AUTOMATIC SPEECH RECOGNITION FOR REAL TIME SYSTEMS

Hemant Yadav[*1], *Mohit Sharma*[*2], *Ranjodh Singh*[*2] , *Sandeep Gosain*[1], *Rajiv Ratn Shah*[2]
1. Humonics Global Pvt. Ltd.
2. MIDAS Lab, IIIT Delhi

ABSTRACT

Automatic Speech Recognition (ASR) systems have proven to be a useful tool to perform various day to day operations when used along with systems like Personal AI Assistants. Various industries require ASR to be trained in their domains. Music on Demand (MoD), over IVR, is one such industry where the user interacts with the dialogue system to play music using voice commands only. Domain adaptation of the model is expected to perform well on this domain as systems trained on public datasets are very generic in nature and not very domain-specific. To train the ASR for MoD, we experiment with the HMM-based classical approach and DeepSpeech2 on Voxforge dataset. We then fine-tune the DeepSpeech2 model on MoD data. With very limited data and little finetuning of the model, we were able to achieve 14.727% Word Error Rate (WER).

Index Terms— DeepSpeech2, DeepLearning, HMMs, Automatic Speech Recognition, Domain Adaptation, Language Model(LM)

1. INTRODUCTION

Automatic Speech Recognition (ASR) systems have been around for a long time. However, due to recent advancements in computation power and innovations in algorithms, now it is being used in different industries, like Healthcare [1], Defence, Education [2], where it could not be imaginable some time ago. Most of the recent ASRs are trained on openly available generic datasets like the Fisher Corpus [3] and Librispeech [4], which are not domain-specific and prone to a substantial amount of noise. In the ASR system, there are two main blocks: Acoustic model and Language model(LM), which work hand in hand to get the required transcriptions. There is always a scarcity, in industry, for an ASR system that could cater to the needs of automatic transcription using domain-specific language models. This is due to the limited availability of labeled data. However, there is still a very genuine need of ASR systems which are domain and industry-specific as they are more accurate and reliable compared to a generic ASR that is trained on the multi-domain dataset and using a language model that is not domain specific.

*Authors contributed equally

Countries where 3G/4G technologies are not fully adopted by people, still use a dial plan to listen to music. With over a hundred thousand candidate songs, selection of a song from the list is quite cumbersome. Using an effective dialogue system can help to resolve this problem. The dialogue system has to work in a quite systematic and structured way, where a user, who wants to listen to a song provides some information like songs name, artists name, album, year of release of song and genre in speech format. The audio can be converted into text using ASR and the entities can be extracted from the text. These entities can be used to search the database to play the song. However, the performance of the dialogue system would degrade if the output from ASR is not correct.

To address this problem, we experiment with the two popular approaches, HMMs based models [5] and DeepSpeech2 [6], to train the ASR system. Both the models were first trained on publicly available Voxforge dataset to learn generic acoustic features. Acoustic features are being used by various applications like ADVISOR [7] combined with other features like geographic and visual to recommend personalized video soundtrack recommendation. Audio and text features are also being used in cross-modal correlation learning as shown by Yi Yu et al. [8]. Once we train DeepSpeech2 to learn generic acoustic features, we then fine-tune the model on our data to adjust the acoustic and language model as per the domain.

The main contributions of this paper are in two-folds:

- Domain adaptation of pre-trained models on domain-specific data.
- Discussion on potential use cases of this kind of ASR in different domains.

The rest of the paper is organized as follows. In Section 2 we discuss previous relevant work. In Section 3 and Section 4 we present an overview of the approaches and experimental setup of both approaches, respectively. Section 5 briefs the data used to train the base model. We present results in Section 6 and conclude in Section 7. Use cases and future work are presented in Section 8 and Section 9, respectively.

2. RELATED WORK

This section reviews similar experiments and inspiration for some of the approaches undertaken during our experiments.

Daniel Povey, inventer of Kaldi toolkit presented results on Resource Management dataset for a basic triphone system in comparison with HTK toolkit in [9], due to which we chose Kaldi for our experiments with classical models. The work done by Kozierski et al. [10] was helped give us a rough starting point approximation for a relationship between the number of hours of data and number of Gaussians in GMMs along with number of decision tree leaves for Triphone modeling. A tutorial on creation of full pipeline along with experiments on Aspire and Iban datasets is given in the work [11].

Deep Learning has been successful in a variety of domains, including Speech and gives better results than classical models along with a scope for construction of hybrid models with the techniques mentioned in the previous paragraph [12]. Recurrent Neural Networks (RNNs) are a powerful type of Deep Neural Networks for sequential data, and with the introduction of Connectionist Temporal Classification (CTC) [13] make it possible to train RNNs for sequence labelling problems where the input-output alignment is unknown. The combination of these methods with the LSTM/GRU RNN architecture has proved particularly fruitful, delivering state-of-the-art results in cursive handwriting recognition [14]. Researchers around the world have tried applying it for ASR systems and the results (WER) are not just comparable to classical methods but are better in the case of noisy environments/input, [15, 16, 6] are some of them. The previous work on ASR using Deep learning prove that in the case of noisy environments deep learning architectures learn better representation of speech compared to their counterparts, i.e. classical methods.

3. APPROACHES

As described earlier, there are mainly two approaches for creating an ASR system. The classical approach mainly deals with HMMs-GMMs and related techniques while the Deep Learning approach focus on a more end to end pipeline.

