End to End Automatic Speech Recognition using Deep RNNs

Niranjan Ramesh Rao, Pruthvi Raju D R, Neeraj and Chiru Charan T S

Abstract— **In this paper we explore Automatic Speech Recognition using end-to-end Deep learning methods like RNN Transducers, Deep Speech 2 and Listen Attend and Spell (LAS). One of the key challenges in sequence transduction is learning to represent both the input and output sequences in a way that is invariant to sequential distortions such as shrinking, stretching and translating. Recurrent neural networks (RNNs) are a powerful sequence learning architecture that has proven capable of learning such representations. We have explored an end-to-end, probabilistic sequence transduction system, based entirely on RNNs, that is in principle able to transform any input sequence into any finite, discrete output sequence. Deep Speech 2 an end-to-end deep learning approach is explored which is used to recognize either English or Mandarin Chinese speech—two vastly different languages. We have also explored Listen, Attend and Spell (LAS), a neural network that learns to transcribe speech utterances to characters. Unlike traditional DNN-HMM models, this model learns all the components of a speech recognizer jointly. These end-to-end deep learning architectures are trained and evaluated on LibriSpeech Dataset.**

I. INTRODUCTION

Sequence transducers learn to represent sequential information in a way that is invariant, or at least robust, to sequential distortions. Transforming audio signals into sequences of words requires the ability to identify speech sounds (such as phonemes or syllables) despite the apparent distortions created by different voices, variable speaking rates, background noise etc. Recurrent neural networks (RNNs) are a promising architecture for general-purpose sequence transduction. The combination of a high-dimensional multivariate internal state and nonlinear state-to-state dynamics offers more expressive power than conventional sequential algorithms such as hidden Markov models.

Deep Speech 2 is an end-to-end deep learning model which replaces the many modules used in state of the art ASR pipelines. As speech recognition consists of wide range of variability in speech and acoustics, modern ASR pipelines are made up of numerous components including complex feature extraction, acoustic models, language and pronunciation models, speaker adaptation, etc. Building and tuning these individual components makes developing a new speech recognizer very hard, especially for a new language. Since Deep Speech 2 (DS2) is an end-to-end deep learning system, we can achieve performance gains by focusing on three crucial components: the model architecture, large labeled training datasets, and computational scale. This approach has also yielded great advances in other application areas such as computer vision and natural language.

Listen, Attend and Spell (LAS), is a neural network that learns to transcribe an audio sequence signal to a word sequence, one character at a time. Unlike previous approaches, LAS does not make independence assumptions in the label sequence and it does not rely on HMMs. LAS is based on the sequence to sequence learning framework with attention.

The next section discusses the previous work that has been done in the domain. Section 3 gives a description of the LibriSpeech dataset that we have used. Section 4 describes the algorithms that we have implemented. This followed by the section that describes the preprocessing and augmentation of data which is followed by Section 6 with experimental details. This is followed by sections results and conclusions.

II. LITERATURE SURVEY

This project was inspired by a lot of previous work that had been done in the domain of Speech processing and Deep Learning. Recurrent neural networks (RNNs) have been useful tools for sequence transduction. In the beginning days of RNNs there were many difficulties in the models [1], recently they are delivering state-of-the-art results in tasks such as handwriting recognition [2], generating text [3] and modelling language [4].

Recently Deep Neural Networks have become a commonality in Audio Speech Recognition with most of the state of the art speech work having deep neural networks [5]. Convolutional networks have been useful for acoustic models [6]. Speaking of RNNs, LSTMs have been used in state-of-the art recognizers [7] and they are useful with convolutional layers for the feature extraction [8]. Models with both bidirectional RNNs and unidirectional RNNs have been used. End-to-end speech recognition is a very agile area of research, with impressive results when used to score a DNN-HMM [9] and standalone. These two methods were used to convert audio sequences with varying lengths to transcriptions with varying lengths.

