RECURRENT POISSON PROCESS UNIT FOR AUTOMATIC SPEECH RECOGNITION

by

Hengguan Huang

A Thesis Submitted to
The Hong Kong University of Science and Technology
in Partial Fulfillment of the Requirements for
the Degree of Master of Philosophy
in Computer Science and Engineering

November 2018, Hong Kong

Authorization

HENGGUAN HUANG

5 December 2018

RECURRENT POISSON PROCESS UNIT FOR AUTOMATIC SPEECH RECOGNITION

by

Hengguan Huang

This is to certify that I have examined the above M.Phil. thesis

and have found that it is complete and satisfactory in all respects,

and that any and all revisions required by

the thesis examination committee have been made.

Associate Prof. Brian Mak, Thesis Supervisor

Prof. Dit-Yan YEUNG, Acting Head of Department

Department of Computer Science and Engineering

12 November 2018

ACKNOWLEDGMENTS

I am grateful that I was able to study and work with Professor Brian Mak. He give me guidance and advice throughout my postgraduate study.

I thank the members of my thesis committee, Professor Dit-Yan Yeung and Professor Yangqiu Song, for their insightful comments on improving this work.

I thank my colleagues in HKUST – Dr. Hao Wang, Dr. Xiaodong Gu,Zhaoyu Liu, Miss YingKe Zhu, Mr. Wei Li and many others. Without them, my graduate study in HKUST would not be so colorful.

Last but not least, I thank my parents and my girl friend, for their support and encouragement.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

RECURRENT POISSON PROCESS UNIT FOR AUTOMATIC SPEECH RECOGNITION

by

Hengguan Huang

Department of Computer Science and Engineering

The Hong Kong University of Science and Technology

ABSTRACT

Over the past few years, there has been a resurgence of interest in using recurrent neural network-hidden Markov model (RNN-HMM) for automatic speech recognition (ASR). Some modern recurrent network models, such as long short-term memory (LSTM) and simple recurrent unit (SRU), have demonstrated promising results on this task. Recently, several scientific perspectives in the fields of neuroethology and speech production suggest that human speech signals may be represented in discrete point patterns involving acoustic events in the speech signal. Based on this hypothesis, it may pose some challenges for RNN-HMM acoustic modeling: firstly, it arbitrarily discretizes the continuous input into the interval features at a fixed frame rate, which may introduce discretization errors; secondly, the occurrences of such acoustic events are unknown. Furthermore, the training targets of RNN-HMM are obtained from other (inferior) models, giving rise to misalignments.

On the other hand, the temporal point process is a powerful mathematical tool to describe the latent mechanisms governing the occurrences of observed random events. It is a random process whose realization consists of a sequence of isolated events with their time-stamps. Due to their generality, point processes have been widely used for modeling phenomena such as earthquakes, human activities, financial data, context-aware recommendations, etc. Major research in this area focuses on exploring the observed event data to model the underlying dynamics of the system, while our work attempts to deal with the situation where acoustic events are not available/observed even during training.

In this paper, we propose a recurrent Poisson process (RPP) which can be seen as a collection of Poisson processes at a series of time intervals in which the intensity evolves according to the RNN hidden states that encode the history of the acoustic signal. It aims at allocating the latent acoustic events in continuous time. Such events are efficiently drawn from the RPP using a sampling-free solution in an analytic form. The speech signal containing latent acoustic events is reconstructed/sampled dynamically from the discretized acoustic features using linear interpolation, in which the weight parameters are estimated from the onset of these events. The above processes are further integrated into an SRU, forming our final model, called recurrent Poisson process unit (RPPU). Experimental evaluations on ASR tasks including ChiME-2, WSJ0 and WSJ0&1 demonstrate the effectiveness and benefits of the RPPU. For example, it achieves a relative WER reduction of 10.7% over state-of-the-art models on WSJ0.

CHAPTER 1

INTRODUCTION

Automatic speech recognition (ASR) [54] can be formulated as the problem of recognizing words from uttered speech signals given a dictionary and some other prior knowledge of the problem. In the beginning, many researchers believed that ASR would become an easy task with the development of new computer technologies. However, a few decades later, we are realizing that it is still a very hard problem, and even nowadays many difficulties are far from being solved. These include an increasingly huge dictionary, free-style tasks, spontaneous speech, robustness to environmental conditions, mixed languages and so on. The details of these problems are described in [71].

1.1 Background

Since 1980s, hidden Markov model (HMM) Gaussian mixture model (GMM) [53, 34] has been the most adopted parametric model at the acoustic level. An HMM statistically describe temporal dependency of the acoustic units and GMMs are used to approximate the statistical distributions of phonemes. In the meantime, many researchers had begun to investigate artificial neural networks (ANNs) [69, 59, 58] for ASR. However, the ANN acoustic model is hard to train due to limited computing and data resources. In the past few years, several deep neural network [47, 31] based approaches have been proposed and demonstrated promising results. Hinton [47] in particular applied deep neural networks (DNNs) that were initialized with the parameters from a pre-trained stacked RBM [60] to replace GMM for approximating HMM state likelihoods. The resulting DNN-HMM hybrid model dramatically outperforms the GMM-HMM model. Although this model can encode contextual information into the hidden layers, it cannot capture the dependency between any two consecutive inputs, which is very important for sequence modeling.

The gain of DNN-HMM hybrid system mainly comes from exploiting information from contextual frames [31]. Using relatively longer contexts yields higher accuracy, while too long a context length leads to convergence difficulties in DNN training. Recurrent neural networks (RNNs) [26] alleviate this problem by introducing feedback cycles into the network architecture and modeling the dependency among the observerd inputs. As a result, the context window was extended to infinite, which can theoretically handle any arbitrary context window. However, the RNN training process encounters the vanishing or exploding gradient problem [33] when it is performed by backpropagation through time (BPTT) [70] and stochaistic gradient descent (SGD) [57]. Long short-term memory(LSTM) [33] is a special implementation of a recurrent neural network (RNN) that is easy to train and partially addresses this drawback. Perhaps the most commonly adopted acoustic model in hybrid ASR systems is the long short term memory recurrent neural network hidden Markov model (LSTM RNN-HMM).

It is beneficial for an acoustic model to capture long-term dependencies of the observations at different times. However, the sequential gates computation of LSTM limits its parallelization potential. Simple recurrent unit [40] and quasi-RNN [8], simplify the implementation of LSTM-RNN, and increase the speed of computation for each processing step by dropping the connections between the hidden states and the LSTM gates, allowing them to be computed in parallel. To speed up the acoustic model training, this thesis use SRU as a basic building block to construct the proposed recurrent Poisson process unit (RPPU) .

1.2 Motivation

A temporal point process is a powerful mathematical tool to describe the latent mechanisms governing the occurrences of observed random events. It is a random process whose realization consists of a sequence of isolated events with their time-stamps. Due to their generality, point processes have been widely used for modeling phenomena such as earthquakes [28], human activities [43], financial data [1], context-aware recommendations [14] , etc. A common property of the problems above is that the precise event time intervals can carry important information about the underlying dynamics, which otherwise are not available from the sequence of interval features that are evenly sampled from the continuous signal. Major research in this area focuses on exploring the observed event data to model the underlying dynamics of the system, while our work attempts to deal with the situation where acoustic events are not available/observed even during training.

The hybrid RNN-HMM in acoustic modeling is essentially a generalized version of a dynamic Bayesian network (DBN), which is usually characterized by discretizing the time series data and capturing the dependency of those discretized items. According to the research in speech production and neuroethology, human speech signals may be encoded in point patterns involving acoustic events in the speech signal and neural spikes in the brain [64, 15]. Such points in time are referred to as acoustic event landmarks in [65]. Based on this hypothesis, it may pose some challenges for RNN-HMM acoustic modeling: firstly, it arbitrarily discretizes the continuous input into the interval features at a fixed frame rate, which may introduce discretization errors and have a negative impact on the model performance accordingly; secondly, the occurrences of such acoustic events are unknown and such data are unavailable. On the other hand, the training targets of RNN-HMM are usually obtained from the recognition results of other (usually inferior) DBN models and misalignments are inevitable.

In this thesis, we develop a deep probabilistic model called recurrent Poisson process unit (RPPU) to deal with the aforementioned problems. The hybrid ASR system under the above hypothesis can be factored into three steps:

- Allocate the training acoustic events localized in time at the HMM state level to better align with the training targets.
- Reconstruct/sample a series of acoustic features from the interval features originally sampled at a fixed frame rate from the allocated acoustic events.
- Follow the traditional ASR processing procedure using the newly reconstructed acoustic features as additional inputs.

The first step is achieved by constructing a recurrent Poisson process (RPP), which consists of a collection of homogeneous Poisson processes [38] at a series of time intervals. In the proposed point process, the intensity function is determined by an RNN hidden state encoding the past history of the acoustic signal. Sampling from intensity-based models is usually performed via a thinning algorithm[49], which is indifferentiable and computationally expensive. Our method is sampling-free and it provides a solution in an analytical form which ensures computational efficiency. In the second step, the better aligned acoustic features are dynamically reconstructed through a linear interpolation in which the weight parameters are estimated from the acoustic events drawn from the RPP. Finally, those estimated acoustic features are provided to an RNN as additional input to perform the HMM state prediction in a traditional way.

The objective function of RPPU is designed to strike a balance between the generation of arrival times of the latent acoustic events for clean training data and encoding sufficient uncertainty to capture the variability caused by the discretization errors and misalignments. Notably, RPPU can be trained with the standard backpropagation through time (BPTT) [70]. The experiments on CHiME-2, WSJ0 and WSJ0&1 show that our new model consistently outperforms the conventional LSTM, SRU and quasi-RNNs.

1.3 Related Work

This section will introduce the related concepts and previous works that inspire our thesis. We mainly focused on recurrent neural networks (RNNs), temporal point process and acoustic event models in ASR.

1.3.1 Recurrent Neural Networks

Recurrent neural networks (RNNs) and its modern variant simple recurrent unit [40] are building blocks of our model. RNN can be a very powerful model for sequential signals. When unfolded in time, RNNs may be considered as a very deep neural network (DNN) [47]. Thus, in principle, it can model long-span temporal dependency in sequential signals through simple recurrent network connections. However, in practice, training RNNs using the back-propagation through time (BPTT) algorithm [70] can be difficult due to the well-known vanishing or exploding gradient problem [33]. One typical solution is to bridge the gap between an RNN hidden state and its preceding states by the addition of direct feedback paths to the latter so that the gradients can reach the preceding states more readily. Higher-order NARX RNN [42] is the earliest work that introduced this solution. LSTM attempts to mitigate this problem by using a set of self-learning gates to control the amount of short-term and long-term information to forget, to retain, and to further propagate. This make the decay's gradient base of neighbouring memory units close to one. However, the exponential shrinking of the spectral norm of the gradient component at each time step still cannot be avoided over the long range of time steps. In contrast, our RPPU is less prone to this problem. It learns a recurrent Poisson process in dynamically re-organizing (or shifting) the input sequence to be better aligned with the training targets such that the target may depends more on the current input and less on the input that are far apart.

Another limitation of RNNs is that computation units of RNNs are mostly deterministic. Therefore, it is hard to model the uncertainty and we only have access to point estimates of their parameters and predictions [24]. A variational RNN (VRNN) [10] is proposed to solve this by introducing a variational auto-encoder (VAE)[37] into an RNN architecture for every time step. In this model, each hidden state depends on the uncertainty information captured by VAE for every time step. Here, VAE intends to approximate variational posterior distribution represented by a Gaussian. With the learned latent distribution from VAE, the RNN models are shown much better generalization in generating a natural speech sequence and handwriting samples. However, a standard VRNN can only capture some unknown variabilities exhibiting non-interpretable representations. This is because the approximating distributions are assumed to take a general form: a Gaussian. On the other hand, our model is capable of modeling variabilities caused by the discretization errors and misalignments explicitly, which is achieved by adopting a more flexible recurrent Poisson proces.

1.3.2 Temporal Point Processes

Temporal point processes have been a principled framework for modeling phenomena on an event-by-event basis across a wide range of domains. It has originally been used for modeling earthquakes [29, 28] in seismology. More recently, in social network, a hawkes process has been used to model timing and rich features of social interactions [72]; in human activity modeling, Poisson Processes have been applied to model the inter-arrival time of human activities [17, 43]; and in neurobiology, a determinantal point process (DPP) latent variable model [63] was proposed to capture the interaction and correlation among neurons in neural spiking data. To do this, the DPP is imposed on the latent variables of all neurons to model both inhibitory and competitive interactions among them.

A major limitation of these existing work is that they often make strong assumptions about the generative processes of the event data, which may not be well-suited for real world problem. Further more, the fixed parametric form of the intensity function also have restricted the expressive power of the respective processes. Therefore, most of existing work focus on enhancing the flexibility of point process models, e.g. , a nonparametric Bayesian approach of point processes have been explored in [66], in which the intensity function of modulated renewal process is approximated using a Gaussian process; [44] extended the multivariate Hawkes process [28] to a neurally self-modulating multivariate point process using a continuous-time long short-term memory. Similarly, [13] proposed a model based on marked temporal point process that model the event timings and the markers with the help of a long short-term memory recurrent neural network (LSTM RNN). However, these mothods focus only on modeling the occurrences of events or associated markers, our proposed work try to develop a framework which is capable of making inferences about the underlying events even when event timings are not available during training.

1.3.3 Acoustic event models in ASR

An acoustic model based on marked Poisson process has been proposed for a sub-task of event-based ASR [36]. This subtask is an unusual ASR task in that speech signal is required to be segmented into sonorant and obstruent regions prior to acoustic modeling. Therefore, the occurrences of the acoustic events are provided during training and testing and the model parameters of intensity function is learned by simply using maximum likelihood estimation (MLE). Essentially, the marked Poisson process in their work is a phonetic classifier given the segment information. In contrast, in our work the annotation of the latent acoustic events in the acoustic speech signal is not available; hence direct supervised learning via MLE is not possible. Our model addresses this challenge by treating these latent acoustic events as latent variables, which are then used as part of the generative process that is linked to the training targets.

