END-TO-END SPEECH RECOGNITION IN ENGLISH AND MANDARIN

Baidu Research – Silicon Valley AI Lab*
Dario Amodei, Rishita Anubhai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Jingdong Chen, Mike Chrzanowski, Adam Coates, Greg Diamos, Erich Elsen, Jesse Engel, Linxi Fan, Christopher Fougner, Tony Han, Awni Hannun, Billy Jun, Patrick LeGresley, Libby Lin, Sharan Narang, Andrew Ng, Sherjil Ozair, Ryan Prenger, Jonathan Raiman, Sanjeev Satheesh, David Seetapun, Shubho Sengupta, Yi Wang, Zhiqian Wang, Chong Wang, Bo Xiao, Dani Yogatama, Jun Zhan, Zhenyao Zhu

ABSTRACT

We show that an end-to-end deep learning approach can be used to recognize either English or Mandarin Chinese speech–two vastly different languages. Because it replaces entire pipelines of hand-engineered components with neural networks, end-to-end learning allows us to handle a diverse variety of speech including noisy environments, accents and different languages. Key to our approach is our application of HPC techniques, enabling experiments that previously took weeks to now run in days. This allows us to iterate more quickly to identify superior architectures and algorithms. As a result, in several cases, our system is competitive with the transcription of human workers when benchmarked on standard datasets. Finally, using a technique called Batch Dispatch with GPUs in the data center, we show that our system can be inexpensively deployed in an online setting, delivering low latency when serving users at scale.

1 INTRODUCTION

Decades worth of hand-engineered domain knowledge has gone into current state-of-the-art automatic speech recognition (ASR) pipelines. A simple but powerful alternative solution is to train such ASR models end-to-end, using deep learning to replace most modules with a single model as in Hannun et al. (2014a) and Miao et al. (2015). On such a system, built on end-to-end deep learning, we can employ a range of deep learning techniques: capturing large training sets, training larger models with high performance computing, and methodically exploring the space of neural network architectures.

This paper details our contribution to the model architecture, large labeled training datasets, and computational scale for speech recognition. This includes an extensive investigation of model architectures, and our data capturing pipeline that has enabled us to create larger datasets than what is typically used to train speech recognition systems.

We benchmark our system on several publicly available test sets with a goal of eventually attaining human-level performance not only on specific benchmarks, where it is possible to improve through dataset-specific tuning, but on a range of benchmarks that reflects a diverse set of scenarios where humans excel. To that end, we have also measured the performance of crowd workers on each benchmark for comparison. We find that our best Mandarin Chinese speech system transcribes short voice-query like utterances better than a typical Mandarin Chinese speaker.

The remainder of the paper is as follows. We begin with a review of related work in deep learning, end-to-end speech recognition, and scalability in Section 2. Section 3 describes the architectural and algorithmic improvements to the model and Section 4 explains how to efficiently compute them. We discuss the training data and steps taken to further augment the training set in Section 5. An analysis of results for our system in English and Mandarin is presented in Section 6. We end with a description of the steps needed to deploy our system to real users in Section 7.

[*] Authorship order is alphabetical.

2 Related Work

This work is inspired by previous work in both deep learning and speech recognition. Feed-forward neural network acoustic models were explored more than 20 years ago Bourlard & Morgan (1993); Renals et al. (1994). Recurrent neural networks and networks with convolution were also used in speech recognition around the same time Robinson et al. (1996); Waibel et al. (1989). More recently DNNs have become a fixture in the ASR pipeline with almost all state of the art speech work containing some form of deep neural network Mohamed et al. (2011); Hinton et al. (2012); Dahl et al. (2011); N. Jaitly & Vanhoucke (2012); Seide et al. (2011). Convolutional networks have also been found beneficial for acoustic models Abdel-Hamid et al. (2012); Sainath et al. (2013). Recurrent neural networks are beginning to be deployed in state-of-the art recognizers Graves et al. (2013); H. Sak et al. (2014) and work well with convolutional layers for the feature extraction Sainath et al. (2015).

End-to-end speech recognition is an active area of research, showing compelling results when used to re-score the outputs of a DNN-HMM Graves & Jaitly (2014) and standalone Hannun et al. (2014a). The RNN encoder-decoder paradigm uses an encoder RNN to map the input to a fixed length vector and a decoder network to expand the fixed length vector into a sequence of output predictions Cho et al. (2014); Sutskever et al. (2014). The RNN encoder-decoder with attention performs well in predicting phonemes Chorowski et al. (2015) or graphemes Bahdanau et al. (2015); Chan et al. (2015). The CTC loss function Graves et al. (2006) coupled with an RNN to model temporal information also performs well in end-to-end speech recognition with character outputs Graves & Jaitly (2014); Hannun et al. (2014b;a); Maas et al. (2015). The CTC-RNN model also works well in predicting phonemes Miao et al. (2015); Sak et al. (2015), though a lexicon is still needed in this case.