The RNN encoderd-ecoder design employs an encoder RNN to convert the input to a vector that has a constant length and a decoder network converts this vector into a sequence output [10]. To the decoder the attention mechanism was added and this improves performance of the system very much, this is very useful with long inputs and outputs [11]. In speech recognition, the RNN encoder decoder sequence to sequence models with attention mechanism do very well in modelling and predicting phonemes [12] and graphemes. The other commonly used techniques for converting audio input that is variable length to output that is variable in length is the CTC loss function [13] with an RNN to imitate time information. The CTCRNN model performed well in end to end speech recognition [14]. The CTC-RNN model has also performed superbly in predicting phonemes [15].

Data has been essential for end-to-end speech recognition labeled speech was used in Deep Speech 1 (DS1) [16]. Data augmentation was shown to be very effective and it improved the performance of speech recognition systems [17]. In an approach, a speech engine to align and filter hours of read speech was used [18]. In a different approach, an offline speech recognizer generated transcriptions for numerous hours of speech [19].

III. Dataset

For this project we use LibriSpeech ASR dataset. This dataset is available open source. This is a corpus of thousands of hours of 16 kilohertz English speech. This dataset was prepared by Vassil Panayotov with help from Daniel Povey. The data is taken from audiobooks read in the LibriVox project. The data taken from there has been segmented and aligned carefully. The dataset can be downloaded from https://www.openslr.org/12.

Each dataset item is an audio clip accompanied by the corresponding text. Each audio clip has a chapter_id and id also.

IV. Implemented Algorithms

We implement 2 well known architectures used in end to end Automatic Speech Recognition and evaluate their performance, additionally we propose additional modules implemented over the baseline architecture which shows competitive results. The following models are implemented

1) Deep Speech 2 (2016)
2) Listen Attend and Spell

A. DeepSpeech 2

Dario et al. proposed an end-to-end deep learning approach for automatic speech recognition. Fig 1. shows the architecture of DeepSpeech2. The model follows a recurrent neural network (RNN) with one or more convolutional input layers, followed by multiple recurrent (uni or bidirectional) layers and one fully connected layer before a softmax layer. The network is trained end-to-end using the CTC loss function (Graves et al., 2006), which allows us to directly predict the sequences of characters from input audio. The outputs are the alphabet of each language. At each output time-step t, the RNN makes a prediction, $p(l_t|x)$, where l_t is either a character in the alphabet or the blank symbol. In English we have $l_t \in \{a, b, c, ..., z, space, apostrophe, blank\}$, where we have added the space symbol to denote word boundaries. At inference time, CTC models are paired a with language model trained on a bigger corpus of text. We use a specialized beam search (Hannun et al., 2014b) to find the transcription y that maximizes:

$$Q(y) = log(p_{RNN}(y|x)) + \alpha log(p_{LM}(y)) + \beta wc(y)$$

where wc(y) is the number of words in the transcription y. The weight α controls the relative contributions of the language model and the CTC network. The weight β encourages more words in the transcription.

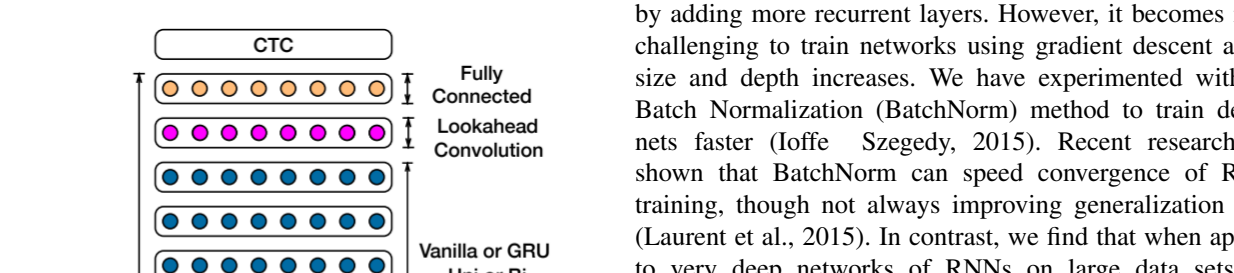

Fig. 1. Deep Speech 2 Architecture

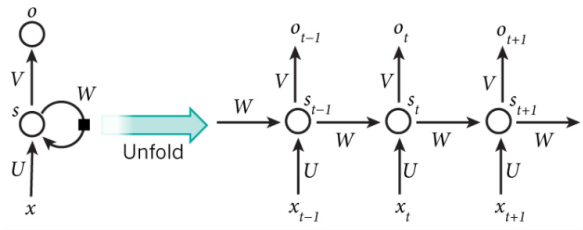

Fig. 2. RNN Architecture is one of the primary components used for sequence to sequence tasks