Similarly, a segment-based event detection framework using HMM was proposed in

[41] for automatic speech recognition, in which the event boundaries are obtained from forced alignment. The training of such HMM-based event detectors require the event labels for each training utterance. Additionally, it assumes uniform sampling, and thus the detected events can only occur at quantized time intervals, which is what RPPU attempts to overcome. Continuous-Time HMM alleviate this problem by introducing another set of latent variables that model the holding/inter-arrival times between any two states. However, it is not clear how to integrate deep learning with Continuous-Time HMM, while RPPU fits well with the current ASR framework using RNN.

1.4 Thesis Summary

In this thesis, we present a novel deep probabilistic model called the recurrent Poisson process unit (RPPU), to address hybrid acoustic modeling. The key idea of our approach is to view the arrivals of the latent acoustic events as several temporal latent variables. Each follows an associated sub-Poisson point process for a dynamic time interval, whose intensity function is parameterized by a recurrent neural network. More specifically, our work is summarized as follows:

- We describe the potential problems caused by the aforementioned hypothesis for the RNN-HMM acoustic model. To solve these problems, we reformulate the ASR problem and divide the hybrid ASR system into three steps as discussed in Section 1.2.
- We propose a novel recurrent Poisson process to jointly model the latent acoustic events and acoustic HMM states by treating the former as latent variables, which are then used as part of the generative process that is linked to the training targets: acoustic HMM states.
- We provide an analytical solution which ensures efficiently drawing samples from a recurrent Poisson process (RPP).
- We integrate the RPP into SRU and create a novel deep probabilistic model called recurrent Poisson process unit (RPPU). We design the respective learning method and all of the RPPU components including the sampler of RPP are differentiable,

so that our model can be trained with the standard backpropagation through time (BPTT) [70] and stochaistic gradient descent (SGD) [57].

- Despite the additional computation overhead caused by RPP, our RPPU runs almost as fast as SRU while having a similar number of parameters.
- We conduct both qualitative and quantitative analysis in speech recognition tasks to show that our model can generate much better alignments while performing the HMM state modeling.
- Our RPPU outperforms several baseline acoustic models including the conventional LSTM, SRU and quasi-RNNs on three speech recognition datasets: CHiME-2, WSJ0 and WSJ0&1.

1.5 Outline of the Thesis

In Chapter 2, we present an overview of automatic speech recognition (ASR) and state-of-the-art ASR systems together with details of RNN-HMMs that are used for acoustic modeling. We review the existing variants of RNN, and fundamental limitations of RNN for acoustic modeling are also described.

In chapter 3, we introduce and formulate the problem of the RNN-HMM acoustic model. We then present the theory of the recurrent Poisson process. Then, details about how the recurrent Poisson process can be integrated into RNN are presented. We finally describe the learning of the recurrent Poisson process unit with a well-designed learning objective which is augmented by an additional regularization term on RPP parameters.

In Chapter 4, we discuss the experimental evaluation of various RNN models for the ASR data corpus including ChiME-2, WSJ0 and WSJ0&1. The effectiveness of RPPU is investigated through carefully designed experiments. The conclusion and the future work are discussed in the last chapter.

CHAPTER 2

THE USE OF RECURRENT NEURAL NETWORKS FOR ACOUSTIC MODELING

2.1 Speech Production and Acoustic Speech Signal

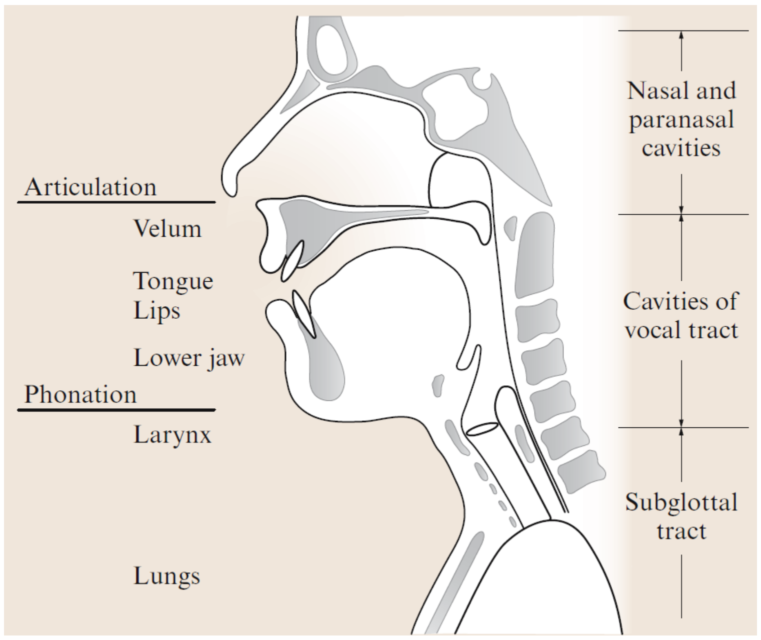

Figure 2.1. Sketch of a speech production system (from [5])

The physical production of speech sound is a complicated process including three functional units: generation of air pressure, regulation of vibration, and control of resonators [5]. First, it starts with airflow from the lung. Then driven by the air pressure in the lungs, voice source sounds are then generated during the vocal fold vibration at the larynx. Here its features such as pitch, volume and timbre are first adjusted. They are further modulated in various ways through movements generated by articulators such as tongue, lower jaw, lip and velum (Fig 2.1). These features help humans to distinguish most of the speech sounds based on how and when they are articulated. As such, the researchers in [64, 15] believe that human sound may be encoded in point patterns involving some articulatory events. Such points in time are referred to as acoustic event landmarks in [65].

In modern communication systems, though an acoustic speech signal is inherently analogue, it is encoded by a sequence of bits which take advantage of the fact that binary representation can be recovered robustly from a noisy channel. Generally, the analogue microphone output is sampled and quantized to a binary representation by an analogue-to-digital (A/D) converter. Common sampling rates are 8 or 16 kHz for narrowband speech and wideband speech, respectively. The quantization precision is usually 16 bits per sample.

2.2 Basic Components in HMM-based Speech Recognition

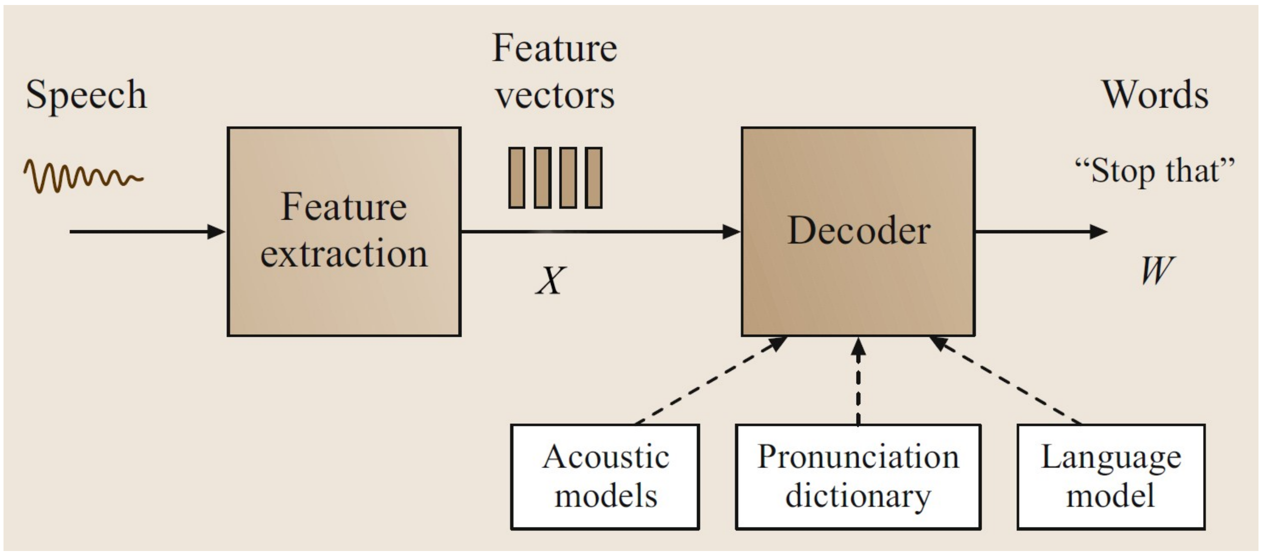

Figure 2.2. Architecture of an HMM-based recognizer (modified from [5])

The principal components of a statistical automatic speech recognition system are illustrated in Fig 2.2. The input audio waveform from a microphone is converted into a sequence of fixed-size acoustic vectors $\mathbf{X} = \mathbf{x}_1, ..., \mathbf{x}_T$ through a component called *feature extraction*. Feature extraction attempts to provide a compact representation of the speech waveform. The output feature vectors are computed with a 20 ms sliding window with a 10 ms shift. One of the simplest and most widely used acoustic features is filter bank energies [48].

The *decoder* aims to search for a word sequence that is most likely to match $\mathbf{X}$. More specifically, it achieves this by maximizing the probability $P(W|\mathbf{X})$. However, since this probability is not directly tractable, it can be factorized into a product of $P(\mathbf{X}|W)$ and $P(W)$ using the Bayes' Rule. $P(W)$ can be computed using a *language model* (LM) [55], which estimates the probability of the word sequence W using the production of items in an N-gram model in which the probability of each word is conditioned only on its N-1 predecessors; $P(\mathbf{X}|W)$ can be computed using an *acoustic model* e.g. hidden Markov model (HMM) [53]; it describes the statistics of the class conditional observation sequences given the corresponding word sequences.

The smallest unit of sound that can be represented by the acoustic model is a *phoneme*. For example, the word cat is composed of three phones /c/ /ae/ /t/. There are about 40 such phonemes in a typical ASR system and there is a *pronunciation dictionary* to define any words involved in W.

2.3 HMM and its Usage as Acoustic Models

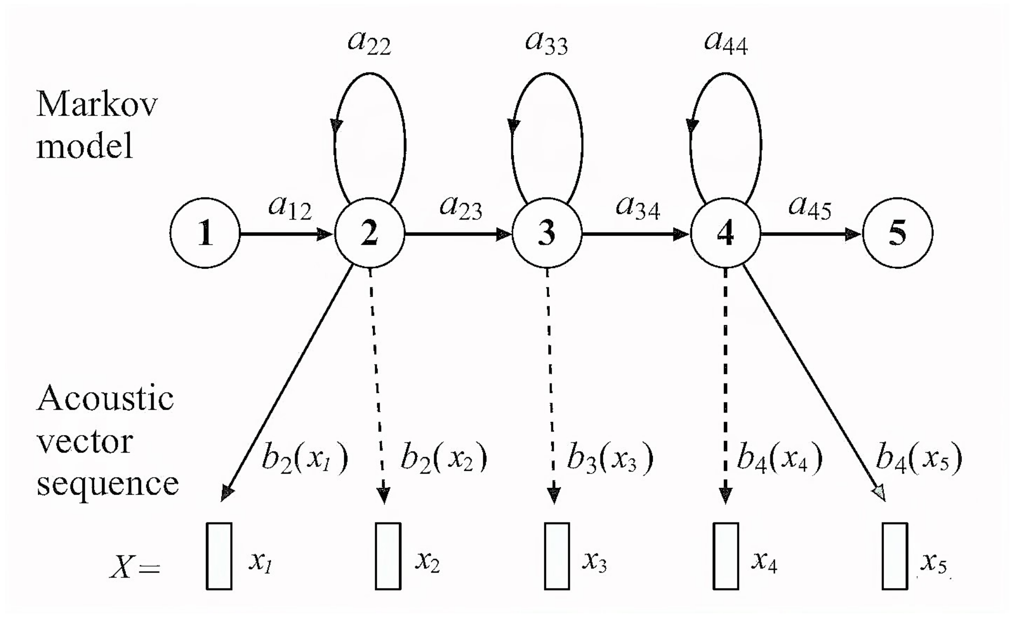

Figure 2.3. HMM-based phone model

The Hidden Markov model (HMM) is a probabilistic finite-state machine, in which internal states cannot be directly observed but can be inferred by observing some outputs generated by each state . It was first introduced in [4] , and has been successfully applied to model time-series data in fields such as speech recognition, speech synthesis and handwriting.

HMM consists of three components: HMM states, state transitions and outputs. Each HMM state has a discrete or continuous probability distribution over possible outputs or emissions. State transitions are governed by a set of probabilities called transition probabilities. In a typical first-order HMM, a transition probability depends solely on the previous state and not on the past state history states. An output is emitted when the system visits a particular state or transits from one state to another. The distribution of emitting such an output is modelled as a function of the emitting state. This is achieved by making the output independence assumption, in which given its emitting state the emitted output is conditionally independent of the previous emitted output and the neighboring states.

In speech recognition, the acoustic model comprised of a set of HMMs is used to model the acoustic feature sequence. Each HMM represents a base phone, which is illustrated in Fig 2.3 with transition probability parameters a_{ij} and continuous output probability density functions $b_j(*)$. The latter are modelled using mixtures of Gaussians [53] in an old GMM-HMM ASR system [34], yet they are computed using the outputs of a recurrent neural network in the new RNN-HMM ASR system [26], as discussed in more detail in Section 3.1. The phone acoustic model shown in the Fig 2.3 is a 3-state straightly left-to-right HMM with the non-emitting entry and exit states designed to simplify the process of concatenating phone models to make word models.

In a typical GMM-HMM ASR system, the model parameters a_{ij} and $b_j(*)$ can be efficiently estimated from a corpus of training utterances using expectation maximization (EM) [3]. Initially, the global mean and covariance of the data are assigned to all Gaussian components and all transition probabilities are set to be equal. In the E step, a forward-backward alignment is used to compute state occupation probabilities and the means and covariances are then estimated via simple weighted averages in the M step. The E and M steps are then repeated until the estimate for model parameters stop changing.

As noted in section 2.2, the most likely word sequence is found by searching all possible state sequences derived from all possible word sequences that were most likely to have generated the observed data. An efficient way to solve this problem is to use the Viterbi algorithm [18]. More specifically, in large-vocabulary continuous speech recognition tasks (LVCSR), the basic Viterbi algorithm will be recast for LVCSR as the token passing algorithm and the associated HMM topology is built by constructing a recognition network which consists of all vocabulary words in parallel in a loop.

