12-2019

Modeling Interacting Time-Series Signals

Kantha Girish Gangadhara
kg2605@rit.edu

Modeling Interacting Time-Series Signals

by

Kantha Girish Gangadhara

THESIS
Presented to the Department of Computer Science
Golisano College of Computing and Information Sciences
Rochester Institute of Technology

in Partial Fulfillment
of the Requirements
for the Degree of

Master of Science in Computer Science

Rochester Institute of Technology
December 2019

Modeling Interacting Time-Series Signals

APPROVED BY

SUPERVISING COMMITTEE:

Dr. Ifeoma Nwogu, Advisor

Dr. Matthew Wright, Reader

Dr. Linwei Wang, Observer

Abstract

Many real-life systems consist of multiple information signals which might be potentially interacting with each other over time. These interactions can be estimated/modeled using techniques like Pearson Correlation (PC), Time lagged Cross Correlation (TLCC) and windowed TLCC, Dynamic Time Warping (DTW), and coupled Hidden Markov Model (cHMM). These techniques, excluding cHMM, cannot capture non-linear interactions and does not work well with multi-variate data. Although cHMM can capture the interactions effectively, it is bound by Markov property and other assumptions like latent variables, prior distributions, etc. These influence the performance of the model significantly. Recurrent Neural Network (RNN) is a variant of Neural Networks which can be used to model time-series data. RNN based architectures are the new state-of-the-art for complex tasks like machine translation. In this research, we explore techniques to extend RNNs to model interacting time-series signals. We propose architectures with cou-

pling and attention mechanism. We evaluate the performance of the models on synthetically generated and real-life data sets. We compare the performance of our proposed architectures to similar ones in the literature. The goal of this exercise is to determine the most effective architecture to capture interaction information in the given interrelated time-series signals.

Acknowledgments

I would like to express my deepest gratitude to my advisor, Professor Ifeoma Nwogu, for guiding me to endeavor this journey and helping me through this project. Without your support, this project wouldn't have been an amazing research experience. I would like to thank Dr. Matthew Wright and Dr. Linwei Wang for being on my committee and providing valuable feedback, which helped me sharpen this thesis.

I would like to thank Fei Xu, a Ph.D. student at the University of Buffalo, for putting together a real-life dataset and providing us with extracted features to work with. Without your contribution, we would not have had a chance to evaluate our proposed techniques on real-life data.

I would like to thank Parikshit Prashant Shembekar for helping us with feature extraction on a real-life dataset. I would like to thank my academic advisor, Rebecca O Connor, for guiding me through the administrative details of the project and providing the necessary help when in need.

I would like to express my gratitude to Aishwarya Uniyal, my friend and room-mate, for insightful discussions, and being a source of support and inspiration throughout this journey.

Finally, I would like to thank my friends and family; this project would not have been possible without your emotional support.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Many systems in real-life consist of multiple information signals interacting with each other over time. Social interactions serve as good examples of such systems. These include dyadic interactions of a married couple, interrogation room interaction between the investigator and the crime suspect, corporate board room meetings, etc, to name a few. The harmony or synchrony in such interactions is formally known as *rapport*. Rapport is defined as subtle mirroring of emotional behavior such as postures, nods, and smiles. Tickle-Degnen et al. [25] characterize rapport as a composition of the following three components:

1. *Mutual attentiveness:* the feeling of intense mutual interest between participants in a conversation, which creates a focused and cohesive

interaction.

2. *Positivity:* a feeling of mutual friendliness.

3. *Coordination*

Numerous studies have demonstrated that rapport facilitates negotiations [9], psychotherapy [26], etc. Aimee et al. [9] state that real-life interactions are mixed-motive conflicts due to individual self-interests. The collective optimal outcome of these interactions requires mutual co-operation or rapport. We might be interested in modeling social interactions for potential outcomes like quality of interaction, agreement/disagreement between the individuals, etc. In such cases, modeling interactions in terms of rapport would be successful.

Modeling interactions might be crucial for the potential outcomes we would be interested in. In this project, we propose neural-network-based architectures for modeling interacting signals. We study traditional methods in the literature which can be used to model such systems, and discuss why they might not be effective in modeling multi-variate signals. We compare our proposed architectures to similar architectures in the literature [21], which were not necessarily used to model social interactions. As a result, we evaluate the suitability of these architectures for modeling social interactions. We use synthetically generated and real-life datasets to evaluate the performance of the models we work with.

1.2 Related Work

Modeling interactions have been extensively studied in the literature. Traditionally, the following techniques have been used,

1. **Pearson Correlation (PC):** is a measure of linear correlation between two signals X and Y [3], a value in the range [-1, +1]. A positive value indicates a positive correlation, whereas a negative value indicates a negative correlation. This measure quantifies global synchrony between two signals. It can be calculated using,

$$\rho(X, Y) = \frac{cov(X, Y)}{\sigma_X \sigma_Y} \tag{1.1}$$

   where,

   $cov()$ is the covariance between X and Y

   σ_X, σ_Y are standard deviations of X, Y respectively

2. **Time Lagged Cross Correlation (TLCC):** is an extension of Pearson Correlation. One of the signals is offset by one timestep at a time and PC is calculated using equation 1.1.

3. **Dynamic Time Warping (DTW):** is a measure of similarity between two temporal signals using an algorithm with defined set of rules

and constraints. Figure 1.1 briefly illustrates how the algorithm compares two signals. This technique uses a dynamic programming approach to align two time-series signals by minimizing a distance metric [5].

4. **coupled Hidden Markov Model (cHMM):** is a probabilistic framework which couples multiple temporal signals [6]. Conventional HMMs are coupled by taking Cartesian product of their hidden states (figure 1.2). An HMM is a probabilistic model, modeled as a Markov process with observable variables and unobservable (hidden) states. The formulation is as below,

   Let $X = \{x_1, x_2, \ldots, x_n\}$ be a set of unobservables or hidden states, and $Y = \{y_1, y_2, \ldots, y_n\}$ be a set of observables. Then,

   - $p(y_i|x_i)$ emission probability for the observable at time step i
   - $p(x_i|x_{i-1})$ transition probability

   A cHMM can be setup by extending the above formulations of a HMM, as below,

   Let $X = \{x_1, x_2, \ldots, x_n\}$ and $Z = \{z_1, z_2, \ldots, z_n\}$ be the hidden states.

The coupled transition probabilities are,

$$p(c_{i,j}|c_{i-1,j-1}) = \psi\Big(p(x_i|x_{i-1}),\, p(z_j|z_{j-1}),\, p(x_i|z_{j-1}),\, p(z_j|x_{i-1})\Big)$$

The coupled emission probabilities are as below,

$$p(y|c_{i,j}) = p(y|x_i) \cdot p(y|z_j)$$

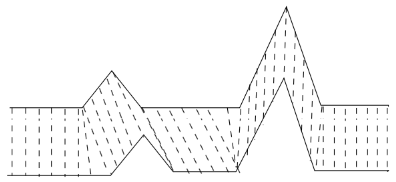


Figure 1.1: DTW algorithm comparing two signals

The techniques: PC, TLCC, and DTW have the following disadvantages:

- cannot be used for multi-variate data and inference based modeling
- doesn't capture non-linear interactions
- cannot handle noisy data

The cHMMs have been applied to classify taichi movements [6, 22], under the assumption that different parts of the body moving in taichi will be synchronous to each other. The cHMMs are bound by first-order Markov assumption, which states that the present state (time step t) is dependent

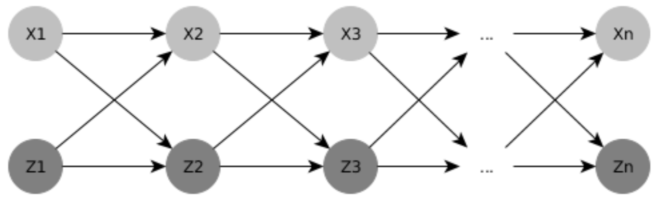


Figure 1.2: A cHMM with two channels

on the immediate past (time step $t-1$) alone, making them restrictive and limited. This assumption makes them tractable and relatively easy to train. Whereas this does not hold true for most real-world systems, and hence cHMMs cannot be used to model such systems.

Recently, neural-network-based architectures have been successfully applied to model time-series data. The two variants of RNNs, Long Short Term Memory (LSTM) and Gated Recurrent Units (GRU), have been successful in learning long term sequences. A few studies have explored methods to model interactions between a pair of signals by coupling the hidden states of two recurrent units. These methods try to achieve coupling by concatenating the hidden states with each other or with the input signal of the succeeding recurrent step. We mention the studies and discuss their coupling approaches below.

Liu et al. [19] present an architecture that couples LSTMs to model sentence similarity amongst pairs. They propose two variants, namely loosely coupled LSTM (LC-LSTM) and tightly coupled LSTM (TC-LSTM). They use two LSTMs, one for each sentence in the pair, and couple the hidden states in four directions, inspired by grid LSTM [16] and multi-dimensional RNN [14]. The hidden states of the LSTMs are concatenated and fed as the hidden input for the subsequent recurrent steps. Both of these proposed architectures become highly complex proportional to the length of sequences, due to multi-directional coupling. This might result in vanishing or exploding gradients. Also, it is unclear how they keep the hidden dimensions constant for the recurrent steps post concatenation of the hidden states.