1) Batch Normalization: To efficiently absorb data as we scale the training set, we increase the depth of the networks by adding more recurrent layers. However, it becomes more challenging to train networks using gradient descent as the size and depth increases. We have experimented with the Batch Normalization (BatchNorm) method to train deeper nets faster (Ioffe Szegedy, 2015). Recent research has shown that BatchNorm can speed convergence of RNNs training, though not always improving generalization error (Laurent et al., 2015). In contrast, we find that when applied to very deep networks of RNNs on large data sets, the variant of BatchNorm we use substantially improves final generalization error in addition to accelerating training. A recurrent layer is implemented as :

$$h_t^l = f(W^l h_t^{l-1} + U^l h_{t-1}^l + b)$$

where the activations of layer l at time step t are computed by combining the activations from the previous layer h_t^{l1} at the same time step t and the activations from the current layer at the previous time step h_{t-1}^l.

2) Comparison of vanilla RNNs and GRUs: The models we have shown so far are vanilla RNNs which are modeled by Equation 3 with ReLU activations. More sophisticated

hidden units such as the Long Short-Term Memory (LSTM) units (Hochreiter Schmidhuber, 1997) and the Gated Recurrent Units (GRU) (Cho et al., 2014), have been shown to be very effective on similar tasks (Bahdanau et al., 2015). We examine GRUs because experiments on smaller data sets show the GRU and LSTM reach similar accuracy for the same number of parameters, but the GRUs are faster to train and less likely to diverge. Both GRU and vanilla RNN architectures benefit from BatchNorm and show strong results with deep networks.

3) Sorta Grad: Even with Batch Normalization, we find training with CTC to be occasionally unstable, particularly in the early stages. In order to make training more stable, we experiment with a training curriculum (Bengio et al., 2009; Zaremba Sutskever, 2014), which accelerates training and results in better generalization as well.

Training very deep networks (or RNNs with many steps) from scratch can fail early in training since outputs and gradients must be propagated through many poorly tuned layers of weights. In addition to exploding gradients (Pascanu et al., 2012), CTC often ends up assigning near-zero probability to very long transcriptions making gradient descent quite volatile. T SortaGrad improves the stability of training, and this effect is particularly pronounced in networks without BatchNorm, since these are even less numerically stable.

Proposed Modifications

Our model will be similar to the Deep Speech 2 architecture. The model will have two main neural network modules - N layers of Residual Convolutional Neural Networks (ResCNN) to learn the relevant audio features, and a set of Bidirectional Recurrent Neural Networks (BiRNN) to leverage the learned ResCNN audio features. The model is topped off with a fully connected layer used to classify characters per time step. Convolutional Neural Networks (CNN) are great at extracting abstract features, and we'll apply the same feature extraction power to audio spectrograms. Instead of just vanilla CNN layers, we choose to use Residual CNN layers. Residual connections enables us to build deep networks with good accuracy gains. Adding these Residual connections also helps the model learn faster and generalize better.

Recurrent Neural Networks (RNN) are naturally great at sequence modeling problems. RNN's processes the audio features step by step, making a prediction for each frame while using context from previous frames. We use BiRNN's because we want the context of not only the frame before each step, but the frames after it as well. This can help the model make better predictions, as each frame in the audio will have more information before making a prediction. We use Gated Recurrent Unit (GRU's) variant of RNN's as it needs less computational resources than LSTM's.