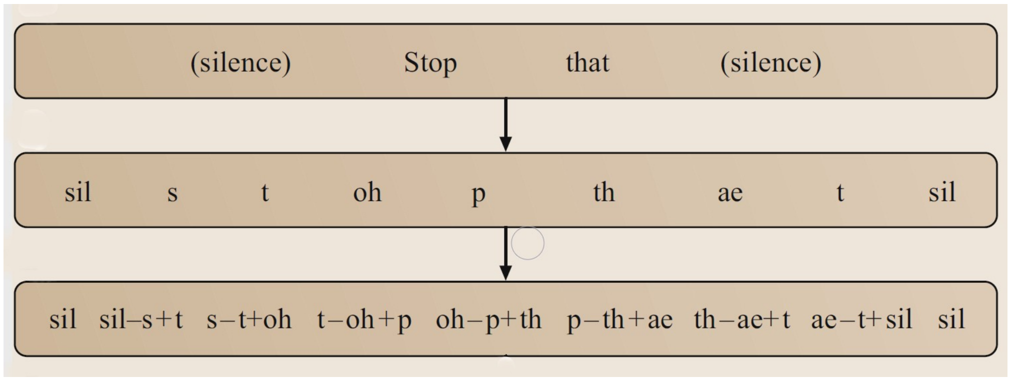

Figure 2.4. Context-dependent phone modeling ((modified from [5]))

In phone-based modelling, context-independent (CI) units are also called monophones. Each phone is modelled by a single HMM. It is assumed that each phone is acoustically independent from their neighboring phones in an utterance. This assumption obviously violates the fact that neighboring units do affect each other (the co-articulatory effects) and therefore the performance of using CI units is modest. A simple way to mitigate this problem is to use a unique phone model for every possible pair of left and right neighbors. The resulting models are called triphones, which are illustrated in Fig 2.4. Notice that the number of total units may result in the data sparsity problem. To avoid this, the complete set of logical triphones can be mapped to a reduced set of physical models by clustering and tying together the parameters in each cluster [62].

2.4 Recurrent Neural Networks

Recurrent neural networks are neural networks with self connections, which allow internal representations to be generated and interaction of the observations to be captured at different times [16]. They have been applied to a wide variety of machine learning problems: automatic speech recognition (ASR) [26], language modeling [46], machine translation [2], computer vision [68], and so forth.

A recurrent neural network (RNN) is essentially a type of nonlinear dynamic system comprised of two components: hidden states and transitions. The hidden state is denoted by a multivariate vector of variables associated with discrete time. Its value summarizes all the information about the historical behavior of the system, and in the meantime takes account of describing its future behavior. The transition is assumed to be deterministic, which determines how the hidden state evolves through time. Typically, in a sequence of training examples $[(\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2), ..., (\mathbf{x}_m, \mathbf{y}_m)]$ with $\mathbf{x}_t \in \mathbb{R}^n$, $\mathbf{y}_t \in \mathbb{R}^k$, for $1 \leq t \leq T$, the vectors $\mathbf{x}_t$ are given as inputs to the network, while the vectors $\mathbf{y}_t$ denote the ground truth represented by a one-hot vector with k classes. The dynamic system in RNN can be represented by the following two equations:

$$\mathbf{h}_t = \sigma(\mathbf{W}_x \mathbf{x}_t + \mathbf{W}_h \mathbf{h}_{t-1} + \mathbf{b}_h) \tag{2.1}$$

$$\hat{\mathbf{y}} = softmax(\mathbf{W}_y \mathbf{h}_t + \mathbf{b}_y) \tag{2.2}$$

where $\mathbf{h}_t \in \mathbb{R}^r$ describes the hidden state. The first equation defines the state transition mapping, in which the hidden state $\mathbf{h}_t$ is a nonlinear function of the current data point $\mathbf{x}_t$ and the previous hidden state $\mathbf{h}_{t-1}$. The state-to-state transition matrix $\mathbf{W}_h$ and input-to-state transition matrix $\mathbf{W}_x$ allows information flow over input $\mathbf{x}_t$, hidden states $\mathbf{h}_t$ and $\mathbf{h}_{t-1}$. The output mapping usually adopts the softmax function to calculate the predictions $\hat{\mathbf{y}}$. $\mathbf{b}_h$ and $\mathbf{b}_y$ are the biases of the hidden layer and output layer respectively.

The dynamics of the unfolded RNN depicted in Figure 2.5 is similar to a feedforward neural network .

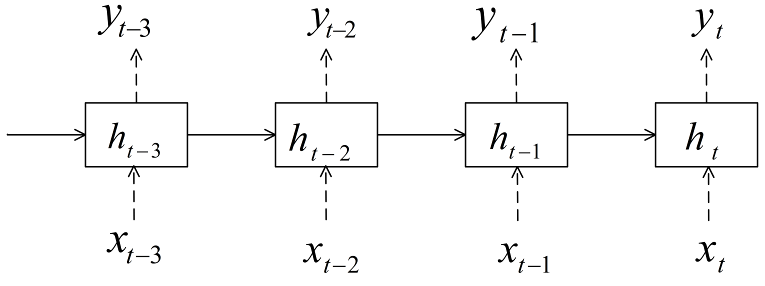

Figure 2.5. The unfolded Simple RNN.

It is natural to adopt backpropagation [30] to train the parameters of the weight matrices and biases of an RNN given the objective function. This algorithm is the so-called backpropagation through time (BPTT) [70]. It is known to perform poorly for the problem associated with long term dependency. One of most successful solutions is described in next subsection.

2.4.1 Long Short-term Memory Recurrent Neural Networks

For the problem which involves long term dependency, that is, the problem in which the desired outputs depend on inputs far in the past, BPTT for a standard RNN does not work very well because of the vanishing and exploding gradient problem [32].

Long short-term memory(LSTM) [33] is a special implementation of a recurrent neural network (RNN) that is easy to train and partially addresses this drawback. LSTM maintains different types of gates to dynamically and recursively distill inputs into memory cells. At each time step, with the observation $\mathbf{x}_t$ fed to LSTM, the gates, hidden states $\mathbf{h}_t$ and the memory cells $\mathbf{c}_t$ are calculated:

$$\left[\hat{\mathbf{i}}_t, \hat{\mathbf{f}}_t, \hat{\mathbf{o}}_t\right] = \mathbf{W}_x\mathbf{x}_t + \mathbf{W}_h\mathbf{h}_{t-1} + \mathbf{w}_c \odot \mathbf{c}_{t-1} + \mathbf{b} \tag{2.3}$$

$$\hat{\mathbf{c}}_t = \mathbf{W}_{xc}\mathbf{x}_t + \mathbf{W}_{hc}\mathbf{h}_{t-1} + \mathbf{b}_c \tag{2.4}$$

$$\mathbf{i}_t = \sigma(\hat{\mathbf{i}}_t) \tag{2.5}$$

$$\mathbf{f}_t = \sigma(\hat{\mathbf{f}}_t) \tag{2.6}$$

$$\mathbf{o}_t = \sigma(\hat{\mathbf{o}}_t) \tag{2.7}$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot tanh(\hat{\mathbf{c}}_t) \tag{2.8}$$

$$\mathbf{h}_t = \mathbf{o}_t \odot tanh(\mathbf{c}_t) \tag{2.9}$$

where $\mathbf{w}_c$ is the weight vector of peephole connection; $\mathbf{W}'s$ are weight matrices; $\mathbf{b}'s$ are bias vectors; $\odot$ is element-wise multiplication operation; $\mathbf{i}_t$, $\mathbf{f}_t$, $\mathbf{o}_t$ are the input gate, forget gate and output gate repectively. The memory cell $\mathbf{c}_t$ learns to make decisions about what to store, and when to allow reads, writes, and forgets, via gates. All the gates are implemented by a sigmoidal function σ, which takes activation from the current observation $\mathbf{x}_t$ as well as from the previous hidden state $\mathbf{h}_{t-1}$. The extent to which the new memory is combined is controlled by the input gate. If the value of the input gate is zero, then the information flow from the observation is cut. If the value of the input gate is one, the information involved in the observation is pumped into the memory cells $\mathbf{c}_t$. Similarly, the extent to which the existing memory is flushed is controlled by the forget gate. The memory cell $\mathbf{c}_t$ could be flushed with the new memory if the forget gate $\mathbf{f}_t$ is turned on. Whether the current cell output $\mathbf{c}_t$ will be propagated to the final output $\mathbf{h}_t$ is further controlled by the output gate $\mathbf{o}_t$. For some timing tasks where the network must learn to measure precise intervals between events, it has been observed that adding a peephole connection that passes from the internal state directly to the gates can improve its performance.

When performing the backpropagation process, the error going through the memory cells is less likely to explode or vanish across many time steps. In the original LSTM, the memory cell is self-connected with a unit weight, and the update of the memory cell $\mathbf{c}_t$ is given by:

$$\mathbf{c}_t = \mathbf{c}_{t-1} + \mathbf{i}_t \odot g(\hat{\mathbf{c}}_t) \tag{2.10}$$

where g is a nonlinear function which squashes $\hat{\mathbf{c}}_t$. Different from the equation defined in equation 2.8, there is no forget gate between $\mathbf{c}_t$ and $\mathbf{c}_{t-1}$. In this case, the error can flow across time steps without vanishing or exploding, which is known as the constant error carousel [33]. Although the forget gate appears to violate the motivation of the constant error carousel, they have proven effective in most modern implementations, for instance, automatic speech recognition [26].

One extension of LSTM is the bidirectional LSTM (BLSTM) [61], which is composed of one forward LSTM and one backward LSTM. The forward one reads the input sequence in the forward direction and the backward one reads the input sequence in a backward order. The final outputs are generated by concatenating both outputs from these two LSTMs. One advantage of BLSTM is that this process can provide temporal dependency in two different directions. However, this model requires an endpoint in both the future and in the past so that some online applications need to delay until the whole sequence has been observed.

2.4.2 Gated Recurrent Units

Another way to deal with the issue of vanishing and exploding gradients was proposed by Kyunghyun and Benjio by the introduction of gated recurrent units (GRU) [9]. Its gate control mechanism is a simplified version of LSTM: the forget gate and input gate are combined into a single reset gate and the output gate is removed. Different from LSTM, gated recurrent units modulate the information without using a memory cell. The dynamic of the hidden state is implemented by a linear interpolation between the previous hidden state $\mathbf{h}_{t-1}$ and the candidate hidden state $\tilde{\mathbf{h}}_{t-1}$:

$$\mathbf{h}_t = \mathbf{z}_t \odot \mathbf{h}_{t-1} + (1 - \mathbf{z}_t) \odot \tilde{\mathbf{h}}_t \tag{2.11}$$

where the update gate $\mathbf{z}_t$ in the formula controls the information flow from the previous hidden state and the candidate hidden state. $\odot$ denotes the element-wise multiplication operation. The $\mathbf{z}_t$ is defined as a sigmoid function σ of the input $\mathbf{x}_t$ and the previous hidden state $\mathbf{h}_{t-1}$:

$$\mathbf{z}_t = \sigma(\mathbf{W}_z \mathbf{x}_t + \mathbf{U}_z \mathbf{h}_{t-1}) \tag{2.12}$$

where $\mathbf{W}_z$ and $\mathbf{U}_z$ are weight matrices. The activation of the candidate hidden state is controlled by the reset gate $\mathbf{r}_t$:

$$\tilde{\mathbf{h}}_t = \tanh(\mathbf{W}\mathbf{x}_t + \mathbf{U}(\mathbf{r}_t \odot \mathbf{h}_{t-1})) \tag{2.13}$$

where tanh denotes the hyperbolic tangent function. $\mathbf{W}$ and $\mathbf{U}$ are weight matrices. When $\mathbf{r}_t$ is turned on, the information both from the input and the previous hidden state will be combined. When it is turned off , the unit tends to focus on the input and ignore its history. The reset gate $\mathbf{r}_t$ is written as:

$$\mathbf{r}_t = \sigma(\mathbf{W}_r \mathbf{x}_t + \mathbf{U}_r \mathbf{h}_{t-1}) \tag{2.14}$$

where $\mathbf{W}_r$ and $\mathbf{U}_r$ are weight matrices.The reset gate $\mathbf{r}_t$ is computed similarly to the update gate, which determines the importance of the historical memory of the candidate hidden state.

Overall, GRU has fewer free parameters than LSTM, thus may require less data for training. It also has the capability of overcoming the vanishing or exploding gradient problem. It has been observed that, by fixing the model size, GRU outperforms LSTM in terms of convergence speed and generalization in some tasks [9].

2.4.3 Simple Recurrent Unit

LSTM runs slowly with the computation bottleneck at the gates. Simple recurrent unit (SRU) [40] simplifies the architecture of LSTM and increases its running speed by dropping the connections between its hidden states and gates so that gates computations can be done in parallel. In this thesis, we use it as our basic building block. Below are the updating formulas of SRU which runs with n histories.

$$\left[\hat{\mathbf{r}}_t, \hat{\mathbf{f}}_t, \hat{\mathbf{c}}_t\right] = \mathbf{w}_x * \mathbf{x}_t + \mathbf{b} \tag{2.15}$$

$$\mathbf{r}_t = \sigma(\hat{\mathbf{r}}_t) \tag{2.16}$$

$$\mathbf{f}_t = \sigma(\hat{\mathbf{f}}_t) \tag{2.17}$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + (1 - \mathbf{f}_t) \odot \hat{\mathbf{c}}_t \tag{2.18}$$

$$\mathbf{h}_t = \mathbf{r}_t \odot tanh(\mathbf{c}_t) + (1 - \mathbf{r}_t) \odot \mathbf{x}_t \tag{2.19}$$

where $\mathbf{r}_t$ is the reset gate outputs; $\mathbf{f}_t$ is the forget gate outputs; $\mathbf{c}_t$ os the memory cell outputs; $\mathbf{w}_x$ and $\mathbf{w}_\mathbf{y}$ are the weight matrices; $\mathbf{b}$ is the gate bias vector; $\mathbf{h}_t$ is the hidden state outputs; any quantity with a ˆis the activation value of the quantity before an activation function is applied; $\odot$ is the element-wise multiplication operation; σ is the logistic sigmoid function.