Exploiting scale in deep learning has been central to the success of the field thus far Krizhevsky et al. (2012); Le et al. (2012). Training on a single GPU resulted in substantial performance gains Raina et al. (2009), which were subsequently scaled linearly to two Krizhevsky et al. (2012) or more GPUs Coates et al. (2013). We take advantage of work in increasing individual GPU efficiency for low-level deep learning primitives Chetlur et al.. We built on the past work in using model-parallelism Coates et al. (2013), data-parallelism Dean et al. (2012) or a combination of the two Szegedy et al. (2014); Hannun et al. (2014a) to create a fast and highly scalable system for training deep RNNs in speech recognition.

Data has also been central to the success of end-to-end speech recognition, with over 7000 hours of labeled speech used in Hannun et al. (2014a). Data augmentation has been highly effective in improving the performance of deep learning in computer vision LeCun et al. (2004); Sapp et al. (2008); Coates et al. (2011) and speech recognition Gales et al. (2009); Hannun et al. (2014a). Existing speech systems can also be used to bootstrap new data collection. For example, an existing speech engine can be used to align and filter thousands of hours of audiobooks Panayotov et al. (2015). We draw inspiration from these past approaches in bootstrapping larger datasets and data augmentation to increase the effective amount of labeled data for our system.

3 Model Architecture

Figure 1 shows the wireframe of our architecture, and lays out the swappable components which we explore in detail in this paper. Our system (similar at its core to the one in Hannun et al. (2014a)), is 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. [1]

The inputs to the network are a sequence of log-spectrograms of power normalized audio clips, calculated on 20ms windows. The outputs are the alphabet of each language. At each output time-step t, the RNN makes a prediction, $p(\ell_t|x)$, where ℓ_t is either a character in the alphabet or the blank symbol. In English we have $\ell_t \in \{$a, b, c, $\ldots,$ z, *space*, *apostrophe*, *blank*$\}$, where we have added the *space* symbol to denote word boundaries. For the Mandarin system the network outputs simplified Chinese characters.

[1] Most of our experiments use bidirectional recurrent layers with clipped rectified-linear units (ReLU) $\sigma(x) = \min\{\max\{x, 0\}, 20\}$ as the activation function.

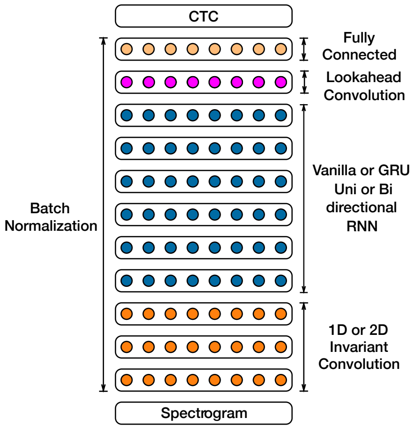
CTC
Fully Connected
Lookahead Convolution
Vanilla or GRU Uni or Bi directional RNN
Batch Normalization
1D or 2D Invariant Convolution
Spectrogram

Figure 1: Architecture of the deep RNN used in both English and Mandarin speech.

Architecture	Baseline	BatchNorm	GRU
5-layer, 1 RNN	13.55	14.40	10.53
5-layer, 3 RNN	11.61	10.56	8.00
7-layer, 5 RNN	10.77	9.78	**7.79**
9-layer, 7 RNN	10.83	9.52	8.19
9-layer, 7 RNN no SortaGrad	11.96	9.78	

Table 1: Comparison of WER on a development set as we vary depth of RNN, application of BatchNorm and SortaGrad, and type of recurrent hidden unit. All networks have 38M parameters—as depth increases, the number of hidden units per layer decreases. The last two columns compare the performance of the model on the dev set as we change the type of the recurrent hidden unit.

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

$$Q(y) = \log(p_{\text{RNN}}(y|x)) + \alpha \log(p_{\text{LM}}(y)) + \beta \text{wc}(y) \tag{1}$$

where $\text{wc}(y)$ is the number of words (English) or characters (Chinese) 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. These parameters are tuned on a held out development set.