3.1. Classical Models

An ASR model can be broken down into two distinct parts: Acoustic model used to learn the various acoustic properties of the training set and language model, to fine-tune the acoustics predicted and is used as a grammar for the model as shown in Equation 1.

$$P(W|O) = \hat{W} = \arg\max_{W} P(O|W)P(W) \tag{1}$$

In Equation 1, W refers to the most likely transcription, O refers to the observation (feature vectors), $P(O|W)$ (from Bayes rule) is contributed by the acoustic model and is handled by the GMM and HMMs, while $P(W)$ is contributed by the language model.

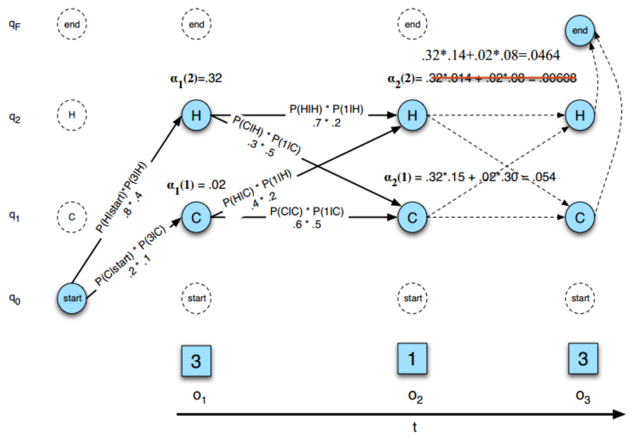


Fig. 1. Forward algorithm computation for two hidden states, H and C [18]

An HMM is an extension of Markov models to model temporal processes in which the set of states emit an observation. Usually the set of states are hidden and we get the observations. In order to train the parameters of these HMMs, there are three algorithms [17].

- Forward Algorithm: Given the set of states, the set of observations and the parameters (transition probabilities and emission probabilities), this algorithm gives us the likelihood of a sequence by doing a forward computation for all the possible hidden state combinations. Figure 1 shows calculation for forward trellis of two states H and C and corresponding numeric observations.

- Viterbi Algorithm: During the forward algorithm, we compute the likelihood by summing over all the possible hidden state combinations. In order to get the best possible hidden state sequence (the one which maximizes the likelihood of the given observation sequence), we use the Viterbi Algorithm. Figure 2 shows Viterbi trellis for two states H and C, choosing of best hypothesis at each step and corresponding numeric observations.

- Forward-Backward Algorithm: Given the set of hidden states and the set of observation sequence, the forward backward algorithm is a version of expectation maximization algorithm to estimate the parameters with respect to the training data.

In the context of speech, the feature vectors computed from the audio signal act as our observations. A set of phonemes act as our hidden states, which emit the feature vectors. A word can be broken down in phonemes, which are basic units of sound and act as a single state in a Bakis (left to right) HMM as shown in Figure 3. The feature vec-

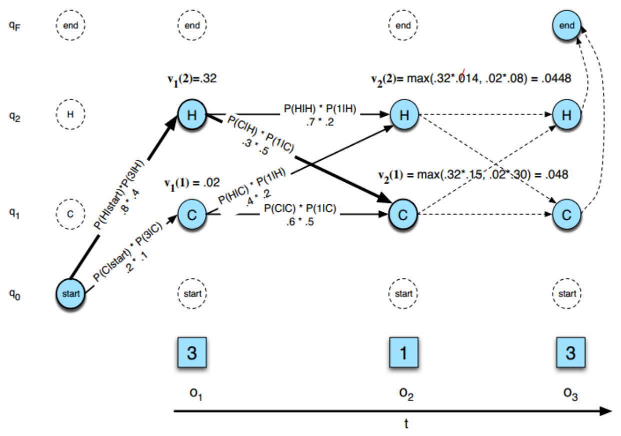


Fig. 2. Viterbi algorithm computation for two hidden states H and C [18]

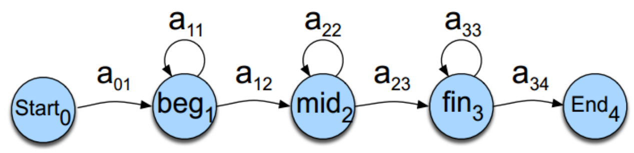


Fig. 3. A Bakis HMM of phones [18]

tors (MFCC or PLP) are assumed to be drawn from a mixture of D-variate Gaussians, where D is the dimensions of our feature vector at each time step [19]. Using the forward-backward algorithm, the HMM model is trained to maximize the likelihood for each audio signal.

Two types of features are mostly commonly extracted to be used with HMMs, MFCCs and PLPs. MFCC feature extraction is a process which extracts the features based on the frequency and pitch present in the audio signal. It uses something known as a Mel Scale, which bears close resemblance with the human auditory system [20]. PLP feature extraction is inspired by psychophysics of hearing and it also tries to model the working of the human auditory system [21].

3.1.1. Algorithms Used

At the most basic level, Monophone modeling is done. By monophone, we mean no context dependency on the preceding and succeeding phone. However it is a common observation that a particular phone is affected by both the preceding and succeeding phone. So we align the triphone model with the monophone one and move further. By alignment, we mean assigning particular MFCC vectors (observations) to particular phonemes (hidden states). This alignment acts as a starting point for the Expectation-Maximization algorithm to better estimate the HMM parameters. For monophone models, the alignment step is done using something known as a Flat start i.e. assigning uniform probabilities. A monophone model is generally trained on a subset of the dataset (usually 25-40%) as its main role is to set up a better starting point for the triphone models.