2.5 RNN-HMM ASR System

2.5.1 Training

As a typical dynamic system, RNN is a very natural way of acoustic modeling because speech is essentially a dynamic process. Graves [25] have successfully applied BLSTM into an ASR hybrid system, in which BLSTM was trained to approximate the posterior probabilities of the context-dependent HMM states. In this sense, BLSTM can be viewed as the "state classifier" optimized based on a negative log-likelihood:

$$-\log P(\mathbf{S}|\mathbf{X}) = -\sum_{t=1}^{T} \mathbf{s}_t \log \mathbf{y}_t \tag{2.20}$$

where the length of the input sequence $\mathbf{X}$, the output sequence of BLSTM and the target context-dependent HMM state sequence $\mathbf{S}$ are T; $\mathbf{y}_t$ is referred to as the probability distributions over K possible context-dependent HMM state labels at time t; $\mathbf{s}_t$ is the target context-dependent HMM state label at time t. It is usually generated by forcefully aligning the training set with a GMM-HMM.

The parameters of BLSTM are updated using truncated BPTT [73]. In this algorithm, the RNN is unfolded to a fixed number of time steps (e.g. 20) and the training mini-batch usually contains a short segment. The error signal at the output layer can be written as:

$$-\frac{\partial logP(\mathbf{S}|\mathbf{X})}{\partial \tilde{\mathbf{y}}_t^k} = \mathbf{y}_t^k - \mathbf{s}_t^k \tag{2.21}$$

where $\tilde{\mathbf{y}}_t^k$ and $\mathbf{y}_t^k$ are the activation before the softmax and the prediction at the k^{th} entry of the output layer. $\mathbf{s}_t^k$ is the k^{th} entry of one-hot vector for the ground truth at time t. The weight gradient attached to the parameters of the unfolded layers are then determined based on this error signal separately and finally added together.

To avoid overfitting and improve convergence, it has been found advantageous to perform the gradient clipping [51], batch normalization [35] and adding Gaussian noise to the network weights [26].

2.5.2 Decoding

The ultimate goal of the hybrid system is to generate the most likely words or phoneme sequence. This is done by a decoder, where the probability of any sequence $\mathbf{W}$ given an acoustic observation sequence $\mathbf{X}$ is maximized. It can be written as:

$$p(\mathbf{W},\mathbf{X}) \approx \max_{\mathbf{S}} p(\mathbf{W},\mathbf{S})p(\mathbf{X}|\mathbf{S}) \tag{2.22}$$

where $\mathbf{S}$ is the context-dependent HMM state sequence; $p(\mathbf{W},\mathbf{S})$ is the language model score. $p(\mathbf{X}|\mathbf{S})$ is the acoustic likelihood. The acoustic likelihood cannot be obtained directly from the output of a BLSTM. Given the feature frame $\mathbf{x}_t$ at time t, the acoustic likelihood of the context-dependent HMM state $\mathbf{s}_t$ can be calculated from the BLSTM output with the Bayes rule:

$$p(\mathbf{x}_t|\mathbf{s}_t) = \frac{p(\mathbf{s}_t|\mathbf{x}_t)p(\mathbf{x}_t)}{p(\mathbf{s}_t)} \tag{2.23}$$

where $p(\mathbf{s}_t|\mathbf{x}_t)$ is the output of the LSTM; $p(\mathbf{s}_t)$ is the prior probability for the HMM state $\mathbf{s}_t$; $p(\mathbf{x}_t)$ can be ignored. Although this conversion appears to have little effect on some recognition tasks, it matters for some tasks where training labels are highly unbalanced.

2.5.3 Evaluation

We evaluate a hybrid LSTM RNN-HMM speech recognition system on the TIMIT corpus [21].

The TIMIT speech corpus

The standard NIST training set which consists of 3,696 utterances from 462 speakers was used to train the various DNN and RNN models. A separate development data set, consisting 400 utterances from 50 speakers, was used for early stopping, and the standard core test set, consisting of 192 utterances spoken by 24 speakers, was used for evaluation. We followed the standard TIMIT protocol and collapsed the original 61 phonetic labels in the corpus into the standard set of 39 phonemes for reporting the recognition performance in terms of phoneme error rate (PER). Phoneme recognition was performed using Viterbi decoding with a phone bigram language model estimated from the TIMIT training transcriptions using the Kaldi toolkit.

Feature extraction and model training procedure

Acoustic hidden Markov models (HMM) based on Gaussian-mixture model (GMM), deep neural network (DNN), LSTM-RNN, HO-LSTM RNN, and MH-LSTM RNN were built. GMM models employed fMLLR-adapted 39-dimensional MFCC features, while all neural-network based models used 40 mel-filterbank coefficients without their derivatives. Inputs to DNN/RNNs were normalized to have zero mean and unit variance.

Table 2.1. Summary of TIMIT phoneme recognition performance. F = input context in number of frames; L = number of hidden layers; N = number of nodes per hidden layer.

Model	F	L	N	PER %
DNN	11	4	1024	21.8
LSTM	1	3	512	21.0
LSTM	5	3	512	20.6

The GMM-HMM was trained using the standard Kaldi TIMIT recipe and there were 1940 tied context-dependent states. It was then used to derive the state targets for subsequent DNN/RNN training through forced alignment. All DNN/RNN models were trained by our own codes developed using Theano. Inputs of DNN consisted of the current frame together with its 5 left and 5 right contextual frames. The number of hidden layers, the number of hidden nodes per layer for LSTM were varied from 2–4, 128–1024 respectively to find the best DNN/RNN architecture for the task. Both DNN and RNNs were trained by optimizing the target cross entropies, using BP and BPTT respectively and SGD.

Results

Table 2.1 shows the TIMIT phoneme recognition performance of the baseline DNN, LSTM RNN. It can be seen that LSTM performs much better than DNN by 0.8% or 1.2% absolute when inputs of 1 or 5 contextual frames are used. Since more contexts give better LSTM performance, all ensuing LSTM experiments used in the reminder of the thesis employed an input context of 5 frames.

2.6 Limitations of RNN for Acoustic Modeling

There are some limitations for RNN based acoustic models:

Limited length of history: Although in theory the length of contextual or historical frames that can be learned in RNN is infinite. In practice, the range of context or history that is effectively learned is quite limited. This is due to two reasons. Firstly, RNN may suffer from the vanishing gradient problem [32]. Secondly, the memory units of LSTM have limited memory capacity [22], so it has a hard time accommodating very long term temporal dependency.

Uncertainty handling: Situations that can lead to uncertainty in automatic speech recognition include environmental noise, speaker, uncertainty in model parameters and so forth. Uncertainty modeling is of great importance in this task—with uncertainty information we can obtain models that generalize well. However, the uncertainty or randomness is hard to obtain for the standard RNN. This may be because the transition of the standard RNN is entirely deterministic. The only source of randomness is from the output probability distribution. This may be an inappropriate way to model the kind of variability observed in spoken speech signal, characterized by strong and complex dependencies among the acoustic events at different time.

CHAPTER 3

ACOUSTIC MODELLING USING RECURRENT POISSON PROCESS UNIT

3.1 Introduction

An ASR hybrid RNN-HMM system [25] consists of a recurrent neural network estimating a posterior probabilities for HMM states of the context-dependent phone conditioned on the acoustic input. Typically, in a sequence of training examples $[(\mathbf{x}_{t_1}, \mathbf{y}_{t_1}), (\mathbf{x}_{t_2}, \mathbf{y}_{t_2}), ..., (\mathbf{x}_{t_m}, \mathbf{y}_{t_m})]$ with $\mathbf{x}_{t_i} \in \mathbb{R}^n$, $\mathbf{y}_{t_i} \in \mathbb{R}^k$, for $1 \leq i \leq m$, the acoustic input $\mathbf{x}_{t_i}$ are given as inputs to the network, while the vectors $\mathbf{y}_{t_i}$ denote the ground truth represented by a one-hot vector with K context-dependent HMM states. It can be represented by the following two equations:

$$\mathbf{h}_{t_i} = g_\theta(\mathbf{x}_{t_i}, \mathbf{h}_{t_{i-1}}) \tag{3.1}$$

$$\hat{\mathbf{y}}_{t_i} = softmax(\mathbf{W}_y \mathbf{h}_{t_i} + \mathbf{b}_y) \tag{3.2}$$

where $\mathbf{h}_{t_i} \in \mathbb{R}^r$ describes the hidden state. The first equation defines the state transition mapping, in which the hidden state $\mathbf{h}_{t_i}$ is a nonlinear function of the current data point $\mathbf{x}_{t_i}$ and the previous hidden state $\mathbf{h}_{t_{i-1}}$ and θ is the parameter set of g. $\mathbf{W}_y$ and $\mathbf{b}_y$ is the weight matrix and the bias of the output layer respectively. The output mapping usually adopts the softmax function to calculate the predictions $\hat{\mathbf{y}}_{t_i}$. In this sense, the RNN can be viewed as the "state classifier" optimized based on a negative log-likelihood or cross-entropy:

$$-\log P(\mathbf{Y}|\mathbf{X}) = -\sum_{i=1}^{M} \sum_{k=1}^{K} \mathbf{y}_{t_i,k} \log \hat{\mathbf{y}}_{t_i,k} \tag{3.3}$$

where both the length of the input sequence $\mathbf{X}$ and the target context-dependent HMM state sequence $\mathbf{Y}$ are M; the target context-dependent HMM state label $\mathbf{s}_{t_i}$ is usually generated by forcefully aligning the training set with an inferior acoustic model such as a GMM-HMM. The ultimate goal of the hybrid system is to generate the most likely words or phoneme sequence. This is done by running a viterbi algorithm [18] within the HMM framework. It requires the acoustic likelihood $P(\mathbf{X}|\mathbf{Y})$, which can be calculated from the RNN output with the Bayes rule:

$$p(\mathbf{x}_{t_i}|\mathbf{y}_{t_i,k}) = \frac{p(\mathbf{y}_{t_i,k}|\mathbf{x}_{t_i})p(\mathbf{x}_{t_i})}{p(\mathbf{y}_{t_i,k})} \tag{3.4}$$

where $p(\mathbf{y}_{t_i,k}|\mathbf{x}_{t_i})$ is the output of the RNN; $p(\mathbf{y}_{t_i,k})$ is the prior probability for the k-th HMM state, which is usually estimated with the frequencies of the HMM states in the training samples; $p(\mathbf{x}_{t_i})$ can be ignored since $\mathbf{x}_{t_i}$ is observed.

One major limitation of hybrid acoustic modelling is that training targets are generated from a family of dynamic Bayesian network models, e.g. GMM-HMM and RNN-HMM, in which the misalignment arises inevitably and the arbitrary discretisation of the continuous acoustic signal should result in more of errors in the alignments. Nonetheless, the capability of handling such uncertainty only comes from the conditional output probability density given the deterministic transition function of a standard RNN. To effectively deal with this issue, the acoustic RNN model must be capable of approximating the arrival time of each training target and reconstructing/sampling the acoustic features dynamically based on the estimated arrival time, which is the main focus of our model.

3.2 Problem Formulation

Let us consider a time interval $[0, t_N]$, where time is discretized into N frames of duration 10ms. Given a sequence of acoustic features $\mathbf{X} = \{\mathbf{x}_{t_1}, \mathbf{x}_{t_2}, ..., \mathbf{x}_{t_N}\}$, our goal is to approximate a sequence of arrival times $\widetilde{L} = \{\tilde{t}_1, \tilde{t}_2, \ldots, \tilde{t}_N\}$ so that a new sequence of acoustic features $\widetilde{\mathbf{X}} = \{\tilde{\mathbf{x}}_{\tilde{t}_1}, \tilde{\mathbf{x}}_{\tilde{t}_2}, ..., \tilde{\mathbf{x}}_{\tilde{t}_N}\}$ can be estimated which should align better with the given targets $Y = \{\mathbf{y}_{t_1}, \mathbf{y}_{t_2}, ..., \mathbf{y}_{t_N}\}$ in terms of resolution and precision, and a more robust acoustic model can be learned from these newly estimated training samples.

3.3 Temporal Point Process

A temporal point process is a random process whose realization consists of a sequence of isolated events with their time-stamps. It is a mathematically simple model which can

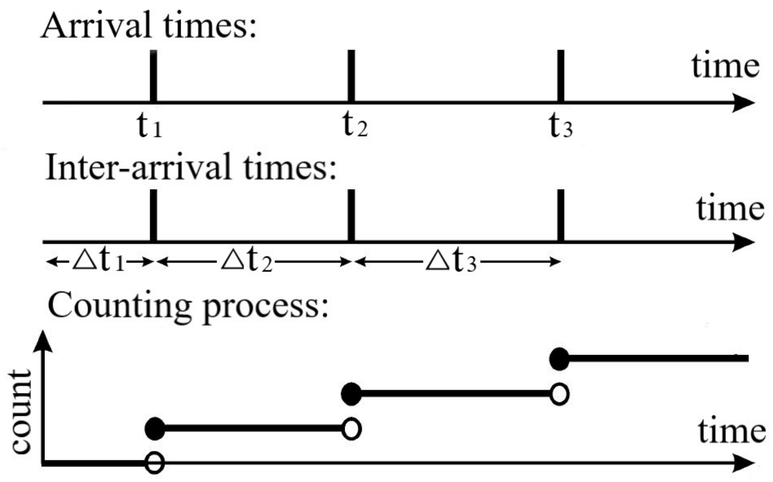

Figure 3.1. Multiple specifications for temporal point process

only produce binary labels that indicate whether an event occurs at that time. Due to their generality, point processes have been widely used for modeling phenomena such as earthquakes [28], human activities [43], financial data [1], context-aware recommendations [14] , etc. In speech processing, many different types of data can be represented as temporal point processes, such as the presence of voice activity in audio signals and onset or offset of various sound events.

A temporal point process can be equivalently represented as multiple collections of random variables: arrival times, inter-arrival times and counting process, which is illustrated in Fig 3.1. Let $t_1, t_2, \ldots, t_N$ be a sequence of arrival times of some event data and $\Delta t_1, \Delta t_2, \ldots, \Delta t_N$ be a sequence of inter-arrival times of some event data. We can compute the inter-arrival times from the arrival times by taking the difference between subsequent arrival times. Similarly, we can compute the arrival times by taking the cumulative sum of all the arrival times. A temporal point process can also be specified as a counting process, $N(t)$, which records the number of events before time t. Clearly, there is a one-to-one correspondence between any set of arrival times and any set of times at which the counting process increases. Although these random variables take different forms in terms of probability distribution, we can use these different random variables to describe the same point process data. For example, the set of arrival times $\{t : t_i > t\}$ when the i-th event has not yet occurred up to time t is equivalent to the set of times $\{t : N(t) < i\}$ when the count of events is less than i. Therefore, specifying any one of these random variables completely specifies the other two and determines the point process as a whole.