3.1 Batch Normalization for Deep RNNs

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

$$h_t^l = f(W^l h_t^{l-1} + U^l h_{t-1}^l + b). \tag{2}$$

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

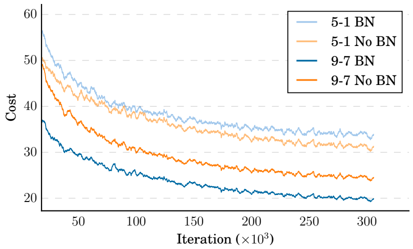

Figure 2: Training curves of two models trained with and without BatchNorm (BN). We see a wider gap in performance on the deeper 9-7 network (which has 9 layers in total, 7 of which are vanilla bidirectional RNNs) than the shallower 5-1 network (in which only 1 of the 5 layers is a bidirectional RNN). We start the plot after the first epoch of training as the curve is more difficult to interpret due to the SortaGrad curriculum method mentioned in Section 3.2

As in Laurent et al. (2015), there are two ways of applying BatchNorm to the recurrent operation. A natural extension is to insert a BatchNorm transformation, $\mathcal{B}(\cdot)$, immediately before every non-linearity as follows:

$$h_t^l = f(\mathcal{B}(W^l h_t^{l-1} + U^l h_{t-1}^l)). \tag{3}$$

In this case the mean and variance statistics are accumulated over a single time-step of the minibatch. We did not find this to be effective.

An alternative (*sequence-wise* normalization) is to batch normalize only the vertical connections. The recurrent computation is given by

$$h_t^l = f(\mathcal{B}(W^l h_t^{l-1}) + U^l h_{t-1}^l). \tag{4}$$

For each hidden unit we compute the mean and variance statistics over all items in the minibatch over the length of the sequence. Figure 2 shows that deep networks converge faster with sequence-wise normalization. Table 1 shows that the performance improvement from sequence-wise normalization increases with the depth of the network, with a 12% performance difference for the deepest network. We store a running average of the mean and variance for the neuron collected during training, and use these for evaluation Ioffe & Szegedy (2015).

3.2 SortaGrad

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. This observation motivates a curriculum learning strategy we title SortaGrad: we use the length of the utterance as a heuristic for difficulty and train on the shorter (easier) utterances first.

Specifically, in the first training epoch we iterate through minibatches in the training set in increasing order of the length of the longest utterance in the minibatch. After the first epoch training reverts back to a random order over minibatches. Table 1 shows a comparison of training cost with and without SortaGrad on the 9 layer model with 7 recurrent layers. SortaGrad improves the stability of training, and this effect is particularly pronounced in networks without BatchNorm, since these are even less numerically stable.

Both GRU and vanilla RNN architectures benefit from BatchNorm and show strong results with deep networks. The last two columns in table 1 show that for a fixed number of parameters the GRU architecture achieves better WER for all network depths.

Architecture	Channels	Filter dimension	Stride	Regular Dev	Noisy Dev
1-layer 1D	1280	11	2	9.52	19.36
2-layer 1D	640, 640	5, 5	1, 2	9.67	19.21
3-layer 1D	512, 512, 512	5, 5, 5	1, 1, 2	9.20	20.22
1-layer 2D	32	41x11	2x2	8.94	16.22
2-layer 2D	32, 32	41x11, 21x11	2x2, 2x1	9.06	15.71
3-layer 2D	32, 32, 96	41x11, 21x11, 21x11	2x2, 2x1, 2x1	8.61	14.74

Table 2: Comparison of WER for different configurations of convolutional layers. In all cases, the convolutions are followed by 7 recurrent layers and 1 fully connected layer. For 2D convolutions the first dimension is frequency and the second dimension is time. Each model is trained with BatchNorm, SortaGrad, and has 35M parameters.

3.3 Frequency Convolutions

Temporal convolution is commonly used in speech recognition to efficiently model temporal translation invariance for variable length utterances. Convolution in frequency attempts to model spectral variance due to speaker variability more concisely than what is possible with large fully connected networks.

We experiment with adding between one and three layers of convolution. These are both in the time-and-frequency domain (2D) and in the time-only domain (1D). In all cases we use a "*same*" convolution. In some cases we specify a stride (subsampling) across either dimension which reduces the size of the output.

We report results on two datasets—a development set of 2048 utterances ("Regular Dev") and a much noisier dataset of 2048 utterances ("Noisy Dev") randomly sampled from the CHiME 2015 development datasets Barker et al. (2015). We find that multiple layers of 1D convolution provides a very small benefit. The 2D convolutions improve results substantially on noisy data, while providing a small benefit on clean data. The change from one layer of 1D convolution to three layers of 2D convolution improves WER by 23.9% on the noisy development set.