B. Vanilla Listen, Attend and Spell

Listen, Attend and Spell (LAS) is an encoder-decoder system. It composes of an encoder (the listener at the bottom) and a decoder (the speller on the top). The encoder takes audio frames and encodes them into a denser representation. The decoder decodes it into the distributions of characters in each decoding step. Before going into the details of the architecture we will need to know the inner workings of the submodules.

a) Pyramid BiLSTMs: LSTMs are designed in a manner to have more persistent memory thereby making it easier for RNNs to capture long-term dependencies. LSTMs consist of many logical gates that control the flow of memory, let us first take a look at the mathematical formulation of LSTM units before diving into the intuition behind this design:

$$i_t = \sigma(W^{(i)}x_t + U^{(i)}h_{t-1}$$
$$f_t = \sigma(W^{(f)}x_t + U^{(i)}h_{t-1}$$
$$o_t = \sigma(W^{(o)}x_t + U^{(o)}h_{t-1}$$
$$c_t = tanh(W^{(c)}x_t + U^{(c)}h_{t-1}$$
$$c_t = f_t \circ c_{t-1} + i_t \circ c_t$$
$$h_t = o_t \circ tan(c_t)$$

Biderctional LSTMs is a redesign over long short-term memory. In bidirectional long short-term memory, each training succession is introduced backwards and forward to isolate repetitive nets.

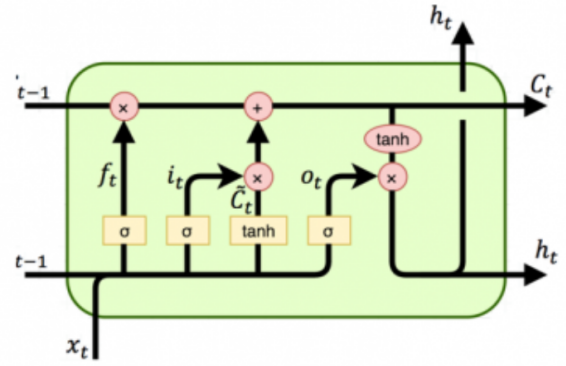

Fig. 3. LSTM Architecture

For speech data, the length of the input feature x can be hundreds to thousands of frames long, and applying a BLSTM on such a long sequence converges slowly and yields poor results. This can be circumvented by using pyramid BLSTM (pBLSTM). This modification is required to reduce the length U of h, from T, the length of the input x, because the input speech signals can be hundreds to thousands of frames long.

In each successive stacked pBLSTM layer, we reduce the time resolution by a factor of 2. In a typical deep BiLSTM architecture, the output at the ith time step, from the jth layer is computed as follows:

$$h_i^j = BILSTM(h_{i-1}^j, h_i^{j-1})$$

In the pBILSTM model, we concatenate the outputs at consecutive steps of each layer before feeding it to the next

layer, i.e.:

$$h_i^j = pBILSTM(h_{i-1}^j, [h_{2i}^{j-1}], [h_{j-1}^{2i+1}])$$

In our model, we stack 3 pBILSTMs on top of the bottom BILSTM layer to reduce the time resolution $2^3 = 8$ times. This allows the attention model (see next section) to extract the relevant information from a smaller number of times steps. In addition to reducing the resolution, the deep architecture allows the model to learn nonlinear feature representations of the data. See Figure for a visualization of the pBILSTM.

b) Attention: The model encodes the input x into a sequence of feature vectors, then computes the probability of the next output y_u as a function of the encoded input and previous outputs. The attention mechanism allows the decoder to look at different parts of the input sequence when predicting each output.

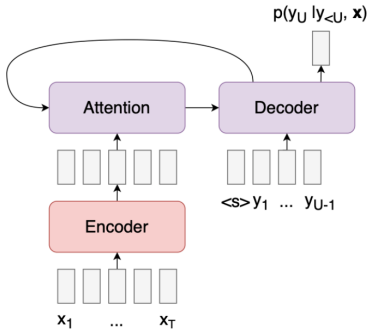

Fig. 4. Encoder Decoder with Attention

Attention models can be applied to any problem, but they are not always the best choice for certain problems, like speech recognition, for a few reasons

- The attention operation is expensive for long input sequences. The complexity of attending to the entire input for every output is O(TU) and for audio T and U are big.
- Attention models also don't take advantage of the fact that, for speech recognition, the alignment between inputs and outputs is monotonic: that is, if word A comes after word B in the transcript, word A must come after word B in the audio signal

c) Transducer: Vanilla CTC models assume that there is a monotonic input-output alignment. This ends up making the model a lot simpler. But CTC models have a couple problems of their own:

- The output sequence length U has to be smaller than the input sequence length, this prevents us from using a model architecture that does a lot of pooling, which can make the model a lot faster.
- The outputs are assumed to be independent of each other.