A conditional intensity function $\lambda(t)$ is defined to be the expected infinitesimal rate at which events are expected to occur around time t. It is an important way to characterize a temporal point process and it is usually a stochastic model for the arrival time of the next event given its history. Of all the variants of temporal point process, the homogeneous Poisson process has the simplest form of the intensity function which is a constant and assumed to be history-independent. This model is easy to implement and understand. However, due to the history independence assumption, most real world data is not well described as a homogeneous Poisson process. Therefore, to capture the dynamics of many real world event data, several functional forms of the intensity function are designed. For example, the Hawkes process is used to forecasting earthquakes in [50]. The intensity function of this model is parameterised to be history-dependent as earthquakes are well known to increase the risk of aftershocks.

It is important to be able to generate new arrival times from point process models because new arrivals can help us make inferences about the underlying events. Sampling from a homogeneous Poisson process is straight-forward. Since the distributions of the inter-arrival times are independent, the arrival time can be calculated by the sum of independent random samples generated according to the given exponential distribution. However, it is not straight-forward to simulate a general point process, which is usually performed via a thinning algorithm[49]. This algorithm follows a two-stage process. In the first stage, a set of candidate arrivals is generated as a homogeneous Poisson process with a rate being the upper bound of the intensity function of this point process. The next stage involves thinning out these candidate arrivals by stochastically rejecting some arrivals and accepting the rest. These stages require sampling from some random variables and are therefore indifferentiable. In general, this method is computationally expensive, because after generation of each sample, the intensity and its upper bound need to be re-evaluated..

Parameter learning for temporal point process models is the process to find a reasonable parameter set to fit a model to Event observations. Generally, it can be conducted by any convex optimization methods, e.g. maximum likelihood estimation (MLE), as log-likelihood is concave. In this work, the annotation of the latent acoustic events in the acoustic speech signal is not available, therefore it is not possible to cast supervised learning via MLE. To overcome this issue, we designed a Kullback–Leibler divergence-based objective and our RPPU, as a fully differentiable model, can be trained with the standard backpropagation through time (BPTT) [70] and stochaistic gradient descent (SGD) [57].

3.4 Poisson Point Process

A Poisson process is a special type of temporal point process defined in continuous time, in which the inter-arrival times are drawn i.i.d from an exponential distribution. It has a strong renewal property that the process can probabilistically restart at each arrival time, independently of the past. This enables us to describe the probabilistic behavior of the process via the intensity function $\lambda(t)$, which is a non-negative function. Within a small interval $[t, t + dt]$, the probability of an arrival is $\lambda(t)dt$.

By considering a sequence of arrival times of acoustic landmarks $L = \{t_1, t_2, \dots, t_N\}$ sampled from a Poisson process $\mathfrak{P}$ over an interval $[0, T]$, we have:

$$L \sim \mathfrak{P}(g(\lambda(t)) \tag{3.5}$$

$$\Delta t_i = t_i - t_{i-1} \sim g(\lambda(t)) \tag{3.6}$$

where g is the exponential density function; Δt_i is the inter-arrival time. We place the first landmark at time 0 for simplicity, and thus $t_0 = 0$. Given the observation of the previous landmark at time t_{i-1}, the probability that no landmark occurs up to time t since t_{i-1} is $\mathbb{P}(t_i > t) = e^{\int_{t_{i-1}}^{t} -\lambda(t)dt}$. Then, the probability that the first landmark lies in the interval $[t_i, t_i + dt]$ since t_{i-1} is computed as the product of $\mathbb{P}(t_i > t)$ and $\lambda(t_i)dt$, leading to the corresponding density function:

$$f_i(t) = \lambda(t)e^{\int_{t_{i-1}}^{t} -\lambda(t)dt} \tag{3.7}$$

which is actually the reverse relation of the hazard function for all intensity-based point processes [56]. By the strong renewal property, the likelihood of the whole arrivals L over an interval $[0, T]$ takes the form:

$$\begin{aligned} P(L|\lambda(t)) &= \mathbb{P}(t_{N+1} > T) \times \prod_{i=1}^{N} f_i(t_i) \\ &= e^{\int_0^T -\lambda(t)dt} \prod_{n=1}^{N} \lambda(t_n) \end{aligned} \tag{3.8}$$

where $\mathbb{P}(t_{N+1} > T)$ is the probability that no landmark is observed in the interval $(t_N, T]$. It may not be tractable as an integral over the intensity function does not always have an analytic expression. But it is not the case for a homogeneous Poisson process with a constant intensity.

3.5 Recurrent Poisson Process

A recurrent Poisson process (RPP), consisting of a collection of homogeneous Poisson processes [38] for a series of time intervals, is a special type of temporal point process, in which the intensity function is determined by an RNN hidden state encoding the history of an acoustic signal. One may be tempted to learn the temporal point process simply using maximum likelihood estimation (MLE). Unfortunately, the annotation of the latent acoustic events in the acoustic speech signal is not available; hence direct supervised learning via MLE is not possible. Our RPP addresses this challenge by modeling these latent acoustic events as latent variables, which are then used as part of the generative process that is linked to the training targets.

3.5.1 Generate Timings for a Recurrent Poisson Process

Assume that we are given N intensities $\{\lambda_{t_1}, \lambda_{t_2}, \ldots, \lambda_{t_N}\}$, and a sequence of input features $\{\mathbf{x}_{t_{1-d}}, \ldots, \mathbf{x}_{t_1}, \mathbf{x}_{t_2}, \ldots, \mathbf{x}_{t_N}\}$, in which d context frames are padded to the left. Suppose the starting acoustic landmark is at time t_{1-d} and it follows a homogeneous Poisson process with intensity λ_{t_1} at the interval $[t_{1-d}, 2t_1 - t_{1-d}]$ which starts at time t_{1-d} and is centered at t_1. We will try to obtain the time estimate $\tilde{t}_1$ of the first acoustic landmark, and then repeat the procedure to obtain the whole $\widetilde{L} = \{\tilde{t}_1, \tilde{t}_2, \ldots, \tilde{t}_N\}$. To be more specific, given the $(i-1)$-th acoustic landmark at the estimated time $\tilde{t}_{i-1}$ and an interval $[\tilde{t}_{i-1}, 2t_i - \tilde{t}_{i-1}]$, the probability density of the next landmark being in this interval can be written as:

$$f_i^*(t) = \frac{f_i(t)}{\int_{\tilde{t}_{i-1}}^{2t_i-\tilde{t}_{i-1}} f_i(t)dt} = \frac{\lambda_{t_i} e^{-\lambda_{t_i}(t-\tilde{t}_{i-1})}}{1-e^{-2\lambda_{t_i}(t_i-\tilde{t}_{i-1})}} . \tag{3.9}$$

Then we can estimate the time for the *i*-th landmark as its expected value in following closed- form solution:

$$\begin{aligned} \tilde{t}_i &= \int_{\tilde{t}_{i-1}}^{2t_i-\tilde{t}_{i-1}} t f_i^*(t)dt \\ &= 2t_i - \tilde{t}_{i-1} + \frac{1}{\lambda_{t_i}} - \frac{2(t_i-\tilde{t}_{i-1})}{1-e^{-2\lambda_{t_i}(t_i-\tilde{t}_{i-1})}} \end{aligned} \tag{3.10}$$

Generally, the aforementioned point process can be factored into N independent homogeneous Poisson processes. For the *i*-th sub-process with intensity λ_{t_i}, the arrival subsequence L_i is drawn:

$$L_i \sim \mathfrak{P}_i(g(\lambda_{t_i})) . \tag{3.11}$$

It has only one single arrival and $L_i = \{t_i\}$. Thus, for a sequence of observations $\mathbf{L} = \{L_1, L_2, \ldots, L_N\}$, its likelihood is the following joint probability density:

$$P(\mathbf{L}|\lambda_{t_1}, \lambda_{t_2}, \ldots, \lambda_{t_N}) = \prod_{n=1}^{N} P(L_i|\lambda_{t_1}) \tag{3.12}$$

where

$$P(L_i|\lambda_{t_1}) = \lambda_{t_i} e^{-\lambda_{t_i}(t_i-t_{i-1})} . \tag{3.13}$$

3.5.2 Conditional Intensity Function for a Recurrent Poisson Process

In the neural spiking modelling [63], the intensity function of the neural spikes is usually conditioned on the external covariate. In a similar spirit, we determine our intensity function using the hidden state $\tilde{h}_{t_i}$ of an RNN, which encodes the temporal dependencies among the past history of the acoustic signal X. To avoid the explosion of $\frac{1}{\lambda_{t_i}}$, we define the inverse of the intensity function as:

$$\frac{1}{\lambda_{t_i}} = c\sigma(\phi(\tilde{h}_{t_i})) + \epsilon \tag{3.14}$$

such that the inverse of the intensity function is upper bounded by c and the intensity function is upper bounded by $\frac{1}{\epsilon}$. This is important as it limits the search space during optimization when $\phi(\cdot)$ are neural networks, which transform the hidden states into a scalar.

3.6 Recurrent Poisson Process Unit: Integrate Recurrent Poisson Process into RNN

The arrival time sequence of the acoustic landmarks generated from a recurrent Poisson process is on the real line. However, we are only given the discretized input sequence. The missing input vectors are reconstructed by linear interpolation as follows:

$$\widetilde{\mathbf{x}}_{\tilde{t}_i} = \sum_{n=1}^{N} \mathbf{x}_{t_n} \max(0, 1 - |\tilde{t}_i - n|) \ . \tag{3.15}$$

This enables the loss gradients to reach both the inputs and the estimated arrival times from the recurrent Poisson process.

In this paper, we use simple recurrent unit [40] to implement RNN. SRU simplifies the architecture of LSTM and dramatically reduces the computational time by dropping the connections between its hidden states and gates so that computation at the gates can be done in parallel.

Below are the updating formulas of recurrent Poisson process unit.

$$\left[\hat{\mathbf{r}}_{t_i}, \hat{\mathbf{f}}_{t_i}, \widehat{\mathbf{c}}_{t_i}\right] = \mathbf{W}_x \left[\mathbf{x}_{t_i}, \tilde{\mathbf{x}}_{\tilde{t}_i}\right] + \mathbf{b} \tag{3.16}$$

$$\mathbf{r}_{t_i} = \sigma(\hat{\mathbf{r}_{t_i}}) \tag{3.17}$$

$$\mathbf{f}_{t_i} = \sigma(\hat{\mathbf{f}_{t_i}}) \tag{3.18}$$

$$\mathbf{c}_{t_i} = \mathbf{f}_{t_i} \odot \mathbf{c}_{t_i - 1} + (1 - \mathbf{f}_{t_i}) \odot \hat{\mathbf{c}}_{t_i} \tag{3.19}$$

$$\mathbf{h}_{t_i} = \mathbf{r}_{t_i} \odot tanh(\mathbf{c}_{t_i}) + (1 - \mathbf{r}_{t_i}) \odot \mathbf{W}_h \left[\mathbf{x}_{t_i}, \tilde{\mathbf{x}}_{\tilde{t}_i}\right] \tag{3.20}$$

where $\mathbf{r}_{t_i}$ are the reset gate outputs; $\mathbf{f}_{t_i}$ are the forget gate outputs; $\mathbf{c}_{t_i}$ are the memory cell outputs; $\mathbf{W}_x$ and $\mathbf{W}_h$ are the weight matrices; $\mathbf{b}$ are the gate bias vectors; $\mathbf{h}_{t_i}$ are the hidden state outputs; any quantity with a 'hat' (e.g. $\widehat{\mathbf{c}}_{t_i}$) is the activation value of the quantity before an activation function is applied; $\odot$ is the element-wise multiplication operation; σ is the sigmoid function.

3.7 Learning

Our design of the loss function aims at striking a balance between the generation of arrival times of the latent acoustic events for clean training data and encoding sufficient uncertainty to capture the variability caused by the discretization errors and misalignments.

We use the standard Poisson process as the prior for the recurrent Poisson process to restrict the complexity of the approximated recurrent Poisson process. We measure the distance between the recurrent poisson process and the standard Poisson process by Kullback–Leibler divergence in terms of the inter-arrival time distribution. The inter-arrival time distribution for the i-th sub-process is defined as an exponential distribution $g(\lambda_{t_i})$. Since all these distributions are independent, we can enjoy the additive property of Kullback–Leibler divergence of these two processes:

$$\sum_{i=1}^{N} KL(g_s(\lambda = 1)||g(\lambda_{t_i})) = \sum_{i=1}^{N}(\lambda_{t_i} - \log(\lambda_{t_i}) - 1) \tag{3.21}$$

where $g_s(\lambda = 1)$ is the inter-arrival time distribution for a stand Poisson process.

Although we assume the original arrival times of landmarks, $\{1, 2, \ldots, N\}$, are noisy, the negative likelihood of this "incorrec" time sequence can be a desirable regularizer to avoid overfitting in noisy conditions.

As such, the total loss is the sum of the cross-entropy loss, negative log likelihood of noisy arrival time and the KL divergence between the underlying recurrent Poisson process and the standard process:

$$\begin{aligned} &-\log P(\mathbf{Y}|\mathbf{X}) - \alpha \log(P(\mathbf{L}|\lambda_{t_1}, \lambda_{t_2}, \ldots, \lambda_{t_N})) \\ &+ \beta \sum_{i=1}^{N} KL(g_s(\lambda = 1)||g(\lambda_{t_i}))) \end{aligned} \tag{3.22}$$

Since the negative log likelihood terms and the KL terms has exactly the same form in term of optimization. The final objective can be written as:

$$-\log P(\mathbf{Y}|\mathbf{X}) + \gamma \sum_{i=1}^{N} (\lambda_{t_i} - \log(\lambda_{t_i})) \tag{3.23}$$

where γ is the weight for the regularization term. We adopt the backpropagation through time (BPTT) for joinly training both recurrent Poisson process and recurrent Poisson process Unit.

CHAPTER 4

EXPERIMENTAL EVALUATION

4.1 Data Setup: ChiME-2, WSJ0 and WSJ0&1

We evaluated the proposed RPPU on three ASR corpora: ChiME-2 [67], WSJ0 [20] and WSJ0&1 [20, 11].