3.4 Lookahead Convolution and Unidirectional Models

Bidirectional RNN models are challenging to deploy in an online, low-latency setting because they cannot stream the transcription process as the utterance arrives from the user. However, models with only forward recurrences routinely perform worse than similar bidirectional models, implying some amount of future context is vital to good performance. One possible solution is to delay the system from emitting a label until it has more context as in Sak et al. (2015), but we found it difficult to induce this behavior in our models. In order to build a unidirectional model without any loss in accuracy, we develop a special layer that we call lookahead convolution, shown in Figure 3. The layer learns weights to linearly combine each neuron's activations τ timesteps into the future, and thus allows us to control the amount of future context needed. The lookahead layer is defined by a parameter matrix $W \in \mathbb{R}^{(d,\tau)}$, where d matches the number of neurons in the previous layer. The activations r_t for the new layer at time-step t are

$$r_{t,i} = \sum_{j=1}^{\tau+1} W_{i,j} h_{t+j-1,i}, \text{ for } 1 \leq i \leq d. \tag{5}$$

We place the lookahead convolution above all recurrent layers. This allows us to stream all computation below the lookahead convolution on a finer granularity.

3.5 Adaptation to Mandarin

To port a traditional speech recognition pipeline to another language typically requires a significant amount of new language-specific development. For example, one often needs to hand-engineer a pronunciation model Shan et al. (2010). We may also need to explicitly model language-specific pronunciation features, such as tones in Mandarin Shan et al. (2010); Niu et al. (2013). Since our end-to-end system directly predicts characters, these time consuming efforts are no longer needed.

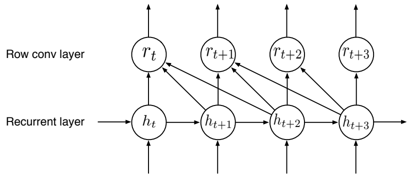

Figure 3: Lookahead convolution architecture with future context size of 2.

This has enabled us to quickly create an end-to-end Mandarin speech recognition system (that outputs Chinese characters) using the approach described above with only a few changes.

The only architectural changes we make to our networks are due to the characteristics of the Chinese character set. The network outputs probabilities for about 6000 characters, which includes the Roman alphabet, since hybrid Chinese-English transcripts are common. We incur an out of vocabulary error at evaluation time if a character is not contained in this set. This is not a major concern, as our test set has only 0.74% out of vocab characters.

We use a character level language model in Mandarin as words are not usually segmented in text. In Section 6.2 we show that our Mandarin speech models show roughly the same improvements to architectural changes as our English speech models, suggesting that modeling knowledge from development in one language transfers well to others.

4 System Optimizations

Our networks have tens of millions of parameters, and a training experiment involves tens of single-precision exaFLOPs. Since our ability to evaluate hypotheses about our data and models depends on training speed, we created a highly optimized training system based on high performance computing (HPC) infrastructure.[2] Although many frameworks exist for training deep networks on parallel machines, we have found that our ability to scale well is often bottlenecked by unoptimized routines that are taken for granted. Therefore, we focus on careful optimization of the most important routines used for training. Specifically, we created customized All-Reduce code for OpenMPI to sum gradients across GPUs on multiple nodes, developed a fast implementation of CTC for GPUs, and use custom memory allocators. Taken together, these techniques enable us to sustain *overall* 45% of theoretical peak performance on each node.

Our training distributes work over multiple GPUs in a data-parallel fashion with synchronous SGD, where each GPU uses a local copy of the model to work on a portion of the current minibatch and then exchanges computed gradients with all other GPUs. We prefer synchronous SGD because it is reproducible, which facilitates discovering and fixing regressions. In this setup, however, the GPUs must communicate quickly (using an "All-Reduce" operation) at each iteration in order to avoid wasting computational cycles. Prior work has used asynchronous updates to mitigate this issue Dean et al. (2012); Recht et al. (2011). We instead focused on optimizing the All-Reduce operation itself, achieving a 4x-21x speedup using techniques to reduce CPU-GPU communication for our specific workloads. Similarly, to enhance overall computation, we have used highly-optimized kernels from Nervana Systems and NVIDIA that are tuned for our deep learning applications. We similarly discovered that custom memory allocation routines were crucial to maximizing performance as they reduce the number of synchronizations between GPU and CPU.

We also found that the CTC cost computation accounted for a significant fraction of running time. Since no public well-optimized code for CTC existed, we developed a fast GPU implementation that reduced overall training time by 10-20%.[3]

[2] Our software runs on dense compute nodes with 8 NVIDIA Titan X GPUs per node with a theoretical peak throughput of 48 single-precision TFLOP/s.

[3] Details of our CTC implementation will be made available along with open source code.

Fraction of Data	Hours	Regular Dev	Noisy Dev
1%	120	29.23	50.97
10%	1200	13.80	22.99
20%	2400	11.65	20.41
50%	6000	9.51	15.90
100%	12000	8.46	13.59

Table 3: Comparison of English WER for Regular and Noisy development sets on increasing training dataset size. The model has 9-layers (2 layers of 2D convolution and 7 recurrent layers) with 68M parameters.