Sun et al. [24] propose Coupled Recurrent Network (CRN), an architecture with two parallel streams of LSTMs. The hidden state of one stream is concatenated with the input signal of the other stream in the succeeding recurrent step. They use convolution operations for the LSTM memory cell, as they work with images. Similar to CRN in terms of coupling, Morais et al. [21] introduce MPED-RNN with two parallel streams of GRUs for anomaly detection in videos. They use skeleton trajectories extracted from the videos and decompose them into two input signals. These signals are fed to MPED-RNN, an encoder-decoder architecture with a 2-stream encoder for inputs and two 2-stream decoders for outputs, one for reconstructing input and one for predicting future trajectories. Both CRN and MPED-RNN are similar and they do not couple the hidden representations directly. Rather, the hidden states are concatenated with inputs, which introduces complexity as the number of signals increase.

Inspired by the work of Liu et al. [19], we propose an architecture with two parallel GRU streams and uni-directional coupling. We use a multi-layer perceptron (MLP) to couple and reduce the dimensionality of the hidden states. We hypothesize that using an MLP for coupling would allow for capturing the interactions better. By increasing the complexity of this unit, we could capture non-linear interactions better. Recent literature has shown that attention mechanism helps to better retain temporal memory for long sequences [1]. Inspired by this, we implement a variant of the proposed network with a simple attention mechanism. This causes the network to consider all the hidden states when performing supervised learning tasks such as classification or regression, and generative modeling tasks. We compare the performance of our proposed architectures to MPED-RNN [21] on synthetic and real-life datasets.

1.3 Hypothesis

We hypothesize the following,

1. If systems consist of interacting information signals, modeling the interactions might be crucial for supervised learning tasks - classification and regression

2. Interactions can be modeled by coupling hidden states of RNNs

3. Attention mechanism further improves the learning task of coupled models

4. The performance of our proposed architectures are comparable to similar methods in the literature while being more compact and easily scalable to complex systems

We build and implement architectures to evaluate these hypotheses by training them on real-life and synthetically generated datasets.

Chapter 2

Background

In this chapter, we will cover the necessary concepts. We describe RNNs and its variants, and attention mechanism in detail.

2.1 Recurrent Neural Networks

Recurrent neural networks are an extension of conventional feed-forward neural networks. As opposed to conventional neural networks, RNNs can be used to model inputs of variable sequence lengths. This is achieved by using a recurrent hidden state. At each time step t, the activation of the hidden state is dependent on the previous time step $t-1$.

Consider a sequence $X = \{x_1, x_2, \ldots, x_n\}$ with N time steps, and h_t as the hidden state at time step t such that $t \in [1, n]$. Then, h_t is updated as,

$$h_t = \begin{cases} 0 & t = 0 \\ f(h_{t-1}, x_t) & \text{otherwise} \end{cases} \tag{2.1}$$

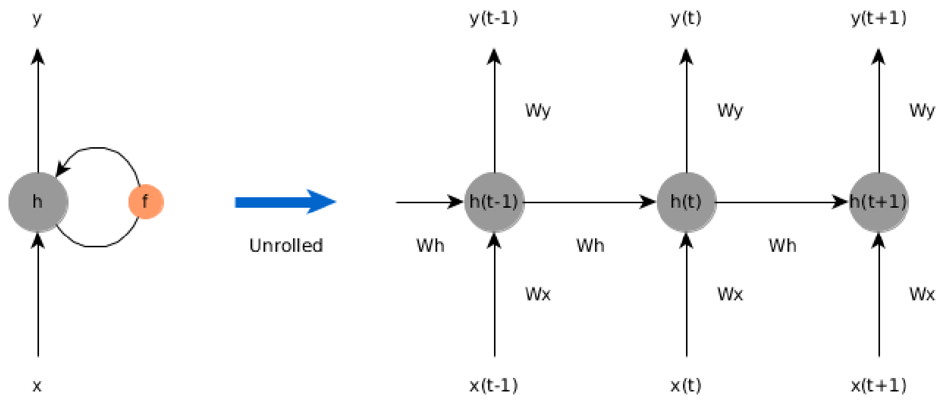


Figure 2.1: An RNN unrolled in time

This update to the hidden state is implemented as (Figure 2.1),

$$h_t = f(W_x x_t + W_h h_{t-1}) \tag{2.2}$$

where, $f()$ is a smooth bounded non-linear function such as *logistic sigmoid* or *hyperbolic tangent* (*tanh()*). The output could either be a variable length sequence such as $Y = y_1, y_2, \ldots, y_m$, or it could be a fixed number of units to perform classification or regression.

An RNN can be trained as a generative model to generate sequences. Given its hidden state and the input for the time step t, this model outputs a probability distribution for the next element in the sequence. When training such models, a special input/output symbol is used to represent the start or end of the sequence. In the case of a word-level language model, the start of a sentence would be <*SOS*> and the end of the sentence would be <*EOS*>. The probability distribution of the sequence element is modeled as a conditional probability distribution over past elements in the sequence, as below,

$$p(x_t|x_1, x_2, \dots, x_{t-1}) = g(h_t) \tag{2.3}$$

where, $g()$ is a smooth bounded non-linear function such as *logistic sigmoid* or *hyperbolic tangent* (*tanh()*).

An RNN can be used for sequence classification or regression problems. Throughout this project, we stick to problems of this nature. Consider the above sequence $X = \{x_1, x_2, \dots x_n\}$, which maps to a binary class $Y \in \{0, 1\}$. The probability distribution can be modeled as a conditional,

$$p(Y \mid x_1, x_2, \dots, x_n) = \text{sigmoid}(l(h_n)) \tag{2.4}$$

where, $l()$ is a linear function which reduces the dimension of h_n to one. The same formulation can be used to model a regression problem by removing sigmoid

RNNs are trained using the gradient-based learning algorithm *back-propagation through time* [28] (BPTT). It is an extension of the back-propagation algorithm [23] to facilitate gradient propagation through the recurrent states. Theoretically, RNNs can capture the temporal dependencies in relatively long sequences effectively. In practice, this has been proven false [4]. The gradients either explode or vanish when the sequences are long due to long product chains of partial derivatives. Alternatives like gradient clipping have been effective in alleviating the exploding gradient problem. However, the best alternatives so far have been the variants of RNNs - Long Short Term Memory (LSTM), and Gated Recurrent Units (GRU). Both of them use additional gates to mitigate the problem and help in learning long-term dependencies in the sequences. The following section describes GRUs in detail. We skip LSTMs as they haven't been used in this project.

2.2 Gated Recurrent Unit

Gated Recurrent Unit (GRU) was introduced by Cho et al. [7], inspired by LSTM. However, it is much simpler compared to LSTMs in terms of implementation and computation.

A GRU is made up of (Figure 2.2),

- **Reset gate:** This gate learns to decide how much of the past information to forget. It resets the past information when the sigmoid activation is close to zero (equation 2.7).

$$r_t = \sigma(W_r x_t + U_r h_{t-1}) \tag{2.5}$$

- **Update gate:** This gate controls the amount of information flow from the previous hidden state to the current state. The model can essentially decide to copy all the past information and eliminate the risk of vanishing gradient problem (equation 2.8).

$$z_t = \sigma(W_z x_t + U_z h_{t-1}) \tag{2.6}$$

- **Candidate for current activation:** The element-wise product of h_{t-1} with r_t is to decide what to remove from the previous time steps.

$$\tilde{h_t} = \tanh(W_h x_t + U_h(r_t \cdot h_{t-1})) \tag{2.7}$$

- **Current hidden state:**

$$h_t = z_t h_{t-1} + (1 - z_t)\tilde{h_t} \tag{2.8}$$

When learning sequences, the update and reset gates pass down the relevant past information to the future time steps. This alleviates the vanishing gradient problem and learn long-term dependencies better. The performance has been shown to be comparable to that of LSTMs [8]. For this project, we choose GRUs over LSTMs to leverage its simplicity and computational advantages.

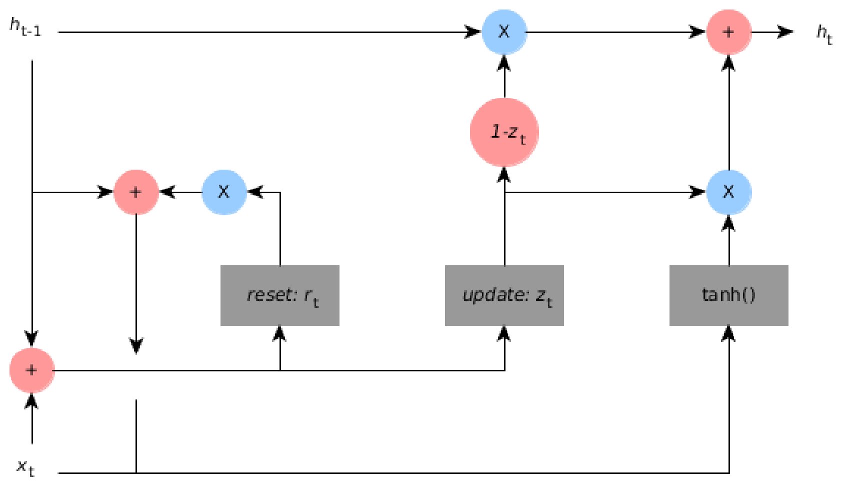


Figure 2.2: Gated Recurrent Unit

2.3 Encoder-Decoder architecture

Encoder-Decoder architecture is a way of stacking recurrent units for modeling multi-input and multi-output data. A popular example is a word-level machine translation model (Figure 2.3). The number of words in a source sentence and the target sentence may or may not be the same. The encoder contains a recurrent unit that encodes the words in the source sentence into a fixed-length context vector. The context vector is the last hidden state of the encoder. The decoder contains a recurrent unit that is initialized with this context vector. The target sentence is decoded one word at a time.