The monophone aligned triphone model generally gives the best results and hence higher order dependencies (quinphones) are not used. Given that there are N phones, the number of triphones states can be N^3. However not all combinations are linguistically valid (like three nasal sounds are not possible together). Hence, a lot of these triphone combinations are invalid. Further, to speed up the training process similar triphones (in terms of their linguistic structure) are clustered together and their parameters are trained together, a phenomenon known as tied-state training. This clustering is done using Decision trees, as we go down the tree we ask more linguistic questions to tie the triphones together.

In cases where speaker adaptation needs to be done, i.e. adapting our model to the acoustics of the test speakers, we can use the Maximum Likelihood Linear Transform (MLLT). This is done with LDA (Linear Discriminant Analysis). LDA reduces the feature space, while MLLT learns a transformation matrix for each speaker in the test set to reduce acoustic differences between speakers and normalize the final models with respect to new speakers. Techniques like fMLLR [22] also take phoneme variation into account. Since in our project we are building a speaker independent ASR system, this technique was not useful for us.

Until now, the type of training done was generative, i.e. trying to model the distribution from which our data is generated. We also tried a discriminative approach, where we not only tried to maximize the likelihood of correct transcriptions but also try to minimize the likelihood of incorrect ones obtained during the decoding process. To achieve this, we used MMI (Maximum Mutual Information) [23]. This approach trains ”denominator lattices”, which can be defined in simpler terms as all possible transcriptions for the given audio input. Now we try to maximize the log ratio of the given lattice and the denominator lattice, which ensures that we are ”discriminating” the most likely sequence. We can further make this objective function more strong by boosting the denominator (to add resistance to minimization of denominator) and also using metrics such as MPE (Minimum Phone Error) and MWE (Minimum Word Error) as explained in [23] and [24].

3.2. Deep Learning Models

The researchers have tried to use deep learning based methods in the past [25] but compared to classical methods the results were not good mainly because of two reasons : data and computational power (both of which are true in the present times). Since audio signals [26] are time dependent and the output at a particular time step depends on the previous and next time step, the initial experiments using feed forward gave poor results when compared to the state of the art classical methods. It is also one of the reasons classical methods were more popular around in the past, because of these two major reasons classical methods dominated the ASR research in the past.

RNNs [27] solved the problem of time dependent input but still when the experimentation started on sequence modeling architectures like RNNs, the results were not as good as those obtained by classical methods, since RNN based architectures cannot model dependency of longer time steps [28, 29] (a single input can be than 500 steps). Also, RNNs require a lot of computation power when compare to conventional deep Neural Network architectures. It hindered the popularity of deep learning methods in the research community.

With better computation power and data, researchers tried to use a modified version of RNN i.e. LSTMs [30, 31] or GRUs [32, 33, 34], but still researchers faced a big bottleneck. Though a lot of data was available, sufficient amount of segmented data was not available i.e. alignment of particular frames of input to the output. Moreover, it was not feasible to hand annotate the data as it is prone to a lot of errors. In 2006, the concept of the Connectionist Temporal Classification (CTC) [13] loss emerged,which did not require the exact alignment of input to the output and thus, we do not need segmented data anymore.

One important reason to choose deep neural networks over the classical ones is their simplicity as they provide an end to end solution to ASR when compared to classical methods. Language model [35] plays an important role in the ASR pipeline. Given the character probabilities from the acoustic model, the language model provides context to distinguish between words and phrases that sound similar. A Language model can be understood as statistical language model which is a probability distribution over sequences of words. Given such a sequence, say of length m, it assigns a probability $P(w^1, .., w^{\mathrm{m}})$ to the whole sequence. Language models (LM) can be classified into two categories: Count-based LMs [36] and Continuous-space LM [37, 38].

3.2.1. *Architectures used*

We tried DeepSpeech1 (DS1) [16] and DeepSpeech2 (DS2) [6] architectures, also one other with a little modification in DS2. The DS1 architecture, shown in Figure 4 is significantly simpler than traditional classical speech systems. The network has five layers: the input is fed into three fully connected layers, followed by a bidirectional RNN layer, and finally a fully connected layer. The hidden fully connected layers use the ReLU activation. The RNN layer uses LSTM cells with tanh activation. The input is the features extracted from the audio signals which are MFCC features in this case and output of the network is a matrix of character probabilities over time. In other words, for each time step, the network outputs one probability for each character in the alphabet, which represents the likelihood of that character corresponding to whats being said in the audio at that time.