5 Training Data

Large-scale deep learning systems require an abundance of labeled training data. For training our English model, we use 11,940 hours of labeled speech containing 8 million utterances, and the Mandarin system uses 9,400 hours of labeled speech containing 11 million utterances.

5.1 Dataset Construction

Parts of the English and Mandarin datasets were created from raw data captured as long audio clips with noisy transcriptions. In order to segment the audio into several second long clips, we align the speech with the transcript. For a given audio-transcript pair (x, y), the most likely alignment is calculated as

$$\ell^* = \underset{\ell \in \text{Align}(x,y)}{\arg\max} \prod_t^T p_{\text{ctc}}(\ell_t | x; \theta). \tag{6}$$

This is essentially a Viterbi alignment found using a RNN model trained with CTC. Since the CTC loss function integrates over all alignments, this is not guaranteed to produce an accurate alignment. However, we found that this approach produces an accurate alignment when using a bidirectional RNN.

In order to filter out clips with poor transcriptions, we build a simple classifier with the following features: the raw CTC cost, the CTC cost normalized by the sequence length, the CTC cost normalized by the transcript length, the ratio of the sequence length to the transcript length, the number of words in the transcription and the number of characters in the transcription. We crowd source the labels for building this dataset. For the English dataset, we find that the filtering pipeline reduces the WER from 17% to 5% while retaining more than 50% of the examples.

Additionally, we dynamically augment the dataset by adding unique noise every epoch with an SNR between 0dB and 30dB, just as in Hannun et al. (2014a); Sainath et al. (2015).

5.2 Scaling Data

We show the effect of increasing the amount of labeled training data on WER in Table 3. This is done by randomly sampling the full dataset before training. For each dataset, the model was trained for up to 20 epochs with early-stopping based on the error on a held out development set to prevent overfitting. The WER decreases by ~40% relative for each factor of 10 increase in training set size. We also observe a consistent gap in WER (~60% relative) between the regular and noisy datasets, implying that more data benefits both cases equally.

6 Results

To better assess the real-world applicability of our speech system, we evaluate on a wide range of test sets. We use several publicly available benchmarks and several test sets collected internally. All models are trained for 20 epochs on either the full English dataset, or the full Mandarin dataset described in Section 5. We use stochastic gradient descent with Nesterov momentum Sutskever et al. (2013) along with a minibatch of 512 utterances. If the norm of the gradient exceeds a threshold of 400, it is rescaled to 400 Pascanu et al. (2012). The model which performs the best on a held-out development set during training is chosen for evaluation. The learning rate is chosen from $[1 \times 10^{-4}, 6 \times 10^{-4}]$ to yield fastest convergence and annealed by a constant factor of 1.2 after each epoch. We use a momentum of 0.99 for all models.

	Test set	Ours	Human
Read	WSJ eval'92	3.10	5.03
	WSJ eval'93	4.42	8.08
	LibriSpeech test-clean	5.15	5.83
	LibriSpeech test-other	12.73	12.69
Accented	VoxForge American-Canadian	7.94	4.85
	VoxForge Commonwealth	14.85	8.15
	VoxForge European	18.44	12.76
	VoxForge Indian	22.89	22.15
Noisy	CHiME eval real	21.59	11.84
	CHiME eval sim	42.55	31.33

Table 4: Comparison of WER for our speech system and human level performance.

6.1 English

The best English model has 2 layers of 2D convolution, followed by 3 layers of unidirectional recurrent layers with 2560 GRU cells each, followed by a lookahead convolution layer with $\tau = 80$, trained with BatchNorm and SortaGrad. We do not adapt the model to any of the speech conditions in the test sets. Language model decoding parameters are set once on a held-out development set.

We report results on several test sets for both our system and an estimate of human accuracy. We obtain a measure of human level performance by asking workers from Amazon Mechanical Turk to hand-transcribe all of our test sets. Two workers transcribe the same audio clip that is on average about 5 seconds long. We then take the better of the two transcriptions for the final WER calculation. Most workers are based in the United States and on average spend 27 seconds per transcription. The hand-transcribed results are compared to the existing ground truth to produce a WER estimate. While the existing ground truth transcriptions do have some label error, this is rarely more than 1%. This implies that disagreement between the ground truth transcripts and the crowd sourced transcripts is a good proxy for actual accuracy of a typical human worker.

6.1.1 Benchmark results

Read speech with high signal-to-noise ratio is arguably the easiest task in large vocabulary continuous speech recognition. We benchmark our system on two test sets from the Wall Street Journal (WSJ) corpus of read news articles and the LibriSpeech corpus constructed from audio books Panayotov et al. (2015). Table 4 shows that our system outperforms humans on 3 out of 4 test sets.