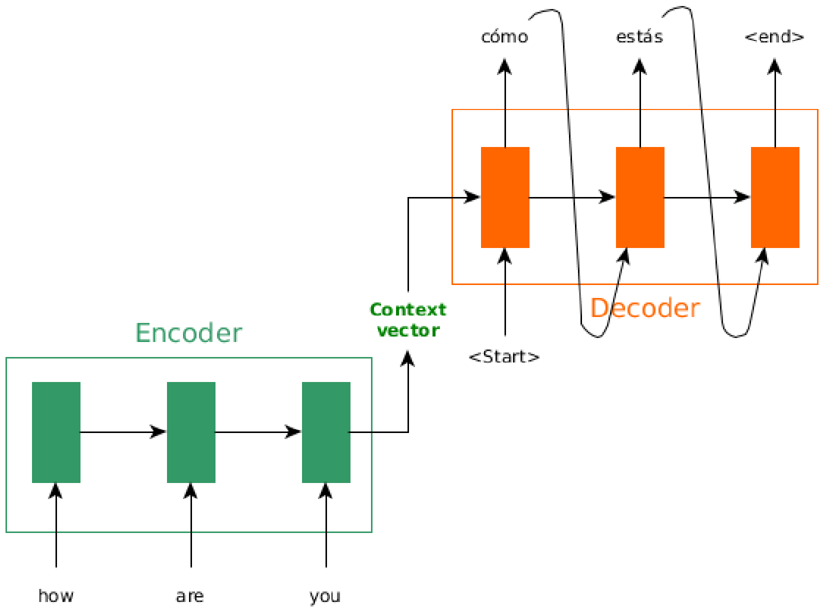


Figure 2.3: Encoder-Decoder architecture - machine translation

2.4 Attention mechanism

The fixed-length context vector of an RNN has the burden of encoding the temporal information in sequences of arbitrary length. Hence, it might forget the information about the initial time steps when the sequences are longer. This problem has been observed in Neural Machine Translation (NMT) tasks. When translating a long sentence from a source language to a target language, the context vector has the burden of encoding temporal information of all the words. Bahdanau et al. [1] proposed **attention mechanism** to overcome this problem. Essentially, this mechanism exposes all the hidden states of an encoder to the decoder. At the time of decoding a word to the target language, the decoder decides the amount of information to use from each hidden state by computing a weighted sum of all the hidden states. The model learns to pay *attention* to relevant parts of information to make predictions, hence the name. This has been successful in producing state-of-the-art results in neural machine translation [27].

The formulation we use for this project is from the work of Bahdanau et al. [1]. We modify the formulation for classification or regression problems. We use the same notations as earlier.

A d dimensional hidden state h_t is reduced to a scalar as below,

$$e_t = a(h_t) \tag{2.9}$$

Using this, we compute the weight α_t for the state h_t as,

$$\alpha_t = \frac{\exp(e_t)}{\sum_{j=1}^{N} \exp(e_j)} \tag{2.10}$$

The above equation is the definition of the standard *softmax* function, which is used for training multi-class classification models. This function computes probability distribution such that $\sum_{j=1}^{N} \alpha_j = 1$.

We then compute the context vector c as,

$$c = \sum_{j=1}^{N} \alpha_j h_j \tag{2.11}$$

This new context vector is a weighted sum of all the hidden states. The conditional probability formulation for binary classification changes to,

$$p(Y \mid x_1, x_2, \ldots, x_n) = \text{sigmoid}(l(c)) \tag{2.12}$$

where, $l()$ is a linear function which reduces the dimension of c to one.

We can train an RNN with the attention mechanism in an end-to-end fashion. The function $a()$ and computation of α_i together can be parameterized as a feed-forward neural network. These parameters are learned in the back-propagation step. This approach is called self-attention. An RNN trained in this fashion can be more effective at learning the amount of temporal information necessary for assigning a class label.

Chapter 3

Methodology

In this chapter, we introduce our proposed architectures built using GRUs. Throughout this project, we stick to two interacting signals. For the rest of the chapter, we consider the following notations,

- Signals $S^1 = \{s_1^1, s_2^1, \ldots, s_n^1\}$ and $S^2 = \{s_1^2, s_2^2, \ldots, s_n^2\}$ are two interacting sequences of equal length
- $g()$ is a smooth bounded non-linear function such as *logistic sigmoid* for the binary classification problem, or a linear function for the regression problem
- We use a simplified formulation for a GRU as below,

$$h_t = \mathrm{GRU}(x_t, h_{t-1}) \tag{3.1}$$

In addition to our proposed architectures, we also describe a version of

MPED-RNN [21], which we call MP-RNN (Section 3.4). This is a simplified version of MPED-RNN without the encoder-decoder architecture. We add appropriate non-linear functions for classification and regression problems. We compare the performance of our architectures against MP-RNN.

3.1 End to End network (end2end-GRU)

This architecture outlines the simplest way of modeling two time-series signals (Figure 3.1). It consists of two GRUs, GRU-1 and GRU-2, one for each signal. The corresponding hidden states are h_t^1 and h_t^2. The context vectors of both the GRUs are concatenated and fed to a function to perform classification/regression. The formulation is as below,

Hidden states h_t^1, h_t^2 are computed as,

$$h_t^1 = \text{GRU-1}(s_t^1, h_{t-1}^1) \tag{3.2}$$

$$h_t^2 = \text{GRU-2}(s_t^2, h_{t-1}^2) \tag{3.3}$$

Supervised machine learning problems can be formulated as,

$$p(Y \mid S^1, S^2) = g(W_h * [h_n^1; h_n^2]) \tag{3.4}$$

It is evident that this architecture can be used to model sequences of arbitrary lengths. However, this architecture may not capture the local/global

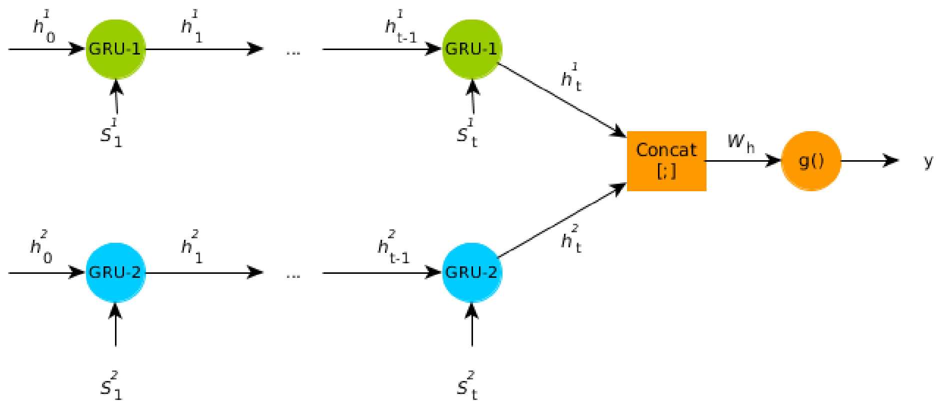


Figure 3.1: end2end-GRU unrolled in time

temporal interactions between the signals being studied. We use this architecture as a baseline to evaluate our hypothesis 1.

3.2 Coupled GRU (cGRU)

This architecture is an extension of end2end-GRU (Figure 3.2). The hidden states of each time step are fed to a Multi-layer perceptron (MLP) with a non-linear function. The output dimension of this unit is the same as the dimension of the hidden state of either GRU's. We call this the coupled hidden state. This is fed to both GRUs as the input hidden state for the succeeding time step. The MLP is a standard multi-layered feed-forward neural network with a non-linear activation function following each layer. Increasing layers might better capture non-linear local interactions between the signals. The formulation is as below,