To find the optimal word sequence, we can incorporate a language model in the beam-search. A language model predicts what is next based on the previous decoded words. We can expand the concept further to create a deep network that makes predictions in the context of what it has already predicted.

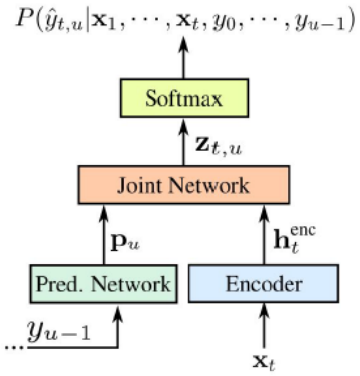

Fig. 5. Sequence Transducer model

The change requires two more components: the prediction network and the joint network. The predictor is autoregressive: it takes as input the previous outputs and produces feature vector g_u that can be used for predicting the next output, like a standard language model. g_u depends on the last prediction which also depends on the one before. Therefore, g_u actually depends on the whole sequence of previous predictions. g_u can be viewed as a guess on the coming prediction based on the previous predictions. On the other side, we have an encoder that creates a dense representation of the current input. We joint both information together using the equations above.

$$h_{t,u}^{joint} = tanh(Ah_t^{enc} + Bp_u + b)$$
$$z_{t,u} = Dh_{t,u}^{joint} + d$$

The joiner is a simple feedforward network that combines the encoder vector f_t and predictor vector g_u and ouputs a softmax $h_{t,u}$ over all the labels Intuitively, we form a new deep network that makes predictions based on previous contexts and the current observations. This is the concept of RNN transducer which integrates a language model concept into the deep network.

LAS Architecture

Now that we are familiar with the working of the inner modules, we will formally describe LAS which accepts acoustic features as inputs and emits English characters as outputs. Let $\mathbf{x} = (\mathbf{x_1}, ..., \mathbf{x_T})$ be our input sequence of filter bank spectra features, and let $\mathbf{y} = (\langle sos \rangle, y_1, ..., y_S, \langle eos \rangle), y_i \in \{a, b, c, ..., z, 0, ..., 9, \langle space \rangle, \langle comma \rangle, \langle period \rangle, \langle apostrophe \rangle, \langle unk \rangle\}$, be the output characters.

We want to model each character output yi as a conditional distribution over the previous characters $y_{<i}$ and the input signal x using the chain rule:

$$\prod_i P(y_i|x, y_{<i})$$

The basic LAS model consists of two sub-modules: the listener and the speller.

1) Listen: The listener is an acoustic model encoder, which takes the input speech features x , and transforms it into high level representation $h = (h_1, ..., h_U))$ with UT (T is the original frame number). The Listen operation uses a Bidirectional Long Short Term Memory RNN (BLSTM) with a pyramid structure. This allows us to form a dense representation for hundreds or thousands of audio frames. It allows helps us to explore the context gradually and hierarchically.

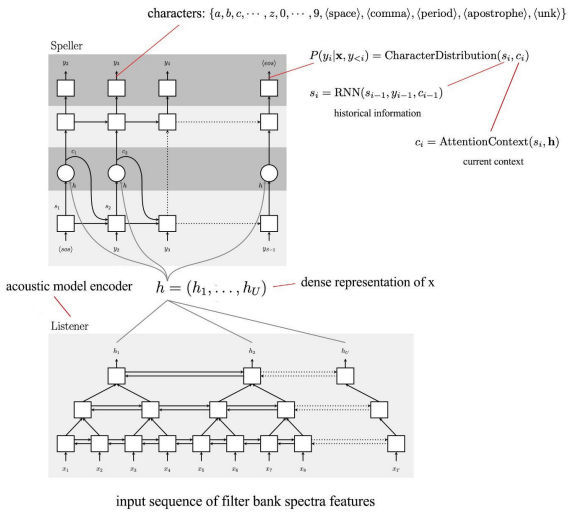

Fig. 6. Listen, Attend and Spell (LAS) model: the listener is a pyramidal BLSTM encoding our input sequence x into high level features h, the speller is an attention-based decoder generating the y characters from h.