We also tested our system for robustness to common accents using the VoxForge (`http://www.voxforge.org`) dataset. The set contains speech read by speakers with many different accents. We group these accents into four categories: American-Canadian, Indian, Commonwealth[4] and European[5]. We construct a test set from the VoxForge data with 1024 examples from each accent group for a total of 4096 examples. Human level performance is still notably better than that of our system for all but the Indian accent.

Finally, we tested our performance on noisy speech using the test sets from the recently completed third CHiME challenge Barker et al. (2015). This dataset has utterances from the WSJ test set collected in real noisy environments and with artificially added noise. Using all 6 channels of the CHiME audio can provide substantial performance improvements Yoshioka et al. (2015). We use a *single* channel for all our models, since access to multi-channel audio is not pervasive on most devices. The gap between our system and human level performance is larger when the data comes from a real noisy environment instead of synthetically adding noise to clean speech.

6.2 Mandarin

In Table 5 we compare several architectures trained on Mandarin Chinese speech on a development set of 2000 utterances as well as a test set of 1882 examples of noisy speech. This development set

[4] "Commonwealth" includes British, Irish, South African, Australian and New Zealand accents.

[5] "European" includes countries in Europe without English as a first language.

was also used to tune the decoding parameters. We see that the deepest model with 2D convolution and BatchNorm outperforms the shallow RNN by 48% relative.

Architecture	Dev	Test
5-layer, 1 RNN	7.13	15.41
5-layer, 3 RNN	6.49	11.85
5-layer, 3 RNN + BatchNorm	6.22	9.39
9-layer, 7 RNN + BatchNorm + frequency Convolution	5.81	7.93

Table 5: Comparison of the different RNN architectures. The development and test sets are internal corpora. All the models in the table have about 80 million parameters each.

Test	Human	RNN
100 utterances / committee	4.0	3.7
250 utterances / individual	9.7	5.7

Table 6: We benchmark the best Mandarin system against humans on two randomly selected test sets. The first set has 100 examples and is labelled by a committee of 5 Chinese speakers. The second has 250 examples and is labelled by a single human transcriber.

Table 6 shows that our best Mandarin Chinese speech system transcribes short voice-query like utterances better than a typical Mandarin Chinese speaker and a committee of 5 Chinese speakers working together.

7 Deployment

Bidirectional models are not well-designed for real time transcription: since the RNN has several bidirectional layers, transcribing an utterance requires the entire utterance to be presented to the RNN; and since we use a wide beam search for decoding, beam search can be expensive.

We modify our system to produce a model that is almost as accurate (there is only 5% relative difference between the research and production systems) while having much lower latency. We focus on our Mandarin system, but the same techniques could also be applied in English. We focus on latency measured from when the end of an utterance is received until the transcription is computed, because this latency is important to applications using speech recognition.

To increase deployment scalability, while still providing low latency transcription, we built a batching scheduler called Batch Dispatch that assembles streams of data from user requests into batches before performing RNN forward propagation on these batches. With this scheduler, we can trade increased batch size, and consequently improved efficiency, with increased latency.

We use an eager batching scheme that processes each batch as soon as the previous batch is completed, regardless of how much work is ready by that point. This scheduling algorithm balances efficiency and latency, achieving relatively small dynamic batch sizes up to 10 samples per batch, with median batch size proportional to server load.

Load	Median	98%ile
10 streams	44	70
20 streams	48	86
30 streams	67	114

Table 7: Latency distribution (ms) versus load

We see in Table 7 that our system achieves a median latency of 44 ms, and a 98th percentile latency of 70 ms when loaded with 10 concurrent streams. This server uses one NVIDIA Quadro K1200 GPU for RNN evaluation. As designed, Batch Dispatch shifts work to larger batches as server load grows, keeping latency low.

Our deployment system evaluates RNNs in half-precision arithmetic, which has no measurable accuracy impact, but significantly improves efficiency. We wrote our own 16-bit matrix-matrix multiply routines for this task, substantially improving throughput for our relatively small batches.

Performing the beam search involves repeated lookups in the *n*-gram language model, most of which translate to uncached reads from memory. To reduce the cost of these lookups, we employ a heuristic: only consider the fewest number of characters whose cumulative probability is at least p. In practice, we have found that $p = 0.99$ works well, and additionally we limit the search to 40 characters. This speeds up the cumulative Mandarin language model lookup time by a factor of 150x, and has a negligible effect on the CER (0.1-0.3% relative).