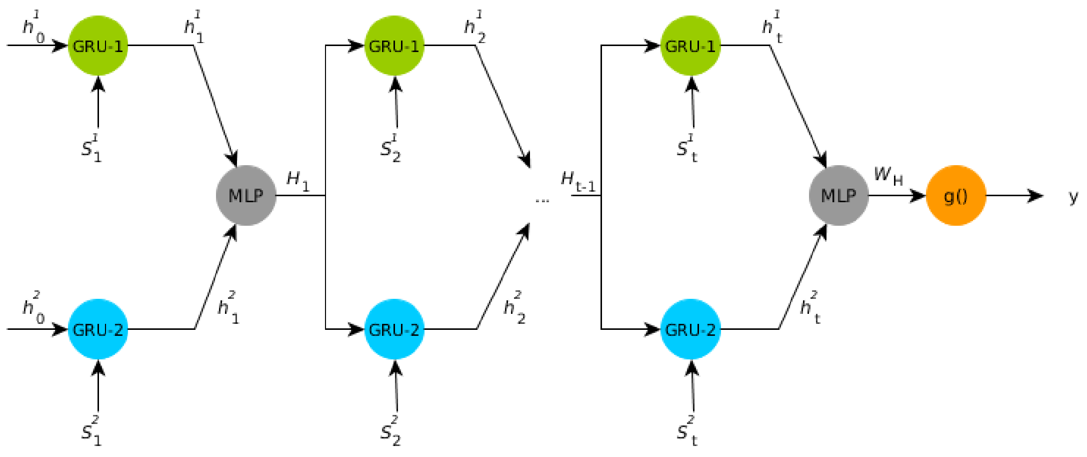


Figure 3.2: cGRU unrolled in time

Hidden states h_t^1, h_t^2 are computed as,

$$h_t^1 = \text{GRU-1}(s_t^1, H_{t-1}) \tag{3.5}$$

$$h_t^2 = \text{GRU-2}(s_t^2, H_{t-1}) \tag{3.6}$$

The coupled hidden state H_t is computed as,

$$H_t = f([h_t^1\,;\, h_t^2]) \tag{3.7}$$

where, $f()$ is an MLP with a non-linear activation function such as *tanh*. Supervised machine learning problems can be formulated using the following conditional probability function *tanh*,

$$p(Y \mid S^1, S^2) = g(W_H * H_t) \tag{3.8}$$

3.3 Coupled GRU with Attention mechanism (cGRU-Attention)

This architecture is an extension of cGRU (Figure 3.4). The coupled hidden states are used to apply self-attention and create a weighted context vector. We use the following formulations,

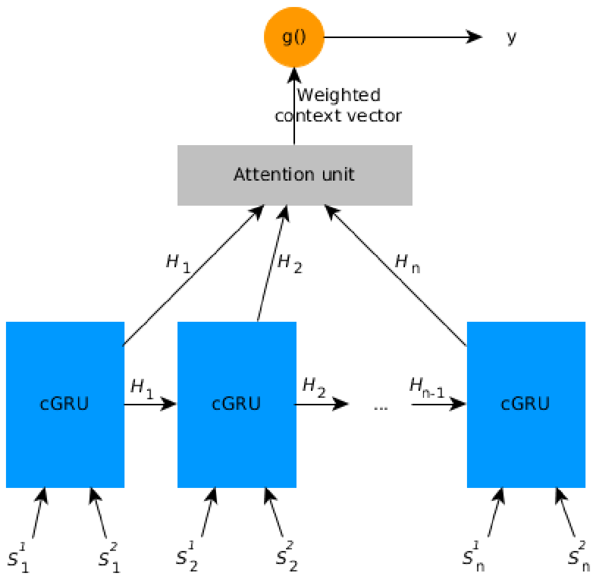


Figure 3.3: cGRU with attention

Using equation 2.9, the coupled hidden H_t state is reduced to a scalar as below,

$$e_t = a(H_t)$$

We use equation 2.10 to compute the weight α_t for each state H_t. We then compute the weighted context vector C as (equation 2.10),

$$C = \sum_{j=1}^{N} \alpha_j H_j$$

The conditional probability formulation for a supervised learning problem is,

$$p(Y \mid S^1, S^2) = g(C) \tag{3.9}$$

3.4 Message Passing (MP-RNN)

This architecture is a version of MPED-RNN [21] with Encoder and an appropriate activation function for a supervised learning task. A rectified linear unit is used as an activation function to convert hidden states to messages.

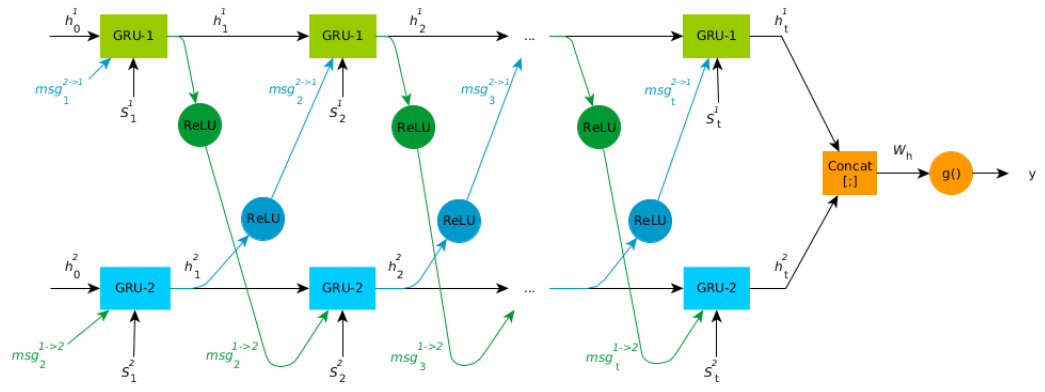


Figure 3.4: MP-RNN

Message signals are computed as,

$$msg_t^{1\to2} = ReLU(W^{1\to2} * h_t^1) \tag{3.10}$$

$$msg_t^{2\to1} = ReLU(W^{2\to1} * h_t^2) \tag{3.11}$$

The hidden states h_t^1, h_t^2 are computed using message signals as below,

$$h_t^1 = \text{GRU-1}([S_t^1; msg_t^{2\to1}], h_{t-1}^1) \tag{3.12}$$

$$h_t^2 = \text{GRU-2}([S_t^2; msg_t^{1\to2}], h_{t-1}^2) \tag{3.13}$$

The conditional probability formulation for a supervised learning problem is,

$$P(y|S^1, S^2) = g(W_h * [h_t^1; h_t^2]) \tag{3.14}$$

3.5 Coupled GRU Encoder-Decoder (cGRU-ED)

This is a generative architecture similar MPED-RNN [21]. The architecture is built using cGRUs (figure 3.6). We use an abstraction of cGRU to simplify the illustration of cGRU-ED architecture (figure 3.5). It consists of the following components,

1. **Encoder:** encodes a pair of sequences of length T into a coupled context vector H_t using,

$$H_t = \text{cGRU}(H_{t-1}, S_t^1, S_t^2)$$

2. **Reconstruction Decoder:** is initialized with the hidden state $H_t^R = H_t$. It reconstructs the input pair in the reverse order using,

$$H_{t-1}^R, S_{t-1}^1, S_{t-1}^2 = \text{cGRU}(H_t^R, S_t^1, S_t^2)$$

3. **Prediction Decoder:** is initialized with the hidden state $H_t^P = H_t$, predicts future pair of length k. The formulation is similar to the reconstruction decoder, as below,

$$H_{t+k}^P, S_{t+k}^1, S_{t+k}^2 = \text{cGRU}(H_{t+k-1}^P, S_{t+k-1}^1, S_{t+k-1}^2)$$