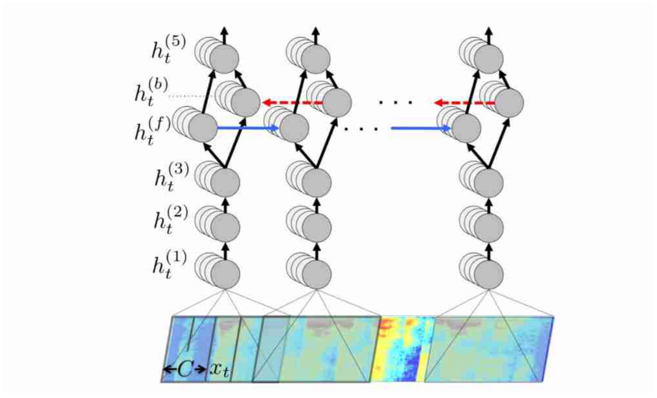
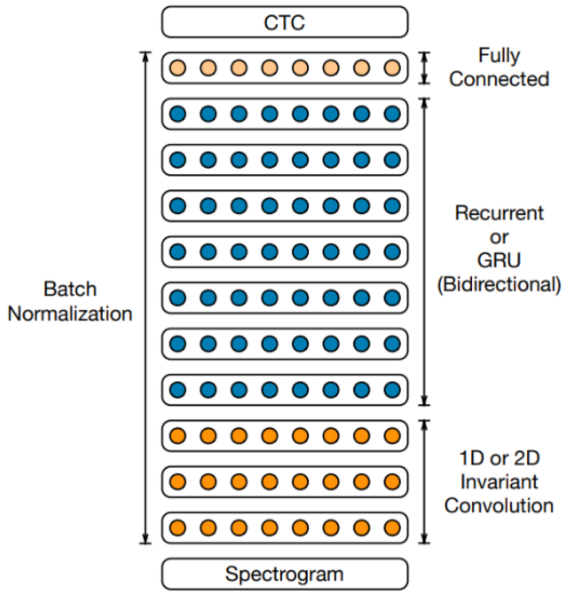


Fig. 4. deep speech 1 Architecture [16]



CTC
Fully Connected
Recurrent or GRU (Bidirectional)
Batch Normalization
1D or 2D Invariant Convolution
Spectrogram

Fig. 5. deep speech 2 Architecture [6]

The DS2 architecture, shown in Figure 5 has more layers (9 in total) than DS1, shown in Figure 4. Also, the input is a spectogram, not the MFCC features. The three fully connected layers from Deep Speech 1 are replaced by the Convolution layers followed by bidirectional RNN layers (1-7), and finally a fully connected layer. The hidden fully connected layers use the ReLU activation. The RNN layer uses GRU cells with tanh activation. Batch norm is applied to all the layers. There is an another variant of DS2 with look ahead convolution.

We also tried a 3rd architecture, shown in Figure 6, with a little modification to DS2, more convolution layers in parallel to the pre-defined convolution layers, to make it more robust to noise. It would require more careful benchmark-

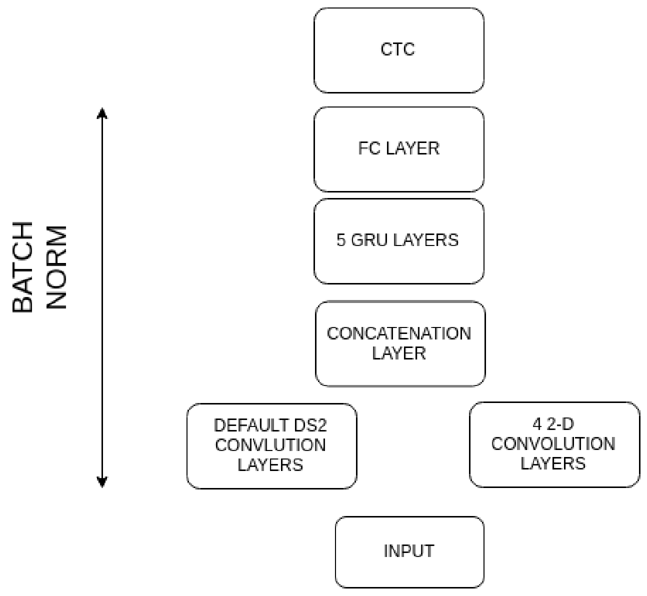


Fig. 6. Variant of Deep Speech 2

ing and rigorous testing before making any concrete claims that it is working better than DS2 [6] or to call to it DS3. Though, for our case it did work better in the noisy case. As the input/feature is an image(Spectogram) so let the convolution layers handle the removal of noise from the input instead of RNN layers . Additionally, we used smaller size filter to make more small decisions rather than one big decision with the bigger default filters in parallel as it is shown in Figure 6 because in a real world scenario the noise changes every second. The results prove our hypothesis as the WER decreases compared to the default DS2.

4. EXPERIMENTAL SETUP

4.1. Classical Setup

4.1.1. Toolkit Used

Experiments are performed using Kaldi, an open source Speech recognition toolkit developed by Daniel Povey, written in C++ and Shell scripting language. To run experiments in Kaldi, the data should be in a specific format like a speaker to utterance file, a file mapping absolute path of utterances to the name of the speaker etc. Utilities to prepare these format files have been provided in the toolkit for most of the standard open source datasets available. Recipes are provided to run standard algorithms like Monophone and Triphone models and some baseline discriminative algorithms. To prepare any ASR experiment for a custom dataset, the below steps should be followed in order:

- Audio data: Audio data with same sampling rate and normalized transcriptions(no pronunciation or special symbols and for numbers)