8 Conclusion

End-to-end deep learning presents the exciting opportunity to improve speech recognition systems continually with increases in data and computation. Since the approach is highly generic, we have shown that it can quickly be applied to new languages. Creating high-performing recognizers for two very different languages, English and Mandarin, required essentially no expert knowledge of the languages. Finally, we have also shown that this approach can be efficiently deployed by batching user requests together on a GPU server, paving the way to deliver end-to-end Deep Learning technologies to users.

To achieve these results, we have explored various network architectures, finding several effective techniques: enhancements to numerical optimization through SortaGrad and Batch Normalization, and lookahead convolution for unidirectional models. This exploration was powered by a well optimized, high performance computing inspired training system that allows us to train full-scale models on our large datasets in just a few days.

Overall, we believe our results confirm and exemplify the value of end-to-end deep learning methods for speech recognition in several settings. We believe these techniques will continue to scale.

References

Ossama Abdel-Hamid, Abdel-rahman Mohamed, Hui Jang, and Gerald Penn. Applying convolutional neural networks concepts to hybrid nn-hmm model for speech recognition. In *ICASSP*, 2012.

Dzmitry Bahdanau, Jan Chorowski, Dmitriy Serdyuk, Philemon Brakel, and Yoshua Bengio. End-to-end attention-based large vocabulary speech recognition. abs/1508.04395, 2015. http://arxiv.org/abs/1508.04395.

Jon Barker, Emmanuel Marxer, Ricard Vincent, and Shinji Watanabe. The third 'CHiME' speech separation and recognition challenge: Dataset, task and baselines. 2015. Submitted to IEEE 2015 Automatic Speech Recognition and Understanding Workshop (ASRU).

Yoshua Bengio, JérĘome Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *International Conference on Machine Learning*, 2009.

H. Bourlard and N. Morgan. *Connectionist Speech Recognition: A Hybrid Approach*. Kluwer Academic Publishers, Norwell, MA, 1993.

William Chan, Navdeep Jaitly, Quoc Le, and Oriol Vinyals. Listen, attend, and spell. abs/1508.01211, 2015. http://arxiv.org/abs/1508.01211.

Sharan Chetlur, Cliff Woolley, Philippe Vandermersch, Jonathan Cohen, John Tran, Bryan Catanzaro, and Evan Shelhamer. cuDNN: Efficient primitives for deep learning. URL `http://arxiv.org/abs/1410.0759`.

Kyunghyun Cho, Bart Van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *EMNLP*, 2014.

Jan Chorowski, Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. End-to-end continuous speech recognition using attention-based recurrent nn: First results. abs/1412.1602, 2015. http://arxiv.org/abs/1412.1602.

Adam Coates, Blake Carpenter, Carl Case, Sanjeev Satheesh, Bipin Suresh, Tao Wang, David J. Wu, and Andrew Y. Ng. Text detection and character recognition in scene images with unsupervised feature learning. In *International Conference on Document Analysis and Recognition*, 2011.

Adam Coates, Brody Huval, Tao Wang, David J. Wu, Andrew Y. Ng, and Bryan Catanzaro. Deep learning with COTS HPC. In *International Conference on Machine Learning*, 2013.

G.E. Dahl, D. Yu, L. Deng, and A. Acero. Context-dependent pre-trained deep neural networks for large vocabulary speech recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 2011.

Jeffrey Dean, Greg S. Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Quoc Le, Mark Mao, MarcâĂŹAurelio Ranzato, Andrew Senior, Paul Tucker, Ke Yang, and Andrew Ng. Large scale distributed deep networks. In *Advances in Neural Information Processing Systems 25*, 2012.

M. J. F. Gales, A. Ragni, H. Aldamarki, and C. Gautier. Support vector machines for noise robust ASR. In *ASRU*, pp. 205–2010, 2009.

A. Graves and N. Jaitly. Towards end-to-end speech recognition with recurrent neural networks. In *ICML*, 2014.

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

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *ICASSP*, 2013.

Hasim H. Sak, Andrew Senior, and Francoise Beaufays. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In *Interspeech*, 2014.

Awni Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, and Andrew Y. Ng. Deep speech: Scaling up end-to-end speech recognition. 1412.5567, 2014a. http://arxiv.org/abs/1412.5567.

Awni Y. Hannun, Andrew L. Maas, Daniel Jurafsky, and Andrew Y. Ng. First-pass large vocabulary continuous speech recognition using bi-directional recurrent DNNs. abs/1408.2873, 2014b. http://arxiv.org/abs/1408.2873.

G.E. Hinton, L. Deng, D. Yu, G.E. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, and B. Kingsbury. Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal Processing Magazine*, 29(November):82–97, 2012.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. abs/1502.03167, 2015. http://arxiv.org/abs/1502.03167.

Alex Krizhevsky, Ilya Sutskever, and Geoff Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*, pp. 1106–1114, 2012.