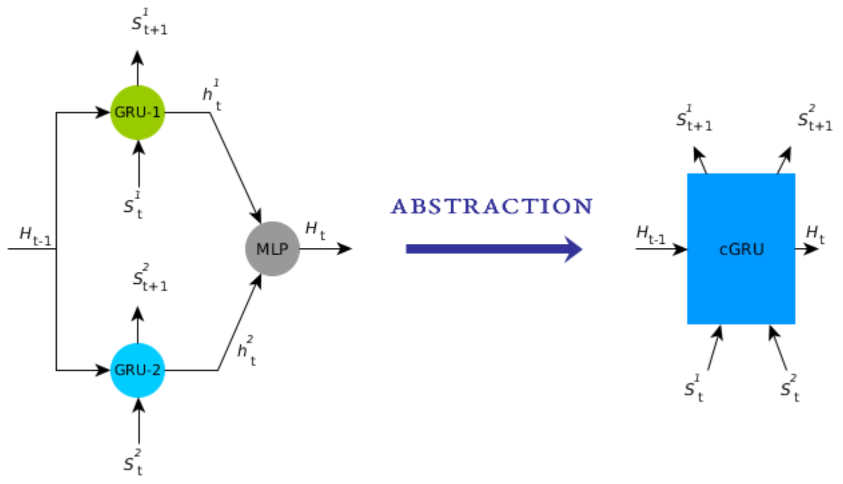


Figure 3.5: cGRU abstraction

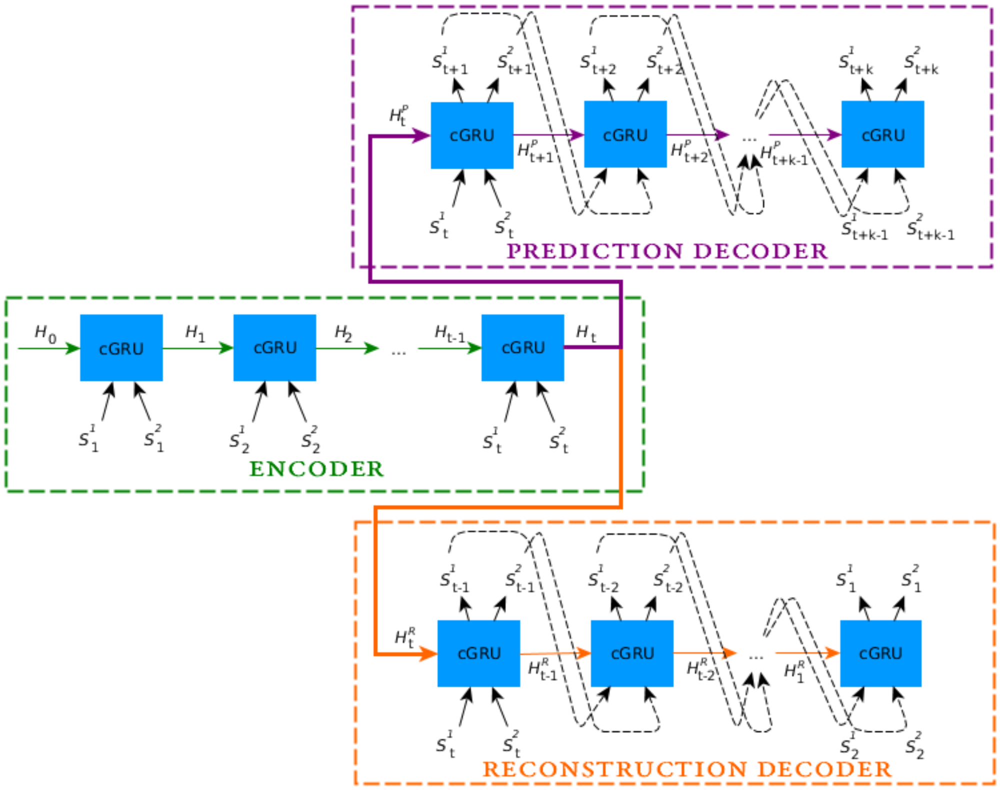


Figure 3.6: cGRU-ED architecture

3.6 Coupled GRU Encoder-Decoder with Attention Mechanism (cGRU-ED-Attention)

This architecture is an extension of cGRU-ED with two self-attention units, one for each decoder. The formulation of the encoder is the same as that of the encoder in cGRU-ED. The components of this architecture are as below,

1. **Encoder:** encodes a pair of sequences of length T into a coupled con-

text vector H_t using,

$$H_t = \text{cGRU}(H_{t-1}, S_t^1, S_t^2)$$

2. **Reconstruction Decoder:** is initialized with the hidden state $H_t^R = H_t$. The self-attention unit takes as input $[H_1, H_2, \dots, H_t]$ and outputs a weighted context vectors c_t^1, c_t^2 using equations below,

$$e_{tj}^* = a(S_t^*, H_j^R)$$
$$\alpha_{tj}^* = \frac{exp(e_{tj}^*)}{\sum_k exp(e_{tk}^*)}$$
$$c_t^* = \sum_j \alpha_{tj}^* H_j^*$$

Replacing $* = 1, 2$ yields equations for the appropriate context vector. This is a slight variation of the formulation by Bahndanau et al. [1], modified to accommodate two input signals and coupling mechanism. Using these weighted context vectors, the decoder reconstructs the input pair in the reverse order as below,

$$H_{t-1}^R, S_{t-1}^1, S_{t-1}^2 = \text{cGRU}(H_t^R, [S_t^1; c_t^1], [S_t^2; c_t^2])$$

3. **Prediction Decoder:** is initialized with the hidden state $H_t^P = H_t$, predicts future pair of length k. The formulation is similar to the

reconstruction decoder, as below,

$$
\begin{aligned}
e^*_{t+k-1,j} &= a(S^*_{t+k-1}, H^P_j) \\
\alpha^*_{t+k-1,j} &= \frac{exp(e^*_{t+k-1,j})}{\sum_k exp(e^*_{t+k-1,k})} \\
c^*_{t+k-1} &= \sum_j \alpha^*_{t+k-1,j} H^*_j \\
H^P_{t+k}, S^1_{t+k}, S^2_{t+k} &= \mathrm{cGRU}(H^P_{t+k-1}, [S^1_{t+k-1}, c^1_{t+k-1}], [S^2_{t+k-1}, c^2_{t+k-1}])
\end{aligned}
$$

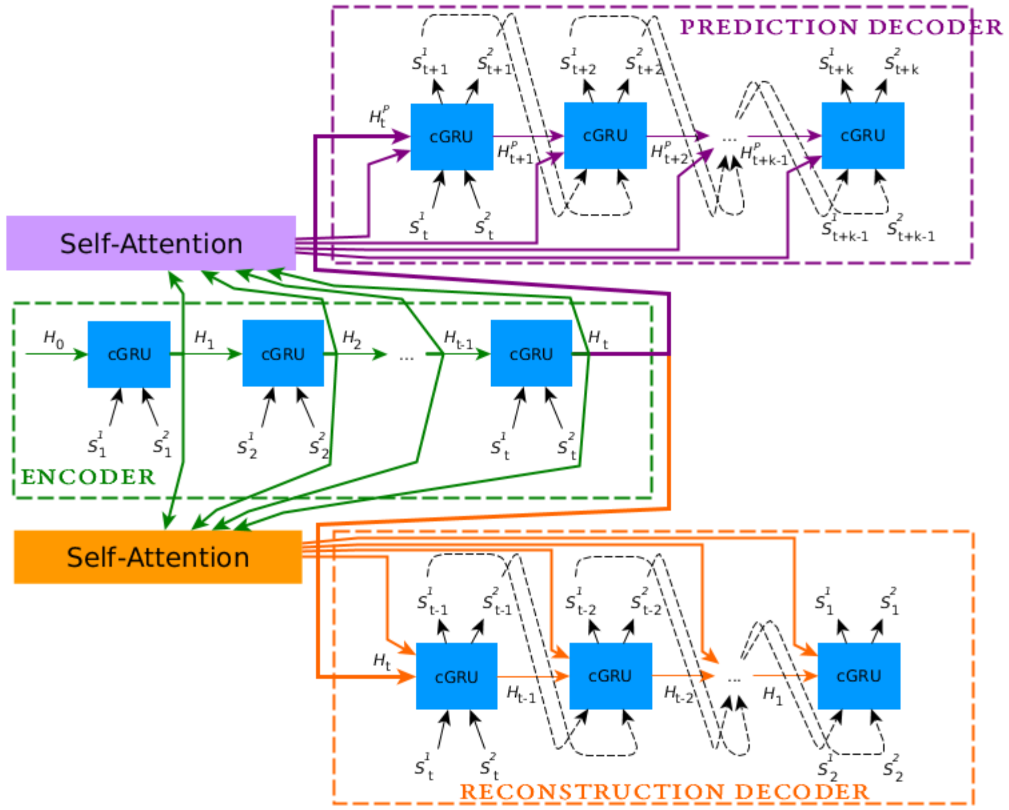

PREDICTION DECODER
Self-Attention
ENCODER
Self-Attention
RECONSTRUCTION DECODER
cGRU
H_0, H_1, H_2, H_{t-1}, H_t
H^P_t, H^P_{t+1}, H^P_{t+2}, H^P_{t+k-1}
S^1_{t+1} S^2_{t+1} S^1_{t+2} S^2_{t+2} S^1_{t+k} S^2_{t+k}
S^1_t S^2_t S^1_{t+k-1} S^2_{t+k-1}
S^1_1 S^2_1 S^1_2 S^2_2
S^1_{t-1} S^2_{t-1} S^1_{t-2} S^2_{t-2}
H_{t-2}

Figure 3.7: cGRU-ED with attention mechanism

3.7 Implementation details

We used following tools and libraries for our implementations,

- python 2.7
- NumPy
- SciPy
- Matplotlib
- PyTorch
- Pandas

The architectures are implemented to utilize GPUs for training, if available. This requires CUDA libraries. The code repository will be made publicly available for further research and exploration.

Chapter 4

Data

In this chapter, we list and describe all the datasets which were used for this project.

4.1 Synthetic data - a pair of coupled Gaussian stochastic processes

This dataset is made up of pairs of coupled Gaussian stochastic processes generated using the algorithm presented by Jamali and Jafari [15]. A pair is generated with a correlation factor in the range $[0, 1]$ and the desired sequence length. The correlation factor is sampled from a uniform distribution. Figure 4.1 shows an example of a pair generated with a correlation factor of 0.5001.

For our experiments, we generated the data and split it into different sets as listed in table 4.1.

This algorithm cannot be used to generate - a pair with different sequence

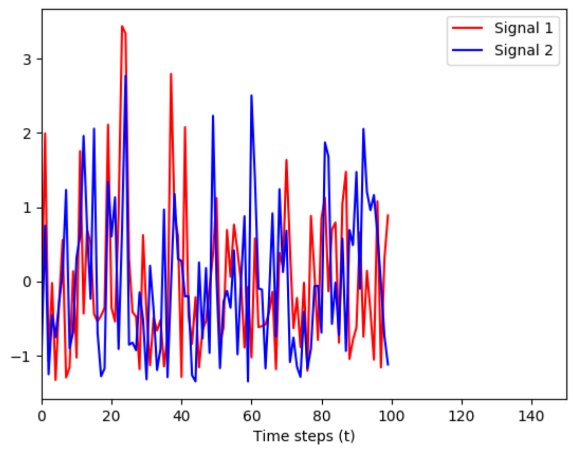


Figure 4.1: A pair of correlated signals with a correlation factor of 0.5001

Sequence length	100
Number of samples	10,000
Train split	8,000
Validation split	1,000
Test split	1,000

Table 4.1: Details of the generated synthetic data

lengths, and multi-variate sequences.

4.2 Synthetic data with delay

Real-life interactions might contain a certain amount of delay. To simulate such scenarios, we incorporate a delay between the pair of signals generated as described in section 4.1. The following steps illustrate the incorporation of delay,

1. Sample an integer d the range [0, 15] as delay from a uniform distribution
2. Generate a pair of sequences (S_1, S_2) of length $100 - d$ with a sampled correlation factor, using Jamali and Jafari's algorithm [15]
3. Pad zeros of length d at the beginning of S_1
4. Pad zeros of length d at the end of S_2

This dataset contains approximately 667 samples (10000/15) without a delay, hence containing a good mix of realistic interactions.