2) Attend and Spell: The function is computed using an attention-based LSTM transducer [16, 15]. At every output step, the transducer produces a probability distribution over the next character conditioned on all the characters seen previously.

The speller is an attention-based decoder, which consumes h and produces a probability distribution of the target sequence $p(y|x)$. The attention mechanism in speller determines, at each time step i, which encoder output features in h should be attended to, and then generates a context vector c_i. The speller then takes the context vector c_i along with the previous prediction y_i-1 and generates a probability distribution $p(y_i|x, y < i)$. Specifically

$$c_i = AttentionContext(s_i, h)$$
$$s_i = RNN(s_{i-1}, y_{i-1}, c_{i-1})$$
$$P(y_i|x, y_{<i}) = CharachterDistribution(s_i, c_i)$$

where CharacterDistribution is a fully connected layer with softmax outputs over characters, and RNN is a two layer LSTM.

At each time step, i, the attention mechanism generates a context vector, c_i capturing the information in the input signal required to generate the next output. The attention model is content based - the contents of the decoder state s_i are matched to the contents of hu representing time step u of h, to generate an attention vector α_i. α_i is used to linearly blend vectors h_u to create c_i. Specifically, at each decoder timestep i, the AttentionContext function computes the scalar energy $e_{i,u}$ for each time step u, using vector $h_u \in h$ and s_i. The scalar energy $e_{i,u}$ is converted into a probability distribution over times steps (or attention) α_i using a softmax function. This is used to create the context vector c_i by linearly blending the listener features, h_u, at different time steps:

$$e_{i,u} = \langle \phi(s_i), \psi(h_u) \rangle$$
$$\alpha_{i,u} = \frac{exp(e_{i,u})}{\sum_u exp(e_{i,u})}$$
$$c_i = \sum_u \alpha_{i,u} h_u$$

where ϕ and ψ are fully connected networks. On convergence, the α_i distribution is typically very sharp, and focused on only a few frames of h; c_i can be seen as a continuous bag of weighted features of h. Figure 6 shows LAS architecture.

V. Preprocessing and Augmentation

128-dimensional log-mel filter bank features were computed for each input audio signal and used as the acoustic inputs to the model.

Filter banks are computed by passing the signal through a pre-emphasis filter; the signal then gets sliced into (overlapping) frames and a window function is applied to each frame; afterwards, we do a Fourier transform on each frame (or more specifically a Short-Time Fourier Transform) and calculate the power spectrum; and subsequently compute the filter banks. The final step is mean normalization.

We perform augmentation directly to the filter bank coefficients to artificially increase the diversity of our dataset in order to increase the dataset size. Park et al. proposed a simple and effective startegy wherein just by simply cutting out random blocks of consecutive time and frequency dimensions improved the models generalization abilities significantly.

VI. Experimental Details

We conduct experiments on the LibriSpeech Dataset. LibriSpeech training data consists of about 1000 hours of read audio books [21]. The dev and test sets are split into clean and other subsets. In our experiments, we only used clean subsets for both Dev and Test. Our system is tuned based on the WER on the Dev set, and the final optimized system is evaluated on Test.

We use 40-dimention static filter-bank features. The encoder network architecture consists of 3 pyramid BLSTM layers with 1024 hidden units in each direction. The decoder is a 2-layer LSTM with 512 hidden units per layer. Networks are trained with the cross-entropy criterion first, and then the model is used as the seed for discriminative training. We use 4-gpu synchronized training. TensorFlow [20] is used for all the experiments. Table 2 shows the contributions of different

TABLE 1

COMPARISON OF WER FOR DIFFERENT CONFIGURATIONS OF CONVOLUTIONAL LAYERS).