Cesar Laurent, Gabriel Pereyra, Philemon Brakel, Ying Zhang, and Yoshua Bengio. Batch normalized recurrent neural networks. abs/1510.01378, 2015. http://arxiv.org/abs/1510.01378.

Q.V. Le, M.A. Ranzato, R. Monga, M. Devin, K. Chen, G.S. Corrado, J. Dean, and A.Y. Ng. Building high-level features using large scale unsupervised learning. In *International Conference on Machine Learning*, 2012.

Yann LeCun, Fu Jie Huang, and Léon Bottou. Learning methods for generic object recognition with invariance to pose and lighting. In *Computer Vision and Pattern Recognition*, volume 2, pp. 97–104, 2004.

Andrew Maas, Ziang Xie, Daniel Jurafsky, and Andrew Ng. Lexicon-free conversational speech recognition with neural networks. In *NAACL*, 2015.

Yajie Miao, Mohammad Gowayyed, and Florian Metz. EESEN: End-to-end speech recognition using deep rnn models and wfst-based decoding. In *ASRU*, 2015.

A. Mohamed, G.E. Dahl, and G.E. Hinton. Acoustic modeling using deep belief networks. *IEEE Transactions on Audio, Speech, and Language Processing*, (99), 2011. URL `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5704567`.

A. Senior N. Jaitly, P. Nguyen and V. Vanhoucke. Application of pretrained deep neural networks to large vocabulary speech recognition. In *Interspeech*, 2012.

Jianwei Niu, Lei Xie, Lei Jia, and Na Hu. Context-dependent deep neural networks for commercial mandarin speech recognition applications. In *APSIPA*, 2013.

Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. Librispeech: an asr corpus based on public domain audio books. In *ICASSP*, 2015.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. abs/1211.5063, 2012. http://arxiv.org/abs/1211.5063.

R. Raina, A. Madhavan, and A.Y. Ng. Large-scale deep unsupervised learning using graphics processors. In *26th International Conference on Machine Learning*, 2009.

Benjamin Recht, Christopher Re, Stephen Wright, and Feng Niu. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *Advances in Neural Information Processing Systems*, pp. 693–701, 2011.

S. Renals, N. Morgan, H. Bourlard, M. Cohen, and H. Franco. Connectionist probability estimators in HMM speech recognition. *IEEE Transactions on Speech and Audio Processing*, 2(1):161–174, 1994.

Tony Robinson, Mike Hochberg, and Steve Renals. The use of recurrent neural networks in continuous speech recognition. pp. 253–258, 1996.

Tara Sainath, Oriol Vinyals, Andrew Senior, and Hasim Sak. Convolutional, long short-term memory, fully connected deep neural networks. In *ICASSP*, 2015.

Tara N. Sainath, Abdel rahman Mohamed, Brian Kingsbury, and Bhuvana Ramabhadran. Deep convolutional neural networks for LVCSR. In *ICASSP*, 2013.

Hasim Sak, Andrew Senior, Kanishka Rao, and Francoise Beaufays. Fast and accurate recurrent neural network acoustic models for speech recognition. abs/1507.06947, 2015. http://arxiv.org/abs/1507.06947.

Benjaminn Sapp, Ashutosh Saxena, and Andrew Ng. A fast data collection and augmentation procedure for object recognition. In *AAAI Twenty-Third Conference on Artificial Intelligence*, 2008.

Frank Seide, Gang Li, and Dong Yu. Conversational speech transcription using context-dependent deep neural networks. In *Interspeech*, pp. 437–440, 2011.

Jiulong Shan, Genqing Wu, Zhihong Hu, Xiliu Tang, Martin Jansche, and Pedro Moreno. Search by voice in mandarin chinese. In *Interspeech*, 2010.

I. Sutskever, J. Martens, G. Dahl, and G. Hinton. On the importance of momentum and initialization in deep learning. In *30th International Conference on Machine Learning*, 2013.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. 2014. http://arxiv.org/abs/1409.3215.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. 2014.

Alexander Waibel, Toshiyuki Hanazawa, Geoffrey Hinton, Kiyohiro Shikano, and Kevin Lang. Phoneme recognition using time-delay neural networks,âĂİ acoustics speech and signal processing. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 37(3):328–339, 1989.

T. Yoshioka, N. Ito, M. Delcroix, A. Ogawa, K. Kinoshita, M. F. C. Yu, W. J. Fabian, M. Espi, T. Higuchi, S. Araki, and T. Nakatani. The ntt chime-3 system: Advances in speech enhancement and recognition for mobile multi-microphone devices. In *IEEE ASRU*, 2015.

Wojciech Zaremba and Ilya Sutskever. Learning to execute. abs/1410.4615, 2014. http://arxiv.org/abs/1410.4615.