4.3 Rapport data

This data was obtained from a study involved in the larger context of deception detection. The goal of the original study was to determine if rapport aided or hindered deception detection. A pair of interactants that have formed rapport tend to coordinate with each other's body language. However, when a lie is told - a lie that can affect the nature of the relationship - it cannot help but disrupt the flow of their interaction. In fact, research has shown that simply making an accusation of lying toward another creates a break in the harmonious mirroring of behaviors on both liars and truthtellers. The original project thus proposed to explore the role of deception on interactional synchrony and on already established rapport.

For this work, however, we are only interested in detecting whether a pair of interlocutors have or have not established a good rapport between them. Hence, we model the interactions as rapport.

The data collection was conducted at the Communication Science Center (CSC) at the University at Buffalo, SUNY. Each participant in the study completed the consent process and was then presented with a theft scenario. The study participants (interviewees) were asked to steal either a ring or a watch [18, 17] and hide it somewhere in the conference room at the center. In order to increase the stakes of the situation, the interviewees were presented with both incentives and punishment for successful or unsuccessful performance [11, 12]. If successful at convincing the interviewer (blind to the forced choice of stealing one object) that they did not steal either object, they received $30. If they were believed on the object they stole only, they received $20. However, if they were only believed on the object they did not steal, they received $10. If they were not believed on either object, they did not receive any money at all and were asked to write an essay about their unsuccessful performance for the duration of 15 minutes. The interviews took place in an interrogation room at the center, specifically designed to increase the realistic nature of the procedures. Figure 4.2 is an image of a sample video frame showing the interviewing process between a retired police officer and an interviewee. During the interview, both the interviewer and the interviewee were recorded with a separate video camera capturing their frontal view. A third camera captured both the interviewer and interviewee simultaneously in order to preserve spatial information (figure 4.2). Upon completion of the interview, the participants left the interrogation room to complete questionnaires that are used as the ground truth.

To investigate the influence of rapport building, each scenario was created with two between-subject conditions; rapport building or neutral interaction. In the rapport condition, the interviewer spent approximately 3 minutes before and in the middle of questioning, interacting with the participant in a positive manner, shaking his or her hand, smiling and nodding in agreement where applicable, and actively finding something he has in common with the participant. In the neutral interaction condition, the interviewer spent approximately 3 minutes before the questioning about the watch/ring, discussing very neutral topics such as the weather, how the participants arrived at the laboratory, etc., while maintaining a very neutral demeanor. The interviews were performed by three retired police officers who were accustomed to high stakes interviews. A total of fifty-nine (N = 59) interviews from the study were used for experiments in this project.

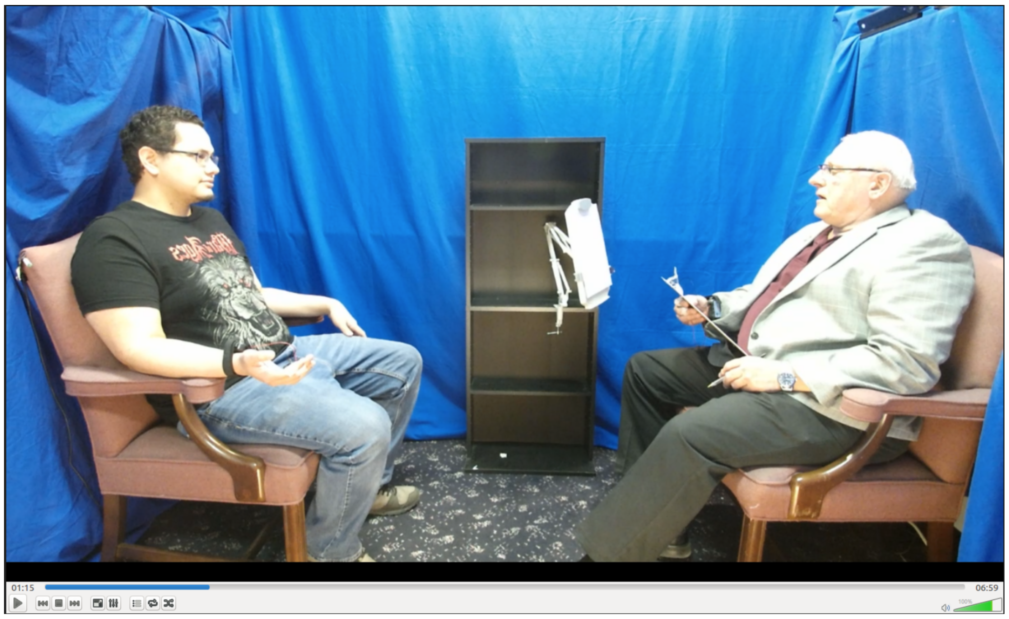


Figure 4.2: Sample video frame showing the interviewing process between an interviewee on the left and a retired police officer on the right

The first three minutes of the interview involved a discussion about topics unrelated to the missing watch or ring. This first period is referred to as *Baseline 1*. The next 3-5 minutes of the interview involved questions about either the ring or the watch; the next 3 minutes were again involving topics unrelated to the experiment items and is referred to as *Baseline 2*. Finally, the last 3-5 minutes of the interview involved more discussions about the missing experiment items again. In this work, we only focus on the conversations that occurred during the Baseline 1 and Baseline 2 time periods, resulting in roughly 7 minutes of video data for each pair of interactants.

The data cleaning, annotation, and feature extraction were done by the researchers at CSC. It included the following steps,

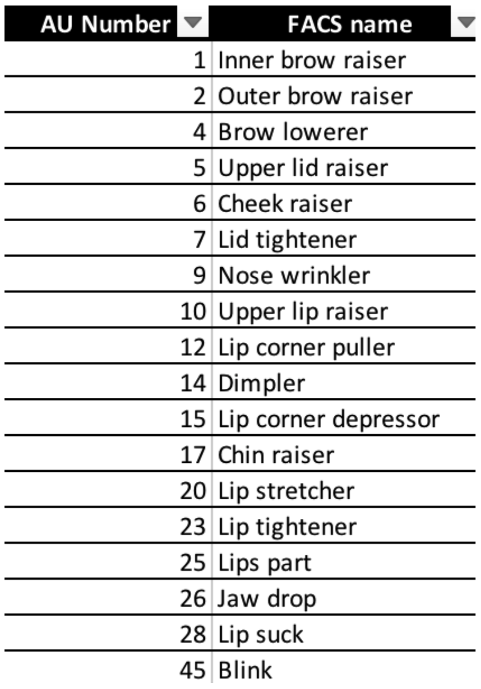
1. synchronize the pair of videos captured from the individual frontal view cameras

2. segment and annotate videos into baselines and critical sections

3. extract segments: Baseline 1 and Baseline 2, and concatenate

4. extract OpenFace features [2] per frame

The data was provided in the form of CSV (comma separated values) files. We use action units (AU) alone as the input features. Action units (AUs) correspond to various muscle groups on the face and can range from being fully activated to not activated [10]. Since the AUs are measured over time, we could now obtain a collection of behavioral signals based on face dynamics. Figure 4.3 shows the codes and descriptions of the AUs detected in the videos. We discarded other features such as head movements (roll/yaw/pitch) and eye gaze directions. The 59 pairs were divided into train: 45, validation: 6 and test: 8.



AU Number	FACS name
1	Inner brow raiser
2	Outer brow raiser
4	Brow lowerer
5	Upper lid raiser
6	Cheek raiser
7	Lid tightener
9	Nose wrinkler
10	Upper lip raiser
12	Lip corner puller
14	Dimpler
15	Lip corner depressor
17	Chin raiser
20	Lip stretcher
23	Lip tightener
25	Lips part
26	Jaw drop
28	Lip suck
45	Blink

Figure 4.3: Description of 18 auction units used for experiments in this project

4.4 Surveillance videos for anomaly detection - ShanghaiTech Campus Dataset

We utilize the ShanghaiTech Campus dataset [20] to build generative models for anomaly detection. This dataset contains footage captured from 13 different cameras around the ShanghaiTech University campus. The authors claim that this is one of the biggest datasets for video anomaly detection.

The dataset contains realistic anomaly events such as chasing and brawling. It includes 13 scenes with 274,515 frames for training and 42,883 for testing and 130 abnormal events in total. Most of the anomaly events in this dataset are related to humans, while some events are not. The authors have removed videos that do not contain human-based anomaly events.

The primary purpose of using this dataset is to compare our proposed architectures to MPED-RNN [21]. The authors use skeleton trajectories as features instead of videos themselves. They state that the skeleton features are compact and highly descriptive about human action and movement, hence they are sufficient for anomaly detection. The authors have published their code on GitHub[1]. This web-page contains a link to the skeleton trajectories data. We use the data they have published as-is to ensure consistency in our comparisons. The authors decompose the skeleton features into local and global features to account for spacial factors and size of the humans in the videos, as below,

They formulate the problem as an interaction between the local and global features. Through their experiments, the authors demonstrate that these signals interact with each other. This dataset serves as a good example for

[1]`skeleton_based_anomaly_detection` codebase: https://github.com/RomeroBarata/skeleton_based_anomaly_detection.

a system other than social interactions, which contains interacting signals.

