropy [19] and with the CTC criterion [20].

We repeat the experiments in Section 2, but with a single target label per phone instead of 3. The DNNs now comprise 8 output labels: 1 for each phone in *Hey Siri*, one label for silence and a label for background noise/filler. We map the alignments obtained previously to the new set of labels and retrain the DNNs with exactly the same training setup and parameters as before without any change. Figure 3 shows a comparison between the baseline 5x32 DNN and the DNN with whole phone targets with the same architecture. We observe a significant loss in accuracy compared to the baseline. There are several possible reasons for the drop in accuracy, including: a) multiplication of probabilities for the same hypothesis conditioned on overlapping evidence is a bad idea, b) we need a minimum duration constraint that each phone must explain at least 3 consecutive frames, c) phonetic segments really do need to be further divided into 3 states. We evaluate hypothesis b) by using the whole phone DNN AM but by replicating each state in the

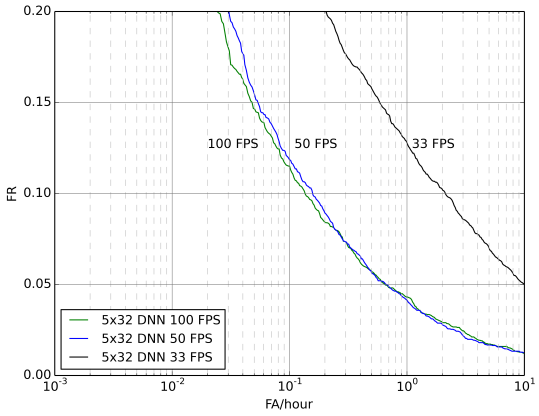

Figure 4: *Baseline 5x32 DNN with 3 states/phone operated at lower frame-rates.*

trigger phrase HMM by a factor multiple, which is equivalent to imposing a minimum duration on each of the labels (Figure 3). We observe that we are able to achieve similar accuracies as the baseline with the additional minimum duration constrains. These observations are similar to the ones made for an LSTM based LVCSR system with whole phone units [19]. Additionally, rather than simply replicating HMM states by the same factor, duration models for each state can be learnt independently.

4. Low Frame Rate Models

There is an alternative way to impose longer minimum durations for each state: run the detector at a lower rate than 100 FPS. This results in longer intervals between predictions, which effectively increases the minimum duration of the HMM states. Previously, LSTM RNN models trained with the CTC objective with context dependent (CD) whole phone targets have been shown to work well when running the AM at 33 FPS [21]. It is hypothesised that this works for 2 reasons. Firstly, the LSTM RNN models maintain an internal state, which helps them make predictions conditioned on longer input time-scales. Secondly, making fewer predictions reduces the space of pos-

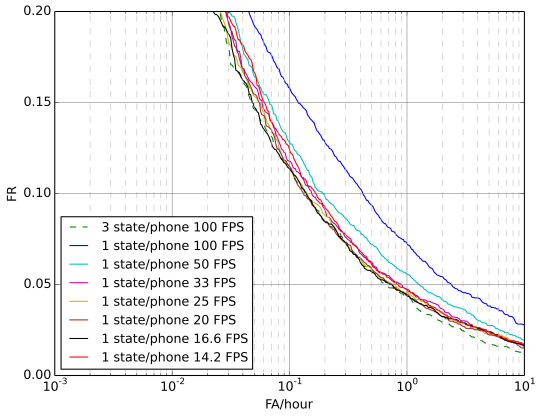

Figure 5: *DET curves for the whole phone DNNs at varying frame-rates.*

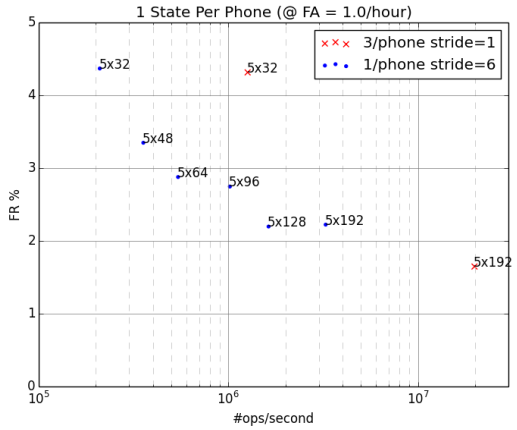

Figure 6: *FR rate calculated at 1 FA/hour as a function of operations/second. The blue 5x32 model yields the same accuracy at a much lower compute compared to the red 5x32. While the blue 5x{48,64,96} models yield better accuracies while still at lower compute than the baseline model.*

sible alignments that the CTC cost function sums over and improves model training [21]. More recently, it has been shown that LSTM RNN models trained with the cross-entropy criterion can also be operated at lower frame rates when trained to output phone labels [22].