4.1.1 ChiME-2

CHiME-2 corpus is a medium-large vocabulary corpus, which was generated by convolving clean Wall Street Journal (WSJ0) utterances with binaural room impulse responses (BRIRs) and real background noise at signal-to-noise ratios (SNRs) in the range [-6,9] dB. The training set contains about 15 hours of speech with 7138 simulated noisy utterances. The transcriptions are based on those of the WSJ0 training set. The development and test sets contain 2460 and 1980 simulated noisy utterances, respectively. The WSJ0 text corpus, consisting of 37M words from 1.6M sentences, is used to train a trigram language model with a vocabulary size of 5k.

4.1.2 WSJ0

WSJ0 is a clean speech corpus recorded in a clean environment using close microphones. The standard WSJ0 si-84 training set with 7138 clean utterances was used for acoustic modeling. The evaluation was performed on eval92-5k which is a 5k-vocabulary non-verbalized test set, and the si-dt-05 dataset was used as the development set. The 5k trigram language model used for evaluation was trained from the WSJ0 text corpus.

4.1.3 WSJ0&1

WSJ0&1 is a complete Wall Street Journal speech corpus, which involves speech data from both WSJ0 and WSJ1. The training set WSJ0&1 si-284 with 36515 utterances contains approximately 80 hours of speech, 95% of which was used for training. The rest was used as the development set. The evaluation of WSJ0&1 was performed on the dev93-20k and eval93-20k test sets, both of which are 20k open-vocabulary non-verbalized test sets. The evaluation was performed with a 20k trigram language model trained from the transcription of WSJ0&1 si-284. We report the speech recognition performance in terms of word error rate (WER).

4.2 Preprocessing and Training Procedure

Acoustic hidden Markov models (HMM) based on Gaussian-mixture model (GMM), LSTM, SRU and quasi-RNN were built. GMM-HMM models employed fMLLR-adapted [19] 39-dimensional MFCC features. All neural-network-based models used 40-dimensional Mel-filterbank coefficients [6] without their derivatives. Inputs of all neural networks consisted of the current frame together with its 4 right contextual frames. We performed per-speaker mean and variance normalization for the input to all the neural network models.

GMM-HMM employed fMLLR-adapted 39-dimensional MFCC features and was trained using the standard Kaldi recipe [52]. They were then used to derive the state targets for subsequent RNN training through forced alignment for ChiME-2, WSJ0 and WSJ0&1. Specifically, the state targets were obtained by aligning the training data with the DNN acoustic model through the iterative procedure outlined in [12].

All RNNs were trained by optimizing the categorical cross entropy using BPTT and SGD. Prior to optimization, all the weight matrices were initialized following a LeCun Normal distribution introduced in [39]. We applied a dropout rate of 0.1 to the connections between recurrent layers. The learning rate for LSTM/SRU, Quasi-RNN and RPPU models was initially set to 0.25, 0.2, and 0.07 respectively. We decayed the learning rate until it went below 1×10^{-6}.

Table 4.1. Model configuraions for all datasets and the training time for CHiME-2. L: number of layers; N: number of hidden states per layer; P: number of model parameters; T: Training time per epoch (hr).

Model	L	N	P	T
LSTM	3	2048	130M	0.71
SRU	12	2048	156M	0.32
Quasi-RNN	12	1024	117M	0.22
RPPU	12	1024	142M	0.37

4.3 Models

We adopted SRU as the building block to construct the proposed RPPU and compare our proposed model with the following baselines: (i) The LSTM with three stacked layers; (ii) SRU with 12 stacked layers; (iii) quasi-RNN with 12 stacked layers and the highway connection [40].

The LSTM has only three stacked layers because we did not observe WER reduction by stacking more layers. To ensure similar numbers of model parameters for different models, we set the number of hidden states per layer to 2048 for both LSTM and SRU, and 1024 for both quasi-RNN and RPPU. The filter width of quasi-RNN was 3, which ensured a similar number of parameters for the RPPU. Our RPPU had 2 context frames padded to the left of the input and two previous hidden states padded to the left of the input of each hidden RPPU layer. For simplicity, in the intensity function of RPPU, c was set to 100 and ϵ was set to 0.01 (these two hyperparameters can be tuned to further improve performance).

4.4 Results and Analysis

4.4.1 Results on CHiME-2

Table 4.1 shows the model configurations of the baseline models and the new RPPU model for all datasets. The training time per epoch for CHiME-2 is also provided. The timing experiments used the Theano package and were performed on a machine running the Ubuntu operating system with a single Intel Core i7-7700 CPU and a GTX 1080Ti GPU.

Table 4.2. WER (%) on test set of CHiME-2.

Model	WER
LSTM	26.1
SRU	26.2
Quasi-RNN	26.1
RPPU	**24.4**

Table 4.3. Detailed WER (%) on the CHiME-2 test set.

Model	-6 Db	-3 Db	0 Db	3 Db	6 Db	9 Db
LSTM	42.4	33.5	26.7	21.1	17.3	15.3
SRU	42.5	34.0	26.2	22.2	17.4	15.1
Quasi-RNN	42.1	32.8	27.8	20.8	17.5	15.6
RPPU	**39.9**	**31.1**	**24.9**	**20.3**	**16.0**	**13.2**

Each model took around 25 iterations, and their average running time is reported. We can see that SRU is much faster than LSTM and our RPPU runs almost as fast as SRU while having a similar number of parameters.

Table 4.2 shows the word recognition performance of the baseline models and the new RPPU model for CHiME-2. Firstly, we can see that in CHiME-2, SRU performs slightly worse than LSTM in WER, and the quasi-RNN achieves a slightly better performance than SRU. Our proposed RPPU performs the best among all the candidates in terms of WER, outperforming the baselines by about 1.7% absolute.

A two-tailed Matched Pairs Sentence-Segment Word Error (MAPSSWE) significance test [23] was conducted with the null hypothesis that there is no performance difference between the RPPU and the baselines using the NIST sc_stats tool. The test results find a significant difference in performance between the RPPU and each baseline system at the level of $p < 0.001$ (The details are shown in Table A.1 on Appendix A).

We also report the detailed WERs as a function of the SNR in CHiME-2 shown in Table 4.3. For all SNRs, the RPPU outperforms other models by a large margin. This suggests that incorporating the recurrent Poisson process into RNN structures lends itself to the model's robustness.

To validate the effectiveness of the regularization term in RPPU for CHiME-2, we varied its weight γ to find the best configuration, as can be seen in Figure 4.1, We obtained

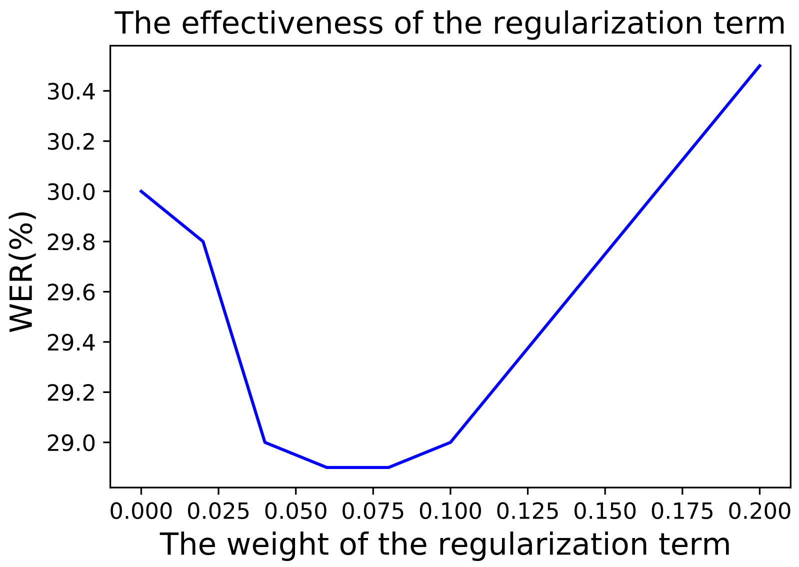

Figure 4.1. WER on Development set of CHiME-2 by varying the weight of the regularization term

the best performance in the development set when the weight γ is around 0.08. We hence set γ to 0.08 as our final configuration based on this observation. These results indicate the effectiveness of our proposed objective function.

4.4.2 Analyze the Property of RPP

Here, we took the generated time points from the recurrent Poisson process (RPP) of the 5-th layer of RPPU to perform both qualitative and quantitative analyses.

The standard Poisson process is the prior of the RPP in RPPU; hence we used the distance between the estimated value and its mean to approximately measure RPP's flexibility. To better understand how RPP works, we randomly took two utterances "423c02162" and "423c02166" at 9DB and -3DB SNR respectively, from the development set and generated the associated arrival times from the RPP. We display the estimated time points associated with acoustic events at two different SNRs in Figure 4.2. We can see that as the noise level increases, the estimated time points go towards the mean of a standard Poisson process. This suggests that RPPU can produce the time points based on the noise level: less flexibility is allowed for RPP's point generation when the data is too noisy.

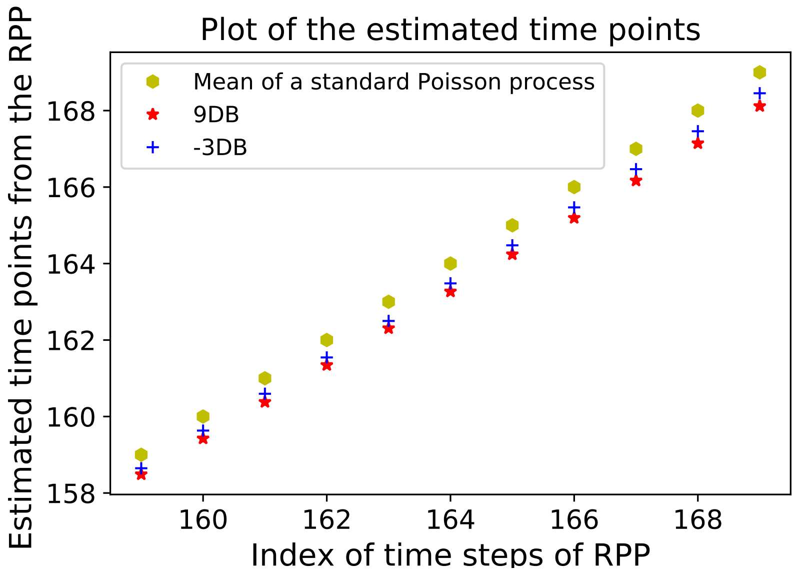

Figure 4.2. Arrival times produced by RPPU at each time step. The yellow points represent the mean of a standard Poisson process. The blue pluses and red stars represent the generated time points estimated from the randomly chosen utterance at 9DB and -3DB SNR, respectively.

Table 4.4. Similarity with the alignment generated from the clean development set of WSJ0

Alignment type	Similarity(%)
Alignment from Dev of ChiME2	55.5
Estimated alignment using RPPU	**64.7**

To evaluate how these generated time points can be helpful in better aligning the acoustic inputs with the acoustic HMM states, we conducted the analysis on the whole development set of CHiME-2. The "ground-truth" alignment for CHiME-2 on the phoneme level is obtained by force aligning the clean development set si-dt-05 in WSJ0 using the fMLLR-based DNN acoustic model trained with clean WSJ0. The WER of the model for the test set of WSJ0 is 2.6%, while the WER of our CHiME-2 DNN and our best RPPU for the noisy test set of CHiME-2 is 29.2% and 24.4%, respectively. Hence we believe our "ground truth" has much better alignment quality than the one obtained by force aligning the CHiME-2 noisy development set using any of our CHiME-2 acoustic models.

We firstly obtained the frame-level alignment by force aligning the CHiME-2 noisy development set using CHiME-2 DNN, which is the model we used to derive the training labels of CHiME-2 RNNs. We then used the generated time points from RPP to obtain an estimated phoneme-level alignment by replacing the integer time indices of the original frame-level alignment with such time points and then transforming it to phoneme-level alignment. We comparde it with both the original and the ground truth to see how RPPU works. We define the similarity between two alignments by calculating the percentage of their overlaps in time. The similarity with the "ground-truth" alignment for the original and the estimated one using RPPU are shown in Table 4.4. We can see that the similarity of estimated alignment achieves 9.2 % absolute gains. This demonstrates RPPU's capability in better aligning the acoustic inputs with the HMM state targets.

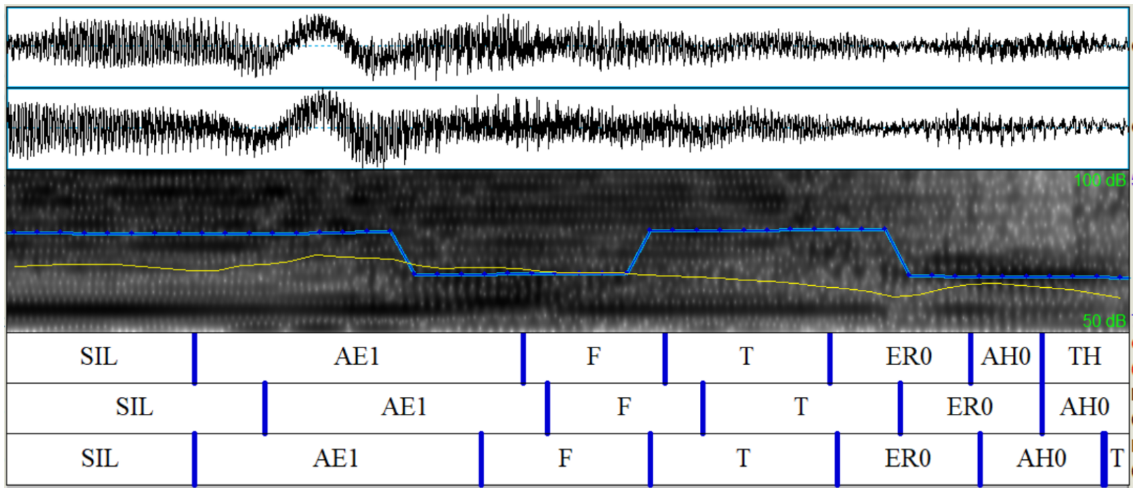

Figure 4.3. In the textgrids, the first tier represents the clean alignment generated from the clean utterance of WSJ0, the second tier represents the noisy alignment generated from the noisy utterance of CHiME-2, this last tier denotes the estimated alignment generated by using RPPU. The blue line and the yellow line in the middle spectrogram represents pitch and intensity, respectively.