4.5 ICT dataset

This dataset was collected by Jonathan Gratch from the Institute of Creative Technologies at the University of Southern California [13]. It was created to investigate the importance of *Contingent feedback* in creating feelings of rapport. The authors define contingency as nonverbal feedback by the listener, such as nods or posture shifts that are tightly coupled to what the speaker is doing at the moment. *Non-contingent* feedback is defined as the listener feedback similar in frequency and characteristics to the contingent feedback, but not synchronous with the speaker's speech, behavior, or expressions.

This study was conducted with 161 participants (61% women, 39% men). The authors use a virtual rapport agent alongside face-to-face interactions with human participants. They study two types of virtual characters:

1. **Good virtual listener (Responsive condition):** the rapport agent which synthesizes the head gestures and posture shifts in response to the real human speaker's speech and movements

2. **A virtual representation of a real listener (Mediated condition):** the rapport agent reproduces actual head movements and pos-

ture shifts of a real human listener

This was a between-subjects experiment with four conditions to which the participants were randomly assigned.

1. **Face to Face (20 speakers, 20 listeners):** the speaker talked to a human listener face to face

2. **Mediated (20 speakers, 20 listeners):** The speaker interacted with a virtual character imitating the head movements and posture of a human listener, excluding facial expressions

3. **Responsive (12 speakers):** The speaker interacted with a virtual character displaying proper listening behaviors contingent on the speaker's speech and head movements. Facial expressions were not generated

4. **Non-contingent (12 speakers, 12 listeners):** The speaker interacted with a virtual character whose behavior is identical to the responsive agent, except the behavior was not contingent on the speaker's speech or head movements. The speaker was presented with a pre-recorded behavior sequence in this case.

The participants were told that they would be undergoing a study to evaluate a communicative technology like a web-camera used to chat with friends and family that was developed at the place. The participants were randomly assigned a speaker/listener role. Subjects were made to answer a pre-questionnaire.

The subjects were led into the computer room where experiments were conducted. The procedure was explained and the equipment was introduced. The speaker was tasked to view a short segment of a video clip on sexual harassment awareness, in the absence of the listener. The speaker was instructed to retell the stories in the clip to the listener. He/she sat in front of a 30-inch computer monitor in the cases - 2, 3, and 4, and interacted with an animated character displayed on the screen. The speakers were told that the avatar on the screen represents the human listener. The listener, on the other hand, was shown a real-time video of the speaker retelling the story. The speaker completed a post-questionnaire in the absence of the listener and assigned a score using the provided rapport scale. The speech and video data were captured and published for research purposes [13]. We changed the rapport scale to binary labels. The scores in the range [1, 6] were assigned the label *rapport* as they corresponded to a certain level of rapport according to the scale. The rest of the values were assigned with the label *neutral*.

Although the study resulted in 124 interactions, just 31 of them have recorded videos available for both speaker and listener. Of the 31, we discarded 8 of them as they did not have rapport scores available. This left us 23 videos to work with. Since these were insufficient to train and test our models, we used these videos as test data for the models trained on rapport data (section 4.3).

4.6 Data preprocessing: rapport, ICT

Both rapport and ICT datasets are multivariate with 18 features (Action Units). The rapport videos were recorded at 30 frames per second, so each 7-minute video would have around 12,600 frames. These sequences are very long for a temporal model. To overcome this potential problem, each video was divided into non-overlapping chunks of 100 frames. All the chunks of a video were assigned with the same label.

Chapter 5

Experiments and results

In this chapter, we list and describe the experiments we conducted on each dataset using all the models and describe results. We use the following loss functions to train and evaluate all the architectures,

- **Mean Absolute Error (MSE)**:

  $$\frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|$$

  where,

  N is the number of data samples

  y_i is the actual value

  $\hat{y}_i$ is the predicted value

- **Mean Squared Error (MAE)**:

  $$\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

- **Classification accuracy**:

$$\frac{TP + TN}{TP + TN + FP + FN}$$

Measure	**Description**	**Predicted label**	**Actual label**
TP	true positives	1	1
FP	false positives	1	0
TN	true negatives	0	0
FN	false negatives	0	1

This metric is used to calculate the binary classification accuracy.

5.1 Synthetic data

We used this dataset for hyper-parameter tuning. We conducted three experiments on this dataset. The following sections describe the experiments in detail. The models were trained on the train split. The validation split was used to choose the best model. The best model was tested on the test split.

5.1.1 Hyper-parameter tuning

In this experiment, we tried different combinations of hyper-parameter values for cGRU and chose the one with lowest MAE on the test data. Table 5.1 lists the cGRU hyper-parameters which were optimized. Figure 5.1 shows training and validation errors (MAE) of the cGRU model with hyper-parameters corresponding to the lowest MAE.

Hyper-parameter	Description	Optimal value
Learning rate (LR)	rate of weight updates	0.001
Hidden size	number of hidden units in GRU	64
Epochs	number of training steps	50

Table 5.1: Hyper-parameters for cGRU and optimized values

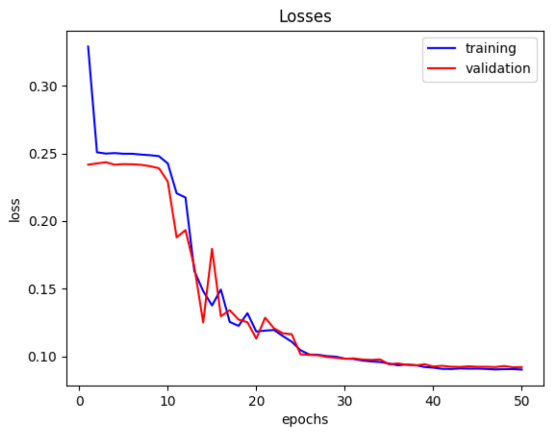


Figure 5.1: cGRU training and validation errors for each epoch. The model was trained using optimized hyper-parameter values in Table 5.1

We continue to use these hyper-parameter values for experiments on other datasets. We also considered end2end-GRU for this step. However, end2end-GRU could not learn with any combination of hyper-parameters. The model with higher number of epochs resulted in over-fitting (figure 5.2).

5.1.2 Baseline: end2end-GRU versus cGRU

In this experiment, we justify the hypotheses 1 and. We model interactions by coupling hidden states of the recurrent units (section 3.2). We trained

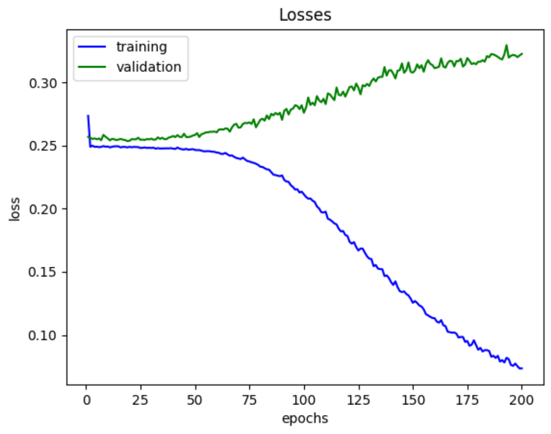


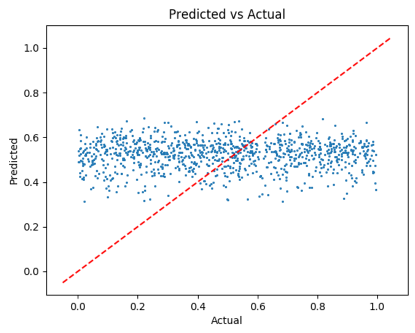
Figure 5.2: end2end-GRU training and validation errors for each epoch. The model was trained using - LR: 0.01, hidden size: 64, epochs: 200

end2end-GRU and cGRU on the data using hyper-parameters values in table 5.1 for a fair comparison. A plot of predicted vs actual values of correlation factors was created for both the models. In this case, the more the number of points closer to the diagonal, the better the model. Figure 5.3 shows that cGRU has more points closer to the diagonal line.

This shows that cGRU can learn better than end2end-GRU, which validates hypotheses 1, and 2. Hence, modeling interactions are crucial for supervised learning tasks.

5.1.3 Error variance of models

For this experiment, we run 100 trials with different train/test/validation splits for each trial. The purpose of this experiment is to compare the per-



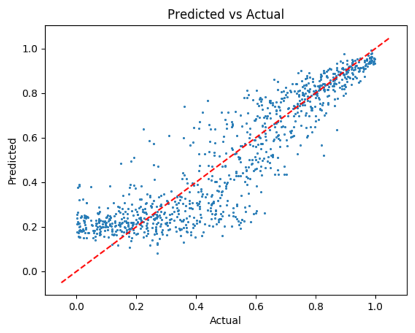


Figure 5.3: Predicted vs actual correlation factors for end2end-GRU (left side) and cGRU (right side)

formance and stability of our proposed models: cGRU, cGRU-Attention with MP-RNN. We use the same set of hyper-parameters from table 5.1 for all the three models. Additionally, we use message size as 16 for MP-RNN. We used the same train/test/validation for all the models in each trial to ensure a fair comparison.