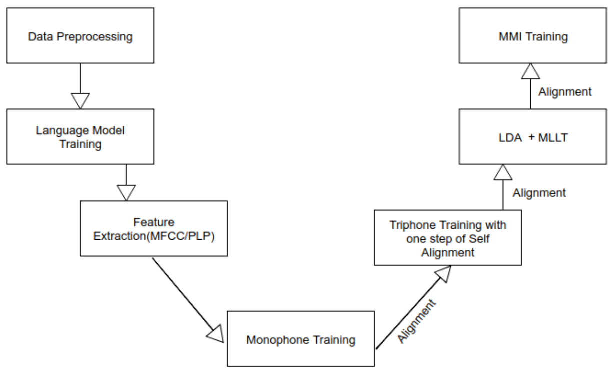


Fig. 7. The general order of execution of algorithms in a Kaldi rcipe for Classical Models

- Preprocessing Files: spk2gender (Speaker ID and gender of the speaker), wav.scp (Speaker utterance with its wav file), text (the utterance itself), utt2spk(The utterance of the speaker with the Speaker ID)

- Language Data: lexicon.txt (Every unique word in the transcripts along with their phonetic expansion, which can be acquired using the CMU Lexicon), nonsilence_phones.txt, silence_phones.txt, optional_silence_phones.txt.

- Tools: IRSTLM (or the language modeling toolkit of your choice), utils and steps from WSJ recipe in kaldi/egs.

- Config files: decode.config (for setting beam width and other parameters), mfcc.conf or plp.conf (parameters to extract MFCC or PLP features like sampling rate, Energy, etc.)

- Recipes: This can be directly picked from kaldi/egs/voxforge/s5/steps. Some common scripts used are train_mono.sh, train_deltas.sh and decode.sh.

The HMM-GMM recipes are CPU intensive and the operations can be parallelized to some extent by using a higher number of cores. This is tweaked by setting the number of jobs parameter in every shell script.

Figure 7 shows the basic flow of algorithms in a Kaldi Recipe (example considered for Voxforge Recipe). We can see from the figure that after preprocessing and an initial flat start model, we can use the alignments of models with each other. We can also use this alignments with Deep NeurNetworks, for which standard scripts have been provided in the toolkit.

4.2. Deep Learning Setup

We used Tensorflow [39] framework for DeepSpeech1 (DS1) and PyTorch [40] framework for DeepSpeech2 (DS2). In order to pre process the transcripts, the characters had to be matched as given to the last layer of the model with normalized transcripts.

The experiment consists of training the models to transcribe the 8KHz sampled audio files with/without augmentation as required by the project. In order to increase robustness of the model, noise was added in the audio samples. White noise was used for this task which included the sounds of train, fan, water, rain, ocean, bus, etc. We did not change anything in the DS1 architecture and used default parameters and values as defined in the paper [16].

For DS2, the variable hyper parameters included total number of hidden layers, the encoding size of the RNN layers, the learning rate, RNN unit (RNN/LSTM/GRU). At first, we focused on increasing the acoustic accuracy of the model i.e. without the language model, and later with the language model combined as we wanted to squeeze out that extra juice out of the acoustic model. It helped us to achieve a WER close to 10 with the language model combined.

We tried 3 variants of DS2. The first and second variants are the default architectures as defined in the DS2 paper [6]. The third one is a little bit different in a way that we added more convolution layers to increase the resistance towards noisy input.

The first architecture has 5 GRU, initial 0.001 Learning rate (lr) and encoding size of 512 with other parameters as default, as mentioned in DS2 paper [6]. One little trick that we used to train the network to overcome the problem of vanishing gradients and the fact that RNNs are harder to train, was to train it progressively from 1, 3 and 5 GRU units. With less number of parameters in the starting the gradient can flow backward easily and, also there are less chances of it to stuck at any local minima .

The second architecture that we tried had look-ahead convolution layers with Uni Directional GRU units compared to Bi directional units in the first architecture. When compared to first architecture the WER was worse off. Therefore we stopped working on this approach.

Finally, to make our pre-trained 1st architecture robust to noise we added additional convolution layers in parallel to the default convolution layers as can be seen in Figure 6.

Language model plays an important role in the ASR pipeline to achieve close to 10 WER as can be inferred from table 3 and 4. We trained/tried both N-Gram/Count and RNN/Continuous-space based LMs. It can be proved in the later sections that RNN/Continuous-space based LM performs better. After training Continuous-space LM, we generated text from that and then trained N gram LM from that text. Also, decoding plays an important role to achieve that WER and therefore we tried Greedy and beam search decoding and particularly the Prefix Beam Search decoding [41]. The results are compared in the later section.

Experiment	Parameters Changed	WER(%)
Triphones(DD)	Default	34
Boosted MMI	Default	23
Monophone(Full)	50 Iters Lm Order=3	41
Triphones	Mono Aligned Lattice beam=10	32
Monophones	Voxforge + CV Tot. Gauss. = 14000	33
Triphones	Voxforge + CV Num. Leaves=7000	27
Triphones	Voxforge + CV + Librispeech	24
MMI with MPE	+ Voxforge + CV + Librispeech	22

Table 1. Some of the results obtained for various HMM-GMM models using the Kaldi Toolkit

5. DATASET USED

The dataset that we used is the Voxforge dataset. Voxforge dataset is a collection of transcribed speech files via user uploads. The dataset is unbalanced and has a lot of male examples. Furthermore it has a variety of dialects of English language, from Indian English to Australian English. The dataset is available at two sample rates: 8k and 16k, we used the 8k version. The testing set for both the classical and Deep Models is a part of this Voxforge dataset and is exactly the same for both models. To make the dataset more diverse and generalize our models well, we have also expanded our dataset to nearly thrice the size by randomly sampling instances from Mozillas CommonVoice dataset and OpenSLRs LibriSpeech corpus. To further make our models robust to noise, we added 10 types of noise files: Call Centre ambiance, Road ambiance, White Noise to name some; randomly in 40% of the dataset.