Architecture	Channels	Filter dimension	WER
1-layer 1D	**1280**	**11**	**14.6**
2-layer 1D	**640, 640**	**5, 5**	**12.9**
3-layer 1D	**512, 512, 512**	**5, 5,5**	**12.6**
1-layer 2D	**32**	**41x11**	**12.2**
2-layer 2D	**32,32**	**41x11, 21x11**	**11.4**
3-layer 2D	**32,32,96**	**41x11, 21x11, 21x11**	**10.8**

components in the LAS model. No external LM is used in these experiments.

For our model, we'll be using AdamW with the One Cycle Learning Rate Scheduler. The One Cycle Learning Rate Scheduler was first introduced in the paper Super-Convergence: Very Fast Training of Neural Networks Using Large Learning Rates. This helps in training neural networks an order of magnitude faster, while keeping their generalizable abilities, using a simple trick. We start with a low learning rate, which warms up to a large maximum learning rate, then decays linearly to the same point of where it originally started. Additionally we gain regularization benefits as the maximum learning rate is magnitudes higher than the lowest, which helps your model generalize better given we have a small dataset.

Loss Function Our model will be trained to predict the probability distribution of all characters in the alphabet for each frame (ie, timestep) in the spectrogram we feed into the model.

The innovation of the CTC loss function is that it allows us to skip this step. Our model will learn to align the transcript itself during training. The key to this is the "blank" label introduced by CTC, which gives the model the ability to say that a certain audio frame did not produce a character.

Architecture	***Baseline***	***BatchNorm***	***GRU***
2 CNN + 3 RNN	16.5	18.20	13.53
2 CNN + 5 RNN	15.1	14.4	13.02
2 CNN + 7 RNN	13.42	12.8	10.9
3 ResCNN + 5 RNN	11.83	10.2	**9.7**

TABLE 2

COMPARISON OF WER APPLICATION OF BATCHNORM AND SORTAGRAD, AND TYPE OF RECURRENT HIDDEN UNIT.

VII. RESULTS

We validate our models on LibriSpeech development set to tune the models based on the Word Error Rate. The optimized model is evaluated on the clean Test Set.

A. Deep Speech 2

We compare our modified Deep Speech model with the original model and its variants. Various setting of number of Recurrent and Convolution Layers are tested, along with modifications in architecture we evaluate different training strategies such as Batch Normalization, Sorta Grad as mentioned in the previous sections.

From Table 1 and Table 2, we can infer that a 3 layered input Residual Convolution for feature extraction is ideal. Over the convolutional layers we stack 5 Recurrent GRUs. From Table 1. we see the effect of Batch Normalization and Sorta Grad, which improves model stability and makes it generalizable.

B. Listen, Attend and Spell

We experiment with different deep recurrent modules on both the encoder and decoder side(Table 3). The baseline Listen, Attend and Spell uses 3 Pyramid BiLSTM with 1024 hidden units as the encoder. And on the decoder side an Attention based RNN Transducer, where the transducer has a 2 Layer RNN with 512 hidden units as the encoder for the language model. From Table 2 we can come to the conclusion that replacing the transducer encoder with LSTM units and local attention performs better. Additional processing by Spectogram Augmentation further improves the model.

Architecture	***Dev***	***Test***
Vanilla LAS	9.8	10.0
LSTM Transducer	8.1	8.8
LSTM+LA	7.40	7.49
LSTM + LA+ SpecAug	**6.58**	**6.67**

TABLE 3

COMPARISON OF WER WITH LAS VARIANTS. LA - LOCAL ATTENTION, SPECAUG- SPECTOGRAM AUGMENTATION

VIII. CONCLUSIONS

State of the art Automatic Speech Recognition models are trained on large amounts of data making using of multiple GPUs to facilitate parallel computations. Hence results of such architectures are difficult to reproduce and build up on. However we show that our proposed modifications to Deep Speech 2 and Listen, Attend and Spell models show competitive results when trained on a small subset of LibriSpeech which only contains 100hrs of data. We hypothesize that training for more epochs using deeper architectures on 960 hours of training data will give stable and more

generalizable results and show significant improvement in WER. We can further improve our models using Transformer models which have produced significant breakthroughs in sequence to sequence NLP tasks. In this work we have not explored speech processing techniques for augmentation and cleaning data which should be able to bolster performance.