Apart from the quantitative analysis, we show one example using Praat [7] to better understand how RPPU works. This example is the partial alignment of the randomly chosen utterance “050c01017” within the duration of the first 0.47 seconds. As shown in Figure 4.3, the third tier, which corresponds to the estimated alignment, is aligned much better with the clean alignment shown in the first tier than that of the noisy one in the second tier. Interestingly, it seems that the first boundary in the textgrids can be determined by the intensity in the yellow line, and that the right-hand boundary of 'F' can be determined by the rising of the pitch. The estimated alignment fits better with the clean alignments in terms of those two boundaries. This might suggest that RPPU is capable of

Table 4.5. WER (%) on evaluation set eval92-5k of WSJ0.

Model	WER
LSTM	2.8
SRU	2.8
Quasi-RNN	2.8
RPPU	**2.5**

Table 4.6. WER (%) on evaluation sets of WSJ0&1.

Model	dev93-20k	eval93-20k
LSTM	7.4	6.8
SRU	7.5	6.8
Quasi-RNN	7.4	7.0
RPPU	**6.9**	**6.2**

predicting the arrival of some acoustic events from some traits of the audio signal.

4.4.3 Results on WSJ0

To evaluate how RPPU behaves in a clean condition, we applied our method to WSJ0 which contains the clean utterances from which the CHiME-2 corpus was derived. We used the same model configurations of CHiME-2 for all RNN models. From Table 4.5, we can observe that all three RNN baseline systems using Mel-filterbank features achieve a WER of 2.8%. Our RPPU achieves the best WER of 2.5%, yielding a 10.7% relative performance gain over the other RNN baseline systems.

We conducted a two-tailed Matched Pairs Sentence-Segment Word Error (MAPSSWE) significance test [23] between the RPPU and the baselines using the NIST sc_stats tool. The test results find a significant difference in performance between the RPPU and each baseline system at the level of $p = 0.05$ (The details are shown in Table A.2 on Appendix A).

4.4.4 Results on WSJ0&1

We also conducted experiments on a larger corpus, WSJ0&1. The same model configurations of CHiME-2 were applied on all RNN models in WSJ0&1. The recognition results are shown in Table 4.6. We can see that our best baseline LSTM achieves a WER of 7.4% and 6.8% and our RPPU gives the lowest WER of 6.9% and 6.2% on dev93-20k and eval93-20k test sets, respectively. Overall, the RPPU achieves 6.8% and 9.1% relative WER reductions over the best LSTM baseline system on the two test sets.

We conducted a two-tailed matched pairs sentence-segment word error (MAPSSWE) significance test [23] between the RPPU and the baselines using the NIST sc_stats tool on dev93-20k and eval93-20k test sets. The test showed that the RPPU is significantly better than LSTM and SRU on dev93-20k test set, but the improvements for Quasi-RNN on this test set are not significant; It also showed that the RPPU is significantly better than SRU and Quasi-RNN on eval93-20k test set, but the improvements for LSTM on this test set are not significant. (The details are shown in Table A.3 and Table A.4 on Appendix A).

CHAPTER 5

CONCLUSION

This thesis is based on the hypothesis proposed in the field of speech generation and neuroethology: human speech signals can be encoded in point patterns involving acoustic events in speech signals and neural spikes in the brain. We attempt to solve the potential problems triggered by the RNN-HMM acoustic modeling given this hypothesis. These problems include discretization errors, unavailable event data, and misalignments issues.

It may be non-trivial to address these issues under the original framework of RNN-HMM. For example, the framework is difficult to avoid discretization errors because it discretizes the time series data at a fixed rate which is a standard way of handling input data for a RNN-HMM acoustic model. To solve these problems, we divide this framework into three steps:

- Allocate the training acoustic events localized in time at the HMM state level to better align with the training targets.
- Reconstruct/sample a series of acoustic features from the interval features originally sampled at a fixed frame rate from the allocated acoustic events.
- Follow the traditional ASR processing procedure using the newly reconstructed acoustic features as additional inputs.

We model these steps separately with the aid of RNN and a proposed recurrent Poisson process to create a more robust acoustic model (RPPU).

In this thesis, we presented a new acoustic model called recurrent Poisson process unit (RPPU). We have demonstrated that our model can generate much better alignments while performing the HMM state modeling. Our experiments on CHiME-2, WSJ0 and WSJ0&1 indicated that our method has achieved much better results than several RNN baselines in ASR.

5.1 Contributions

The most significant contribution of this thesis is that we proposed a new acoustic modeling method which is called recurrent Poisson process unit (RPPU). The key idea of our approach is to consider the arrivals of the latent acoustic events as several temporal latent variables. Each follows an associated sub-Poisson point process for a dynamic time interval, whose intensity function is parameterized by a recurrent neural network. More specifically, our work has made the following contributions:

- We proposed a novel recurrent Poisson process to jointly model the latent acoustic events and acoustic HMM states by treating the former as latent variables, which are then used as part of the generative process that is linked to the training targets: acoustic HMM states.
- We have provided an analytical solution to ensure efficiently drawing samples from a recurrent Poisson process (RPP). All of our RPPU components including the sampler of RPP are differentiable, so our model can be trained with the standard back-propagation through time (BPTT) [70] and stochaistic gradient descent (SGD) [57]. Despite the additional computation overhead caused by RPP, our RPPU runs almost as fast as SRU while having a similar number of parameters.
- We conducted both qualitative and quantitative analysis in speech recognition tasks and demonstrated that our model can generate much better alignments while performing the HMM state modeling. In addition, our method has achieved much better results than several baselines on many speech recognition datasets.

5.2 Future Works

5.2.1 More Experiments

Although the results we obtained in several ASR tasks are very promising, all of these ASR data corpora contain reading speech and text from Wall Street Journal news, and more experiments on spontaneous speech tasks should be conducted to confirm current findings. On the other hand, when we evaluated the effectiveness of RPPU in better aligning the acoustic inputs with the acoustic HMM states, we conducted the analysis on the whole development set of CHiME-2. However, the "ground-truth" alignment for CHiME-2 on the phoneme level was obtained by force aligning the clean development set si-dt-05 in WSJ0 using the fMLLR-based DNN acoustic model trained with clean WSJ0. As a result, we cannot ensure the alignment quality of the "ground-truth", which reduced the credibility of our subsequent associated qualitative and quantitative analysis. To solve this problem, it is better to apply our model to other applications such as sound event detection [45], in which ground truth of arrivals of sound events are available.

5.2.2 Model Improvement

Currently, we use a homogeneous Poisson process for each sub-point process of RPP. The reason we do this is because it is easy to understand and implement. However, due to history independence assumption, most real world data is not well described as a homogeneous Poisson process. Therefore, to capture the dynamics of many real world event data, history-dependent point processes should be taken into account. One alternative model for homogeneous Poisson process could be the Hawkes process [28]. The intensity function of this model is parameterised to be history-dependent. To further enhance the flexibility of our RPP and avoid making specific assumptions about the functional forms of the generative processes, the recurrent marked temporal point process [13] is also worth studying. On the other hand, it has been shown that bidirectional RNN models outperform undirectional ones [27]. It will be interesting to extend RPP by simply replacing the RNN in RPP by a bidirectional RNN to approximate the intensity function or by combining the representations from a forward-RNN and a backward-RNN at some other components within RPP.

REFERENCES

[1] Emmanuel Bacry, Adrian Iuga, Matthieu Lasnier, and Charles-Albert Lehalle. Market impacts and the life cycle of investors orders. *Market Microstructure and Liquidity*, 1(02):1550009, 2015.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[3] Timothy L Bailey, Charles Elkan, et al. Fitting a mixture model by expectation maximization to discover motifs in bipolymers. 1994.

[4] Leonard E Baum and Ted Petrie. Statistical inference for probabilistic functions of finite state markov chains. *The annals of mathematical statistics*, 37(6):1554–1563, 1966.

[5] Jacob Benesty, M Mohan Sondhi, and Yiteng Huang. *Springer handbook of speech processing*. springer, 2007.

[6] Alain Biem, Shigeru Katagiri, Erik McDermott, and Biing-Hwang Juang. An application of discriminative feature extraction to filter-bank-based speech recognition. *IEEE Transactions on Speech and Audio Processing*, 9(2):96–110, 2001.

[7] Paul Boersma et al. Praat, a system for doing phonetics by computer. *Glot international*, 5, 2002.

[8] James Bradbury, Stephen Merity, Caiming Xiong, and Richard Socher. Quasi-recurrent neural networks. *arXiv preprint arXiv:1611.01576*, 2016.

[9] 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.

[10] Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C Courville, and Yoshua Bengio. A recurrent latent variable model for sequential data. In *Proceedings of Advances in Neural Information Processing Systems*, pages 2980–2988, 2015.

[11] Linguistic Data Consortium et al. CSR-II (WSJ1) complete. *Linguistic Data Consortium, Philadelphia, vol. LDC94S13A*, 1994.

[12] George E Dahl, Dong Yu, Li Deng, and Alex Acero. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *IEEE Transactions on audio, speech, and language processing*, 20(1):30–42, 2012.

[13] Nan Du, Hanjun Dai, Rakshit Trivedi, Utkarsh Upadhyay, Manuel Gomez-Rodriguez, and Le Song. Recurrent marked temporal point processes: Embedding event history to vector. In *SIGKDD*, pages 1555–1564, 2016.

[14] Nan Du, Yichen Wang, Niao He, Jimeng Sun, and Le Song. Time-sensitive recommendation from recurrent user activities. In *Advances in Neural Information Processing Systems*, pages 3492–3500, 2015.

[15] Karl-Heinz Esser, Curtis J Condon, Nobuo Suga, and Jagmeet S Kanwal. Syntax processing by auditory cortical neurons in the fm–fm area of the mustached bat pteronotus parnellii. *Proceedings of the National Academy of Sciences*, 94(25):14019–14024, 1997.

[16] Laurene V Fausett. *Fundamentals of neural networks*. Prentice-Hall, 1994.

[17] Alceu Ferraz Costa, Yuto Yamaguchi, Agma Juci Machado Traina, Caetano Traina Jr, and Christos Faloutsos. Rsc: Mining and modeling temporal activity in social media. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 269–278. ACM, 2015.

[18] G David Forney. The Viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278, 1973.

[19] Mark JF Gales et al. Maximum likelihood linear transformations for HMM-based speech recognition. *Computer speech & language*, 12(2):75–98, 1998.

[20] John Garofolo, D Graff, D Paul, and D Pallett. CSR-I (WSJ0) complete. *Linguistic Data Consortium, Philadelphia*, 1993.

[21] John S Garofolo, Lori F Lamel, William M Fisher, Jonathan G Fiscus, and David S Pallett. Darpa timit acoustic-phonetic continous speech corpus cd-rom. nist speech disc 1-1.1. *NASA STI/Recon technical report n*, 93, 1993.

[22] Felix A Gers, Nicol N Schraudolph, and Jürgen Schmidhuber. Learning precise timing with LSTM recurrent networks. *Journal of machine learning research*, 3(Aug):115–143, 2002.

[23] Laurence Gillick and Stephen J Cox. Some statistical issues in the comparison of speech recognition algorithms. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 532–535. IEEE, 1989.

[24] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT Press, 2016.

[25] Alex Graves, Navdeep Jaitly, and Abdel-rahman Mohamed. Hybrid speech recognition with deep bidirectional LSTM. In *Proceedings of the IEEE Automatic Speech Recognition and Understanding Workshop*, pages 273–278. IEEE, 2013.

[26] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 6645–6649. IEEE, 2013.

[27] Navdeep Jaitly Graves, Alex and Abdel-rahman Mohamed. Hybrid speech recognition with deep bidirectional lstm. In *Proceedings of the IEEE Automatic Speech Recognition and Understanding Workshop*, pages 273–278. IEEE, 2013.

[28] Alan G Hawkes. Spectra of some self-exciting and mutually exciting point processes. *Biometrika*, 58(1):83–90, 1971.

[29] Alan G Hawkes. Spectra of some self-exciting and mutually exciting point processes. *Biometrika*, 58(1):83–90, 1971.

[30] Robert Hecht-Nielsen et al. Theory of the backpropagation neural network. *Neural Networks*, 1(Supplement-1):445–448, 1988.

[31] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012.

[32] Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, and Jürgen Schmidhuber. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies, 2001.

[33] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[34] Xuedong D Huang, Yasuo Ariki, and Mervyn A Jack. *Hidden Markov models for speech recognition*, volume 2004. Edinburgh university press Edinburgh, 1990.

[35] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[36] Aren Jansen and Partha Niyogi. Point process models for event-based speech recognition. *Speech Communication*, 51(12):1155–1168, 2009.

[37] Diederik P Kingma and Max Welling. Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[38] John Frank Charles Kingman. *Poisson processes*, volume 3. Clarendon Press, 1992.

[39] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. In *Advances in Neural Information Processing Systems*, pages 971–980, 2017.

[40] Tao Lei and Yu Zhang. Training RNNs as fast as CNNs. *arXiv preprint arXiv:1709.02755*, 2017.

[41] Jinyu Li and Chin-Hui Lee. On designing and evaluating speech event detectors. In *Proceedings of Interspeech*, 2005.

[42] Tsungnan Lin, Bill G Horne, Peter Tino, and C Lee Giles. Learning long-term dependencies in NARX recurrent neural networks. *IEEE Transactions on Neural Networks*, 7(6):1329–1338, 1996.

[43] R Dean Malmgren, Daniel B Stouffer, Adilson E Motter, and Luís AN Amaral. A poissonian explanation for heavy tails in e-mail communication. *Proceedings of the National Academy of Sciences*, pages pnas–0800332105, 2008.

[44] Hongyuan Mei and Jason Eisner. The neural hawkes process: A neurally self-modulating multivariate point process. In *NIPS*, pages 6757–6767, 2017.

[45] Annamaria Mesaros, Toni Heittola, and Tuomas Virtanen. Tut database for acoustic scene classification and sound event detection. In *Signal Processing Conference (EUSIPCO), 2016 24th European*, pages 1128–1132. IEEE, 2016.

[46] Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Proceedings of Interspeech*, volume 2, page 3, 2010.