6. RESULTS

6.1. Classical Models

The default code recipes present in Kaldi for Voxforge dataset were used to train a variety of models, from monophone models to triphone models with speaker adaptation. Table 1 presents the some of the most important results along with the setting of the hyper-parameters. The next experiment takes the in best hyper-parameters from the previous experiments.

We mainly optimized our monophone and triphone models as each advanced model is dependent on the previous

Architecture	Encoding	WER(%)
DS1 -Default	256	34
DS2 -3 Uni Dir. RNN(LA Conv)	512	49.040
DS2 -3 Bi Directional RNN	512	44.326
DS2 -5 Bi Dir. RNN	512	30.885

Table 2. Results obtained for Acoustic model only with greedy decoding and without LM. (LA, look ahead)

Architecture	Noise	WER(%)
DS2 - 5 Bi Dir. RNN	NA	30.885
DS2 - 5 Bi Dir. RNN	YES	44.081
DS2 - 5 Bi Dir. RNN (added conv)	YES	39.720

Table 3. Results obtained for with/without noisy input with greedy decoding and without LM.

model for alignments and these two models formed the base for further experiments. The WER comes down by increasing the number of training iterations and increasing the number of Gaussians to be estimated. A small validation set of 62 speakers kept some control on over fitting. With hyperparameter tuning we go down to around 30% WER for triphone models on the Voxforge dataset alone. Further using these models as alignments to MMI/MPE/MWE models push results down to 24%. To tackle overfitting, we decided to make our dataset more diverse by first creating a pool of three datasets; Voxforge, 15-30% of Commonvoice and 15-30% of Librispeech along with noise from various sources. With this increase in size, we also raised the number of gaussians and number of leaves for triphone decision trees. The performance on real speech samples improved but still were sub optimal in comparison to Deep Learning Models.

6.2. Deep learning Models

As can be seen from table 2, DS2 - 5 Bi Dir. RNN, works better compared to other approaches for the same audio inputs and has a better WER when compared to DS1. The input is real world recordings without any artificial noise injection. Also, it can be inferred from table 1. that Bi Directional RNN works better when compared to Uni Dir. RNN(look-ahead). All the observations are for Acoustic modelling and decoding is greedy without any LM.

It can be concluded from table 3 that for noisy environments (artificial noise is added to the input (white noise to be specific)) WER for raw/default DS 2 architecture is high when compared to the variant with additional Convolution layers parallel to the default convolution layers of the DS 2 Architecture. Therefore, it proves our assumptions as the input is an image therefore it helps to let the convolution layers handle the noise rather than RNN layers. Adittionaly, we further

Beam-size	Noise	alpha, Beta	WER(%)
10	NA	0.0, 0.0	24.477
10	NA	1.95, 4.0	21.464
100	NA	0.0, 0.0	24.043
100	NA	1.95, 4.0	13.160
500	NA	0.0, 0.0	23.989
500	YES	1.95, 4.0	19.791
500	NA	1.95, 4.0	11.114
500 (Neural LM)	NA	1.95, 4.0	10.850

Table 4. Results obtained for DS2 - 5 Bi Dir. RNN Acoustic model with language model(5 Gram) and prefix beam search.

Additionally, it can be inferred from table 4 that beam width and alpha & beta values plays an important role. As the beam width increases from 10 to 500 the WER decreases substantially. Also, the Neural LM works better when compared to traditional N gram LM.

We tried adapting the Model/Architecture to other domains/Accent and got comparable results with tuning it for just under 10 epochs. Specifically, we tried it on two different domains as of writing this paper which are Indian Accent English, Finance Domain (Indian accent). The WER for Acoustic model only are 40%, 14% and with the addition of LM (3-gram) the WER went down to 14.727%, 5.7% in the order. We are also in process of applying it to other domains which are work in progress.

7. CONCLUSION

Classical models like HMM-GMM models, Monophone and Triphone Training, MMI discriminative training are extremely intuitive in the way they make predictions and provide a faster inference time with explainable theory, but depend on the underlying assumptions about the data and require a lot of hand crafting. Future work to improve upon the brittleness of classical models can be done by leveraging the approximation power of Deep Neural Networks to estimate the probabilities in place of GMMs and using the alignments of the best HMM-GMM models as inputs to the various DNN-HMM models. Convolution Neural Networks and Long Short Term Memory networks are extremely successful at modelling temporal dependencies and hence using them in conjunction with Classical methods should result in better performance.

End to end deep learning methods presents the exciting opportunity to improve the ASR systems as the data and computation power increases. The results also prove that compared to classical approaches the WER is substantially less for Deep learning methods. Also, the noise is better handled by the convolution layers. Without a good LM we cannot achieve a WER close to 10% and it can be seen from table 4.