For on-device KWS, operating the detectors at a lower frame-rate is an attractive route for trying to limit the computation performed by the system. Although previous studies show good results with LSTM RNNs, for the power/accuracy regime of interest, we find simple DNN AMs most effective. In experiments not reported here, we found no advantage in accuracy from using an LSTM with a single layer and 32 cells which contains 3 times the number of weights and computations as a 5x32 DNN. We compensate for the lack of recurrent connections by supplying the DNNs with 19 consecutive frames as input (as before) or roughly 200 ms of audio, which is typically longer than the duration of a phone. In order to run the models at lower frame-rates, we still perform the front-end computation at 100 FPS and stack 19 frames, but we move along to the next

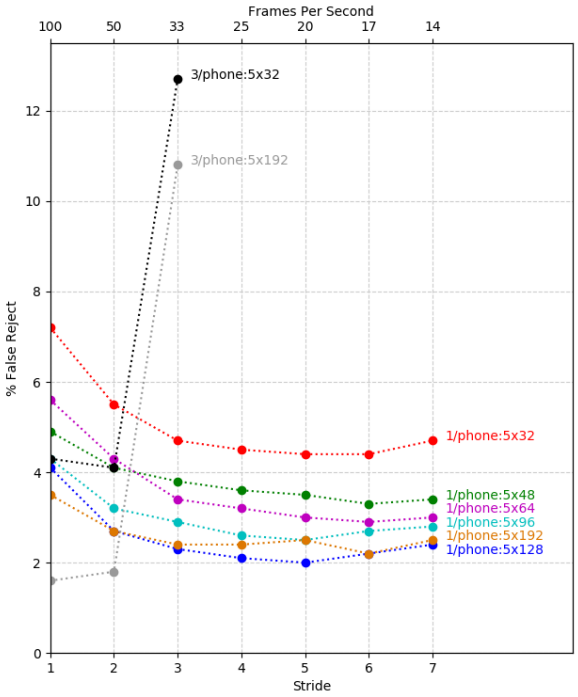

Figure 7: *FR rate as a function of the stride between consecutive input windows @ 1 FA/hour.*

window with a stride $s \in \{2, 3, 4, 5, 6, 7\}$ rather than $s = 1$. Figure 5 shows the results from running the 5x32 DNN model at varying frame-rates. We observe that the model yields similar accuracies as the baseline 3 states/phone DNN at a frame-rate of up to 16.6 FPS. We also note that accuracies increase with increasing stride up to $s = 5$. This reduces the computation performed by the system by a factor of 6 without compromising accuracy. Alternatively, it allows us to use a model with 6 times as many parameters, which roughly corresponds to the number of parameters in a 5x96 DNN. The 5x96 DNN at 16.6 FPS yields 40% relative improvement in FR rate at 1 FA/hour compared to the baseline 5x32 DNN detector at 100 FPS, for roughly the same amount of compute. Figure 6 shows the FR rate as a function of the operations/second while Figure 7 shows the FR rate calculated at 1 FA/hour as a function of the stride. We note the results are consistent over a range of model sizes.

5. Conclusion

We presented a DNN-HMM system for on-device KWS in low-resource conditions. Our results demonstrate that for a voice trigger detection problem it is not necessary to divide phone labels into 3 states for the beginning, middle and end of each phone. We are able to achieve similar results to the baseline with a single label per phone and minimum duration constrains. This principle has been previously demonstrated for LVSCR with LSTM AMs, but our results demonstrate that the same holds true for DNN AMs with large input windows. As a practical consequence, we are able to run the detectors at frame-rates as low as 16.6 FPS without any loss in accuracy compared to the baseline. This represents a factor of 6 reduction in computation, which is significant when the system is deployed on low resource hardware. Alternatively we can run a detector 6 times as large as the baseline without any extra computation.

6. References

[1] Apple Machine Learning Blog, "Hey Siri: An On-device DNN-powered Voice Trigger for Apples Personal Assistant," https://machinelearning.apple.com/2017/10/01/hey-siri.html, October 2017.

[2] E. Marchi, S. Shum, K. Hwang, S. Kajarekar, S. Sigtia, H. Richards, R. Haynes, Y. Kim, and J. Bridle, "Generalised discriminative transform via curriculum learning for speaker recognition," in *Proceedings IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP*. Calgary, Canada: IEEE, April 2018, pp. –, to appear.