[47] Abdel-rahman Mohamed, George Dahl, and Geoffrey Hinton. Deep belief networks for phone recognition. In *NIPS workshop on deep learning for speech recognition and related applications*, volume 1, page 39. Vancouver, Canada, 2009.

[48] Hema A Murthy, Francoise Beaufays, Larry P Heck, and Mitchel Weintraub. Robust text-independent speaker identification over telephone channels. *IEEE Transactions on Speech and Audio Processing*, 7(5):554–568, 1999.

[49] Yosihiko Ogata. On lewis' simulation method for point processes. *IEEE Transactions on Information Theory*, 27(1):23–31, 1981.

[50] Yosihiko Ogata. Statistical models for earthquake occurrences and residual analysis for point processes. *Journal of the American Statistical association*, 83(401):9–27, 1988.

[51] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. *Proceedings of the International Conference on Machine Learning*, 28:1310–1318, 2013.

[52] Daniel Povey et al. The Kaldi speech recognition toolkit. 2011.

[53] Lawrence R Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[54] Lawrence R Rabiner and Biing-Hwang Juang. Fundamentals of speech recognition. 1993.

[55] LR Rabiner, B-H Juang, and C-H Lee. An overview of automatic speech recognition. In *Automatic Speech and Speaker Recognition*, pages 1–30. Springer, 1996.

[56] Jakob Gulddahl Rasmussen. Temporal point processes: the conditional intensity function. *Lecture Notes, Jan*, 2011.

[57] Herbert Robbins and Sutton Monro. A stochastic approximation method. In *Herbert Robbins Selected Papers*, pages 102–109. Springer, 1985.

[58] Anthony J Robinson. An application of recurrent nets to phone probability estimation. *IEEE transactions on Neural Networks*, 5(2):298–305, 1994.

[59] Tony Robinson. A real-time recurrent error propagation network word recognition system. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 617–620. IEEE, 1992.

[60] Ruslan Salakhutdinov and Geoffrey Hinton. Deep Boltzmann machines. In *Proceedings of Artificial Intelligence and Statistics*, pages 448–455, 2009.

[61] Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.

[62] Olivier Siohan, Bhuvana Ramabhadran, and Brian Kingsbury. Constructing ensembles of asr systems using randomized decision trees. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages I–197. IEEE, 2005.

[63] Jasper Snoek, Richard S. Zemel, and Ryan Prescott Adams. A determinantal point process latent variable model for inhibition in neural spiking data. In *NIPS*, pages 1932–1940, 2013.

[64] Kenneth N Stevens. *Acoustic phonetics*, volume 30. MIT press, 2000.

[65] Kenneth N Stevens. Toward a model for lexical access based on acoustic landmarks and distinctive features. *The Journal of the Acoustical Society of America*, 111(4):1872–1891, 2002.

[66] Yee W Teh and Vinayak Rao. Gaussian process modulated renewal processes. In *Advances in Neural Information Processing Systems*, pages 2474–2482, 2011.

[67] Emmanuel Vincent, Jon Barker, Shinji Watanabe, Jonathan Le Roux, Francesco Nesta, and Marco Matassoni. The second CHiME speech separation and recognition challenge: Datasets, tasks and baselines. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 126–130, 2013.

[68] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3156–3164, 2015.

[69] Alex Waibel. Modular construction of time-delay neural networks for speech recognition. *Neural computation*, 1(1):39–46, 1989.

[70] Paul J Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

[71] Dong Yu and Li Deng. *Automatic speech recognition: A deep learning approach*. Springer, 2014.

[72] Ke Zhou, Hongyuan Zha, and Le Song. Learning social infectivity in sparse low-rank networks using multi-dimensional hawkes processes. In *Artificial Intelligence and Statistics*, pages 641–649, 2013.

[73] David Zipser. Subgrouping reduces complexity and speeds up learning in recurrent networks. In *Proceedings of Advances in Neural Information Processing Systems*, pages 638–641, 1990.

APPENDIX A

SIGNIFICANT TESTS

The statistical significance test tool sc_stats from National Institute of Standards and Technology (NIST) is used to compared different systems. It is a two-tailed matched pairs sentence-segment word error (MAPSSWE) significance test with the null hypothesis that there is no performance difference between RPPU and other baseline systems.

Here, we apply the test to compare our proposed RPPU and other baseline RNNs on CHiME-2, WSJ0 and WSJ0&1 . The abbreviations of various acoustic models are summarized as follows:

- lstm.txt.trn: LSTM-HMM acoustic model
- sru.txt.trn: SRU-HMM acoustic model
- quasi.txt.trn: Quasi-RNN-HMM acoustic model
- rppu.txt.trn: RPPU-HMM acoustic model

The test results are shown in Table A.1, Table A.2 Table A.3 and Table A.4.

Table A.1. Significant tests of the CHiME-2 experiments

```
,-----------------------------------------------------------------------------------------------------------.
|                                  Composite Report of All Significance Tests                               |
|                                                For the  Test                                              |
|                                                                                                           |
|                                          Test Name                          Abbrev.                       |
|                            -----------------------------------------      -------                         |
|                            Matched Pair Sentence Segment (Word Error)       MP                            |
|                                                                                                           |
|                                                                                                           |
|-----------------------------------------------------------------------------------------------------------|
| Test     ||              | lstm.txt.trn | sru.txt.trn  | quasi.txt.trn |        rppu.txt.trn          || Test    |
| Abbrev.  ||              |              |              |               |                              || Abbrev. |
|----------++--------------+--------------+--------------+---------------+------------------------------++---------|
|  MP      || lstm.txt.trn |              | ~    0.542   | ~    0.992    | rppu.txt.trn   <0.001    *** ||  MP     |
|----------++--------------+--------------+--------------+---------------+------------------------------++---------|
|  MP      ||  sru.txt.trn |              |              | ~    0.490    | rppu.txt.trn   <0.001    *** ||  MP     |
|----------++--------------+--------------+--------------+---------------+------------------------------++---------|
|  MP      || quasi.txt.trn|              |              |               | rppu.txt.trn   <0.001    *** ||  MP     |
|----------++--------------+--------------+--------------+---------------+------------------------------++---------|
|  MP      || rppu.txt.trn |              |              |               |                              ||  MP     |
|-----------------------------------------------------------------------------------------------------------|
|                  These significance tests are all two-tailed tests with the null hypothesis               |
|                  that there is no performance difference between the two systems.                         |
|                                                                                                           |
|                  The first column indicates if the test finds a significant difference                    |
|                  at the level of p=0.05.  It consists of '~' if no difference is                          |
|                  found at this significance level.  If a difference at this level is                      |
|                  found, this column indicates the system with the higher value on the                     |
|                  performance statistic utilized by the particular test.                                   |
|                                                                                                           |
|                  The second column specifies the minimum value of p for which the test                    |
|                  finds a significant difference at the level of p.                                        |
|                                                                                                           |
|                  The third column indicates if the test finds a significant difference                    |
|                  at the level of p=0.001 ("***"), at the level of p=0.01, but not                         |
|                  p=0.001 ("**"), or at the level of p=0.05, but not p=0.01 ("*").                         |
|                                                                                                           |
```

Table A.2. Significant tests of the WSJ0 experiments

```
,------------------------------------------------------------------------------------------------------------------.
|                                    Composite Report of All Significance Tests                                    |
|                                                  For the  Test                                                   |
|                                                                                                                  |
|                                              Test Name                          Abbrev.                          |
|                                 ------------------------------------------      -------                          |
|                                 Matched Pair Sentence Segment (Word Error)        MP                             |
|                                                                                                                  |
|                                                                                                                  |
|------------------------------------------------------------------------------------------------------------------|
|  Test    ||               | lstm.txt.trn | sru.txt.trn  | quasi.txt.trn |         rppu.txt.trn         ||  Test   |
| Abbrev.  ||               |              |              |               |                              || Abbrev. |
|---------++---------------+--------------+--------------+---------------+------------------------------++---------|
|   MP     || lstm.txt.trn  |              | ~     0.749  | ~     0.881   | rppu.txt.trn    0.030    *   ||   MP    |
|---------++---------------+--------------+--------------+---------------+------------------------------++---------|
|   MP     ||  sru.txt.trn  |              |              | ~     0.555   | rppu.txt.trn    0.043    *   ||   MP    |
|---------++---------------+--------------+--------------+---------------+------------------------------++---------|
|   MP     || quasi.txt.trn |              |              |               | rppu.txt.trn    0.017    *   ||   MP    |
|---------++---------------+--------------+--------------+---------------+------------------------------++---------|
|   MP     || rppu.txt.trn  |              |              |               |                              ||   MP    |
|------------------------------------------------------------------------------------------------------------------|
|                  These significance tests are all two-tailed tests with the null hypothesis                      |
|                  that there is no performance difference between the two systems.                                |
|                                                                                                                  |
|                  The first column indicates if the test finds a significant difference                           |
|                  at the level of p=0.05.  It consists of '~' if no difference is                                 |
|                  found at this significance level.  If a difference at this level is                             |
|                  found, this column indicates the system with the higher value on the                            |
|                  performance statistic utilized by the particular test.                                          |
|                                                                                                                  |
|                  The second column specifies the minimum value of p for which the test                           |
|                  finds a significant difference at the level of p.                                               |
|                                                                                                                  |
|                  The third column indicates if the test finds a significant difference                           |
|                  at the level of p=0.001 ("***"), at the level of p=0.01, but not                                |
|                  p=0.001 ("**"), or at the level of p=0.05, but not p=0.01 ("*").                                |
|                                                                                                                  |
```

Table A.3. Significant tests of the WSJ0&1 dev93-20k task

```
,------------------------------------------------------------------------------------------------------------.
|                                 Composite Report of All Significance Tests                                 |
|                                              For the  Test                                                 |
|                                                                                                            |
|                                           Test Name                       Abbrev.                          |
|                                 ------------------------------------------     -------                     |
|                                 Matched Pair Sentence Segment (Word Error)       MP                        |
|                                                                                                            |
|                                                                                                            |
|------------------------------------------------------------------------------------------------------------|
| Test   ||              | lstm.txt.trn | sru.txt.trn  | quasi.txt.trn |        rppu.txt.trn           ||  Test   |
| Abbrev.||              |              |              |               |                               || Abbrev. |
|--------++--------------+--------------+--------------+---------------+-------------------------------++---------|
|   MP   || lstm.txt.trn |              | ~    0.465   | ~    0.352    | rppu.txt.trn   0.005     **   ||   MP    |
|--------++--------------+--------------+--------------+---------------+-------------------------------++---------|
|   MP   ||  sru.txt.trn |              |              | ~    0.772    | rppu.txt.trn   0.024     *    ||   MP    |
|--------++--------------+--------------+--------------+---------------+-------------------------------++---------|
|   MP   || quasi.txt.trn|              |              |               |             ~  0.066          ||   MP    |
|--------++--------------+--------------+--------------+---------------+-------------------------------++---------|
|   MP   || rppu.txt.trn |              |              |               |                               ||   MP    |
|------------------------------------------------------------------------------------------------------------|
|                  These significance tests are all two-tailed tests with the null hypothesis                |
|                  that there is no performance difference between the two systems.                          |
|                                                                                                            |
|                  The first column indicates if the test finds a significant difference                     |
|                  at the level of p=0.05.  It consists of '~' if no difference is                           |
|                  found at this significance level.  If a difference at this level is                       |
|                  found, this column indicates the system with the higher value on the                      |
|                  performance statistic utilized by the particular test.                                    |
|                                                                                                            |
|                  The second column specifies the minimum value of p for which the test                     |
|                  finds a significant difference at the level of p.                                         |
|                                                                                                            |
|                  The third column indicates if the test finds a significant difference                     |
|                  at the level of p=0.001 ("***"), at the level of p=0.01, but not                          |
|                  p=0.001 ("**"), or at the level of p=0.05, but not p=0.01 ("*").                          |
|                                                                                                            |
```

Table A.4. Significant tests of the WSJ0&1 eval93-20k task

```
,-----------------------------------------------------------------------------------------------------------.
|                                  Composite Report of All Significance Tests                               |
|                                                 For the  Test                                             |
|                                                                                                           |
|                                            Test Name                        Abbrev.                       |
|                            ------------------------------------------     -------                        |
|                            Matched Pair Sentence Segment (Word Error)       MP                            |
|                                                                                                           |
|                                                                                                           |
|-----------------------------------------------------------------------------------------------------------|
|  Test   ||                | lstm.txt.trn | sru.txt.trn  | quasi.txt.trn |          rppu.txt.trn          ||  Test   |
| Abbrev. ||                |              |              |               |                                || Abbrev. |
|---------++----------------+--------------+--------------+---------------+--------------------------------++---------|
|   MP    || lstm.txt.trn   |              | ~    0.638   | ~    0.465    |             ~     0.116        ||   MP    |
|---------++----------------+--------------+--------------+---------------+--------------------------------++---------|
|   MP    ||  sru.txt.trn   |              |              | ~    0.787    | rppu.txt.trn    0.035    *     ||   MP    |
|---------++----------------+--------------+--------------+---------------+--------------------------------++---------|
|   MP    || quasi.txt.trn  |              |              |               | rppu.txt.trn    0.005    **    ||   MP    |
|---------++----------------+--------------+--------------+---------------+--------------------------------++---------|
|   MP    || rppu.txt.trn   |              |              |               |                                ||   MP    |
|-----------------------------------------------------------------------------------------------------------|
|                  These significance tests are all two-tailed tests with the null hypothesis               |
|                  that there is no performance difference between the two systems.                         |
|                                                                                                           |
|                  The first column indicates if the test finds a significant difference                    |
|                  at the level of p=0.05.  It consists of '~' if no difference is                          |
|                  found at this significance level.  If a difference at this level is                      |
|                  found, this column indicates the system with the higher value on the                     |
|                  performance statistic utilized by the particular test.                                   |
|                                                                                                           |
|                  The second column specifies the minimum value of p for which the test                    |
|                  finds a significant difference at the level of p.                                        |
|                                                                                                           |
|                  The third column indicates if the test finds a significant difference                    |
|                  at the level of p=0.001 ("***"), at the level of p=0.01, but not                         |
|                  p=0.001 ("**"), or at the level of p=0.05, but not p=0.01 ("*").                         |
|                                                                                                           |
```