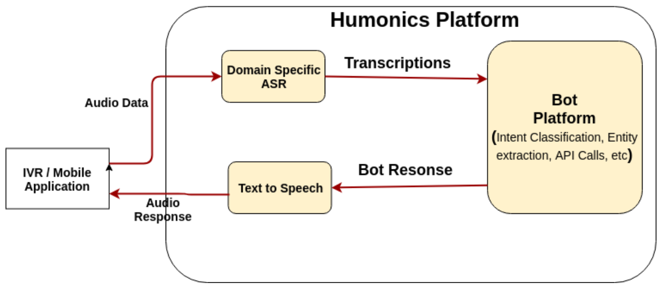


Fig. 8. Variant of Deep Speech 2

The beam width also plays an important role in deciding the WER but as the beam width increase and so is the inference time. To conclude, a beam width of 100 works fine for most of the practical cases. We believe that with more data and the advancement in Deep learning techniques we will see better ASR systems in the future. The results are comparable to

8. USE CASES

There are many use cases of Domain-specific ASR. We discuss two of the major industrial cases where these ASR systems can be most effective.

1. **Customer Service:** In Customer Service, there is a strong and growing interest in chatbots, driven by the promise of intelligent digital assistants always available to resolve customer requests cheaply, quickly and consistently. Chatbots are becoming more sophisticated and contextually-aware, making them able to anticipate and respond to individual customer needs. To take advantage of the inherent speed and simplicity of speech-to-text and text-to-speech based communications, chatbots must be capable of recognizing and translating voice-based commands. Industry Specific ASR engine enables the bots to understand the jargon of that particular industry. The more robust the ASR is the less burden is on the dialogue system to interpret the messages.

2. **Digital Professional Sribes:** A professional scribe is the one who helps to document the conversations/instructions of a professional person. For example, a medical scribe is the one who helps the physicians to ease off the administrative stuff and increase the productivity of the physician. A domain adaptive ASR system along with other techniques like speaker diarization can be used to create an effective professional scribe.

A high level overview of the whole pipeline for Mod can be seen in Figure 8.

9. FUTURE WORK

1. **Vernacular Adaptation:** Since it is shown that we can adapt the ASR to a different domain by fine-tuning the base model and change the language model to create domain-specific ASR system. In this system, the language of the adapted model is the same as that of the base model. In the next experiment, we will be training the models on different vernacular languages like Hindi, Telugu, etc.,

2. **Lightweight models:** To make any industrial use case successful, the system should be cost-effective. The current deep learning ASR model runs on GPUs to get real-time inference results which are costly. The next steps would be to make these models light enough so that they can be deployed on the edge devices. This can be done by using techniques like quantization and weight pruning to make the models lighter without compromising the accuracy of the model.

10. ACKNOWLEDGMENTS

We would like to thank Soumay Seth for bringing such great use cases. We would also like to thank Pallavi Rawat (M. Tech - Batch of 2019, IIIT Delhi), Mehak Piplani (MIDAS Lab), Ashima Survana (MIDAS Lab and DTU Delhi) and Harshit Malhotra for their valuable contributions and fruitful discussions during the course of this project. This project is supported by Humonics Global Pvt. Ltd. Dr. Rajiv Ratn Shah is partly supported by the Infosys Center for AI, IIIT Delhi and ECRA Grant by SERB, Government of India. [42]

11. REFERENCES

[1] Ghulam Muhammad, "Automatic speech recognition using interlaced derivative pattern for cloud based healthcare system," *Cluster Computing*, vol. 18, no. 2, pp. 795–802, 2015.

[2] Mike Wald, "Using automatic speech recognition to enhance education for all students: Turning a vision into reality," in *Proceedings Frontiers in Education 35th Annual Conference*. IEEE, 2005, pp. S3G–S3G.

[3] Christopher Cieri, David Miller, and Kevin Walker, "The fisher corpus: a resource for the next generations of speech-to-text.," .

[4] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur, "Librispeech: an asr corpus based on public domain audio books," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2015, pp. 5206–5210.

[5] M. Mohri, “Finite-state transducers in language and speech processing,” vol. 23. 1997.

[6] Dario Amodei, Sundaram Ananthanarayanan, Rishita Anubhai, and Jingliang et. al. Bai, “Deep speech 2: End-to-end speech recognition in english and mandarin,” 2016, pp. 173–182.

[7] Rajiv Ratn Shah, Yi Yu, and Roger Zimmermann, “Advisor: Personalized video soundtrack recommendation by late fusion with heuristic rankings,” in *ACM Multimedia*, 2014.

[8] Yi Yu, Suhua Tang, Francisco Raposo, and Lei Chen, “Deep cross-modal correlation learning for audio and lyrics in music retrieval,” *ArXiv*, vol. abs/1711.08976, 2017.

[9] G. Boulianne et. al. D. Povey, A. Ghoshal, “The kaldi speech recognition toolkit,” *Proc. of IEEE Workshop on Automatic Speech Recognition and Understanding*, 2011.

[10] Piotr Kozierski, Talar Sadalla, Szymon Drgas, Adam Dabrowski, Joanna Zietkiewicz, and Wojciech Giernacki, “Acoustic model training, using kaldi, for automatic whispery speech recognition,” 09 2018, pp. 109–114.

[11] S. Khudanpur D. Povey, J. Trmal, “Building speech recognition systems with the kaldi toolkit,” *Presentation at Center for Language and Speech Processing, John Hopkins University*, 2016.

[12] Geoffrey Hinton, Li Deng, Dong Yu, George Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Brian Kingsbury, et al., “Deep neural networks for acoustic modeling in speech recognition,” *IEEE Signal processing magazine*, vol. 29, 2012.