We also can leverage Auxiliary learning tasks - tasks we accomplish with the sole objective of better primary tasks. Pseudo reward functions can be designed enable the model to learn the primary task efficiently.

REFERENCES

[1] Hochreiter, S., Bengio, Y., Frasconi, P., and Schmidhuber, J. Gradient Flow in Recurrent Nets: the Difficulty of Learning Long-term Dependencies. In Kremer, S. C. and Kolen, J. F. (eds.), A Field Guide to Dynamical Recurrent Neural Networks. 2001.

[2] Graves, A. and Schmidhuber, J. Offline handwriting recognition with multidimensional recurrent neural networks. In NIPS, 2008.

[3] Sutskever, I., Martens, J., and Hinton, G. Generating text with recurrent neural networks. In ICML, 2011.

[4] Mikolov, T., Karafit, M., Burget, L., Cernocky, J., and Khudanpur, S. Recurrent neural network based language model. In Eleventh Annual Conference of the International Speech Communication Association, 2010.

[5] Mikolov, T., Karafit, M., Burget, L., Cernocky, J., and Khudanpur, S. Recurrent neural network based language model. In Eleventh Annual Conference of the International Speech Communication Association, 2010.

[6] Mikolov, T., Karafit, M., Burget, L., Cernocky, J., and Khudanpur, S. Recurrent neural network based language model. In Eleventh Annual Conference of the International Speech Communication Association, 2010.

[7] Mikolov, T., Karafit, M., Burget, L., Cernocky, J., and Khudanpur, S. Recurrent neural network based language model. In Eleventh Annual Conference of the International Speech Communication Association, 2010.

[8] Mikolov, T., Karafit, M., Burget, L., Cernocky, J., and Khudanpur, S. Recurrent neural network based language model. In Eleventh Annual Conference of the International Speech Communication Association, 2010.

[9] Mikolov, T., Karafit, M., Burget, L., Cernocky, J., and Khudanpur, S. Recurrent neural network based language model. In Eleventh Annual Conference of the International Speech Communication Association, 2010.

[10] Mikolov, T., Karafit, M., Burget, L., Cernocky, J., and Khudanpur, S. Recurrent neural network based language model. In Eleventh Annual Conference of the International Speech Communication Association, 2010.

[11] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. 2014. http://arxiv.org/abs/1409.3215.

[12] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. In ICLR, 2015.

[13] J. Chorowski, D. Bahdanau, K. Cho, and Y. Bengio. End-to-end continuous speech recognition using attention-based recurrent nn: First results. abs/1412.1602, 2015. http://arxiv.org/abs/1412.1602.

[14] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In ICML, pages 369–376. ACM, 2006.

[15] A. Maas, Z. Xie, D. Jurafsky, and A. Ng. Lexicon-free conversational speech recognition with neural networks. In NAACL, 2015.

[16] Y. Miao, M. Gowayyed, and F. Metz. EESEN: End-to-end speech recognition using deep rnn models and wfst-based decoding. In ASRU, 2015.

[17] A. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates, and A. Y. Ng. Deep speech: Scaling up end-to-end speech recognition. 1412.5567, 2014. http://arxiv.org/abs/1412.5567.

[18] A. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates, and A. Y. Ng. Deep speech: Scaling up end-to-end speech recognition. 1412.5567, 2014. http://arxiv.org/abs/1412.5567.

[19] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur. Librispeech: an asr corpus based on public domain audio books. In ICASSP, 2015.

[20] O. Kapralova, J. Alex, E. Weinstein, P. Moreno, and O. Siohan. A big data approach to acoustic model training corpus selection. In Interspeech, 2014.