[3] J. R. Rohlicek, W. Russell, S. Roukos, and H. Gish, "Continuous Hidden Markov Modeling for Speaker-Independent Word Spotting," in *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. IEEE, 1989, pp. 627–630.

[4] R. C. Rose and D. B. Paul, "A Hidden Markov Model Based Keyword Recognition System," in *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. IEEE, 1990, pp. 129–132.

[5] S. Panchapagesan, M. Sun, A. Khare, S. Matsoukas, A. Mandal, B. Hoffmeister, and S. Vitaladevuni, "Multi-Task Learning and Weighted Cross-Entropy for DNN-Based Keyword Spotting," in *INTERSPEECH*, 2016, pp. 760–764.

[6] M. Sun, D. Snyder, Y. Gao, V. Nagaraja, M. Rodehorst, N. S. Panchapagesan, S. Matsoukas, and S. Vitaladevuni, "Compressed Time Delay Neural Network for Small-footprint Keyword Spotting," *INTERSPEECH*, pp. 3607–3611, 2017.

[7] G. Chen, C. Parada, and G. Heigold, "Small-Footprint Keyword Spotting Using Deep Neural Networks," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2014, pp. 4087–4091.

[8] T. N. Sainath and C. Parada, "Convolutional Neural Networks for Small-Footprint Keyword Spotting," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[9] G. Tucker, M. Wu, M. Sun, S. Panchapagesan, G. Fu, and S. Vitaladevuni, "Model Compression Applied to Small-Footprint Keyword Spotting." in *INTERSPEECH*, 2016, pp. 1878–1882.

[10] S. Fernández, A. Graves, and J. Schmidhuber, "An Application of Recurrent Neural Networks to Discriminative Keyword Spotting," in *International Conference on Artificial Neural Networks*. Springer, 2007, pp. 220–229.

[11] M. Woellmer, B. Schuller, and G. Rigoll, "Keyword Spotting Exploiting Long Short-term Memory," *Speech Communication*, vol. 55, no. 2, pp. 252–265, 2013.

[12] S. O. Arik, M. Kliegl, R. Child, J. Hestness, A. Gibiansky, C. Fougner, R. Prenger, and A. Coates, "Convolutional Recurrent Neural Networks for Small-Footprint Keyword Spotting," *arXiv preprint arXiv:1703.05390*, 2017.

[13] M. Sun, A. Raju, G. Tucker, S. Panchapagesan, G. Fu, A. Mandal, S. Matsoukas, N. Strom, and S. Vitaladevuni, "Max-pooling Loss Training of Long Short-term Memory Networks for Small-footprint Keyword Spotting," in *Spoken Language Technology Workshop (SLT), 2016 IEEE*. IEEE, 2016, pp. 474–480.

[14] Y. He, R. Prabhavalkar, K. Rao, W. Li, A. Bakhtin, and I. McGraw, "Streaming Small-Footprint Keyword Spotting using Sequence-to-Sequence Models," in *IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2017, pp. –.

[15] J. S. Bridle, "Probabilistic Interpretation of Feedforward Classification Network Outputs, with Relationships to Statistical Pattern Recognition," in *Neurocomputing*. Springer, 1990, pp. 227–236.

[16] H. A. Bourlard and N. Morgan, *Connectionist Speech Recognition: A Hybrid Approach*. Springer Science & Business Media, 2012, vol. 247.

[17] X. Glorot and Y. Bengio, "Understanding the Difficulty of Training Deep Feedforward Neural Networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 2010, pp. 249–256.

[18] V. Vanhoucke, A. Senior, and M. Z. Mao, "Improving the speed of neural networks on cpus," in *Deep Learning and Unsupervised Feature Learning Workshop, NIPS 2011*, 2011.

[19] A. Senior, H. Sak, and I. Shafran, "Context Dependent Phone Models for LSTM RNN Acoustic Modelling," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2015, pp. 4585–4589.

[20] H. Sak, A. Senior, K. Rao, O. Irsoy, A. Graves, F. Beaufays, and J. Schalkwyk, "Learning Acoustic Frame Labeling for Speech Recognition with Recurrent Neural Networks," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2015, pp. 4280–4284.

[21] H. Sak, F. de Chaumont Quitry, T. Sainath, K. Rao *et al.*, "Acoustic modelling with CD-CTC-SMBR LSTM RNNs," in *IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*. IEEE, 2015, pp. 604–609.

[22] G. Pundak and T. N. Sainath, "Lower Frame Rate Neural Network Acoustic Models," in *INTERSPEECH*, 2016, pp. 22–26.