[13] Faustino Gomez Alex Graves, Santiago Fern andez and J urgen Schmidhuber, “Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks.,” *In Proceedings of the 23rd International Conference on Machine Learning, ICML 06, pages 369376, New York, NY, USA, 2006. ACM.*

[14] Marcus Liwicki, Alex Graves, Santiago Fernàndez, Horst Bunke, and Jürgen Schmidhuber, “A novel approach to on-line handwriting recognition based on bidirectional long short-term memory networks,” in *Proceedings of the 9th International Conference on Document Analysis and Recognition, ICDAR 2007*, 2007.

[15] Alex Graves and Navdeep Jaitly, “Towards end-to-end speech recognition with recurrent neural networks,” in *International conference on machine learning*, 2014, pp. 1764–1772.

[16] Jared Casper Bryan Catanzaro Greg Diamos Erich Elsen Ryan Prenger Sanjeev Satheesh Shubho Sengupta Adam Coates Awni Hannun, Carl Case and Andrew Y. Ng., “Deepspeech: Scaling up end-to-end speech recognition.,” *Baidu Research Silicon Valley AI Lab, 12 2014.*

[17] L. R. Rabiner, “A tutorial on hidden markov models and selected applications in speech recognition,” *Proceedings of the IEEE*, vol. 77(2), pp. 257–286, Feb. 1989.

[18] Andrew Mass, “Cs224s, spoken language processing, stanford university,” *npj Digital Medicine*, vol. Lecture 3, 2017.

[19] Martin Jurafsky, D., *Speech and Language Processing*, Prentice Hall, 2000.

[20] Jianfei Cai Liang-Tien Chia Changsheng Xu Qi Tian Min Xu, Ling-Yu Duan, “Hmm-based audio keyword generation,” *Advances in Multimedia Information Processing PCM 2004: 5th Pacific Rim Conference on Multimedia. Springer*, 2004.

[21] H. Hermansky, “Perceptual linear predictive (plp) analysis of speech,” vol. 87, pp. 1738–1752. 1990.

[22] M.J.F. Gales, “Maximum likelihood linear transformations for hmm-based speech recognition,” *Computer Speech Language*, vol. 12, no. 2, pp. 75–98, 1998.

[23] P. de Souza R. Mercer L. Bahl, P. Brown, “Maximum mutual information estimation of hidden markov model parameters for speech recognition,” *ICASSP ’86. IEEE International Conference on Acoustics, Speech, and Signal Processing*, April 1986.

[24] Keith Vertanen, “An overview of discriminative training for speech recognition,” *University of Cambridge*, pp. 1–14, 2004.

[25] Roger K Moore, “Twenty things we still dont know about speech,” in *Proc. CRIM/FORWISS Workshop on Progress and Prospects of speech Research an Technology*, 1994.

[26] Jérôme Sueur, “A very short introduction to sound analysis for those who like elephant trumpet calls or other wildlife sound,” 2019.

[27] David E Rumelhart, Geoffrey E Hinton, Ronald J Williams, et al., “Learning representations by back-propagating errors,” *Cognitive modeling*, vol. 5, no. 3, pp. 1, 1988.

[28] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur, “Recurrent neural network based language model,” in *Eleventh annual conference of the international speech communication association*, 2010.

[29] Haşim Sak, Andrew Senior, and Françoise Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," in *Fifteenth annual conference of the international speech communication association*, 2014.

[30] Sepp Hochreiter and Jürgen Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[31] Santiago Fernández, Alex Graves, and Jürgen Schmidhuber, "An application of recurrent neural networks to discriminative keyword spotting," in *International Conference on Artificial Neural Networks*. Springer, 2007, pp. 220–229.

[32] Joel C Heck and Fathi M Salem, "Simplified minimal gated unit variations for recurrent neural networks," in *2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS)*. IEEE, 2017, pp. 1593–1596.

[33] Rahul Dey and Fathi M Salemt, "Gate-variants of gated recurrent unit (gru) neural networks," in *2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS)*. IEEE, 2017, pp. 1597–1600.

[34] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[35] Y. Tam, Y. Lei, J. Zheng, and W. Wang, "Asr error detection using recurrent neural network language model and complementary asr," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2014, pp. 2312–2316.

[36] Andreas Stolcke, "Srilm-an extensible language modeling toolkit," in *Seventh international conference on spoken language processing*, 2002.

[37] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur, "Recurrent neural network based language model," in *Eleventh annual conference of the international speech communication association*, 2010.

[38] Tomas Mikolov and Geoffrey Zweig, "Context dependent recurrent neural network language model," in *2012 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2012, pp. 234–239.

[39] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, and Zhifeng Chen et. al., "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, Software available from tensorflow.org.

[40] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer, "Automatic differentiation in PyTorch," in *NIPS Autodiff Workshop*, 2017.

[41] Andrew L. Maas, Awni Y. Hannun, Daniel Jurafsky, and Andrew Y. Ng, "First-pass large vocabulary continuous speech recognition using bi-directional recurrent dnns," *CoRR*, vol. abs/1408.2873, 2014.

[42] Wes McKinney, "Data structures for statistical computing in python," in *Proceedings of the 9th Python in Science Conference*, Stéfan van der Walt and Jarrod Millman, Eds., 2010, pp. 51 – 56.