Figure 5.4 shows a plot of test errors over 100 trials, for all the models. The table 5.2 lists the mean errors (MAE) with hyper-parameter values. The performance of cGRU is relatively unstable as we see a few spikes in test errors. This validates hypotheses 3, and 4. MP-RNN consistently outperforms cGRU and cGRU-Attention. This can be attributed to the following factors,

- The synthetic data is uni-variate
- The MP-RNN architecture is relatively complex with two message sig-

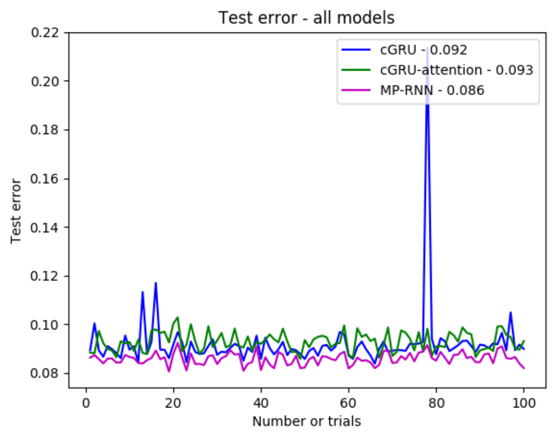


Figure 5.4: Test errors (MAE) for 100 trials on cGRU, cGRU-Attention and MP-RNN with different train/test/validation splits for each trial

nals

Since cGRU and cGRU-Attention are relatively simple and compact with a single coupling unit, they are computationally cheaper. These can be easily scaled to multiple signals. We can further increase layers in the MLP to capture complex non-linear interactions. Whereas MP-RNN cannot be easily scaled for multiple signals.

Architecture	cGRU	cGRU-Attention	MP-RNN
Hidden units	64	64	64(message:16)
Epochs	50	50	50
Learning rate	10^{-3}	10^{-3}	10^{-3}
Mean MAE (100 trials)	0.092	0.093	0.086

Table 5.2: 100 trials on synthetic data with different train/test/validation splits for each trial

5.2 Synthetic data with delay

In this experiment, we train our models on the dataset with delay, generated as described in section 4.2. The purpose of this experiment is to test how well do these models pick-up delayed interactions. We ran an experiment of 25 trials on all the models with this data (figure 5.5). Table 5.3 lists the results with hyper-parameter configurations.

Architecture	cGRU	cGRU-Attention	MP-RNN
Hidden units	64	64	64(message:16)
Epochs	50	50	50
Learning rate	10^{-3}	10^{-3}	10^{-3}
Mean MAE (25 trials)	0.091	0.092	0.086

Table 5.3: 25 trials on synthetic with different train/test/validation splits for each trial

The results show that all the models can learn to capture delayed interactions.

5.3 Rapport data

We used same hyper-parameter configuration listed in table 5.1. Since the number of data samples were huge due to the pre-processing step (section 4.6), we ran 10 trials with different train/test/validation splits on all three models. The results are in the table 5.4. In this case, our proposed architecture **cGRU-Attention** has the best average accuracy over 10 trials. This

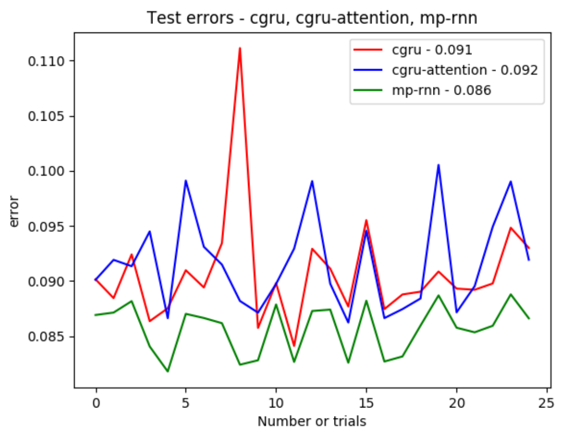


Figure 5.5: Test errors (MAE) for 25 trials on synthetic data with delay, with different train/test/validation splits for each trial

further validates hypotheses 3 and 4.

Architecture	cGRU	cGRU-Attention	MP-RNN
Hidden units	64	64	64(message:16)
Epochs	50	50	50
Learning rate	10^{-3}	10^{-3}	10^{-3}
Mean accuracy (10 trials)	67.5%	77.5%	71.25%

Table 5.4: 10 trials on rapport data with different train/test/validation splits for each trial

5.4 ICT data

For this experiment, we used the models trained on rapport data to test on ICT data. Table 5.5 lists the results on all the models. Although numbers are marginally above chance, they indicate that the models generalize well to unseen data. In addition, this data was recorded in different conditions for a different purpose. Our model - cGRU-Attention continues to perform better.

Architecture	cGRU	cGRU-Attention	MP-RNN
Accuracy	56.52%	65.22%	60.87%

Table 5.5: Results on ICT data using models trained on Rapport data

5.5 Anomaly detection in videos - ShanghaiTech dataset

The purpose of this experiment is to compare the generative versions of architectures we propose to MPED-RNN [21] and test the capability of the coupling mechanism for generative tasks. We implement the architectures, cGRU-ED and cGRU-ED-Attention, extensions of cGRU and cGRU-Attention. MSE is used as the error function to calculate losses for each individual signal. These architectures are jointly trained to minimize reconstruction loss and prediction loss, as below,

$$L_{total} = L_{reconstruction} + L_{prediction} \tag{5.1}$$

The MPED-RNN was implemented in Keras and TensorFlow [21]. We implemented a PyTorch version of MPED-RNN. We used the pre-processing and post-processing snippets from their codebase as-is. The authors use frame-level ROC AUC to evaluate the anomaly detection in the videos. We couldn't achieve the numbers claimed in the paper as we didn't have access to the hyper-parameters used by the authors. We also saw a drop in ROC AUC of our PyTorch implementation of MPED-RNN compared to their Keras and TensorFlow implementation.

Architecture	cGRU-ED	cGRU-ED-Attention	MPED-RNN
ROC AUC	0.6143	0.6239	0.6179

Table 5.6: Results on ShanghaiTech dataset

The results are in table 5.6. cGRU-ED-Attention continues to perform better for generative tasks as well.

Chapter 6

Conclusion

6.1 Conclusion

In this project, we explored methods to model interacting time-series signals. In particular, we were interested in modeling social interactions. We presented traditional methods and discussed why they would fall short for modeling multivariate data such as videos. Inspired by similar studies in the literature, we proposed architectures; cGRU and cGRU-Attention, with coupling mechanism and self-attention to capture interactions in the data. We used algorithms to synthetically generate uni-variate data for our experiments. To simulate realistic conditions, we incorporated delays. This data was used to tune the hyper-parameters of models. We obtained rapport and ICT datasets to evaluate our models on real-life data, both collected to study rapport between individuals.

We ran multiple trials on all the datasets to understand the performance stability of all the models. In our experiments, MP-RNN, a version of MPED-RNN [21] performed well on the synthetic data. We attribute this to the uni-variate nature of the dataset and the complexity of MP-RNN. However, our proposed architecture, cGRU-Attention, performed better on rapport and ICT datasets. Our studies indicate that these architectures are suitable for modeling social interactions. The compact formulation coupled with scalability to multiple signals makes our proposed models desirable to model complex systems. The performance on real-life data further validates this claim.

6.2 Future Work

In this study, we were successful in modeling interacting sequences of the same length. The results of our experiments are promising. We would like to continue this study and develop architectures to model sequences of varying lengths. This would allow us to model problems like sentence similarity. Although the rapport and ICT datasets were audio-visual in nature, we studied rapport using visual features alone. We would like to extend our architectures to perform a multi-modal analysis. This would allow us to scale our models to multiple signals and evaluate the scalability aspect.

Bibliography

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

[2] Tadas Baltrusaitis, Peter Robinson, and Louis-Philippe Morency. Constrained local neural fields for robust facial landmark detection in the wild. In *Proceedings of the IEEE international conference on computer vision workshops*, pages 354–361, 2013.

[3] Jacob Benesty, Jingdong Chen, Yiteng Huang, and Israel Cohen. Pearson correlation coefficient. In *Noise Reduction in Speech Processing*, pages 1–4. Springer, 2009.

[4] Yoshua Bengio, Patrice Simard, Paolo Frasconi, et al. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

[5] Donald J Berndt and James Clifford. Using dynamic time warping to find patterns in time series. In *Knowledge Discovery and Data Mining workshop*, volume 10, pages 359–370. Seattle, WA, 1994.

[6] Matthew Brand, Nuria Oliver, and Alex Pentland. Coupled hidden markov models for complex action recognition. In *Computer Vision and Pattern Recognition (CVPR)*, volume 97, page 994, 1997.

[7] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[8] 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.

[9] Aimee L Drolet and Michael W Morris. Rapport in conflict resolution: Accounting for how face-to-face contact fosters mutual cooperation in mixed-motive conflicts. *Journal of Experimental Social Psychology*, 36(1):26–50, 2000.

[10] Rosenberg Ekman. *What the face reveals: Basic and applied studies of spontaneous expression using the Facial Action Coding System (FACS).* Oxford University Press, USA, 1997.

[11] Mark G Frank and Paul Ekman. The ability to detect deceit generalizes across different types of high-stake lies. *Journal of personality and social psychology*, 72(6):1429, 1997.

[12] Mark G Frank and Paul Ekman. Appearing truthful generalizes across different deception situations. *Journal of personality and social psychology*, 86(3):486, 2004.

[13] Jonathan Gratch, Ning Wang, Jillian Gerten, Edward Fast, and Robin Duffy. Creating rapport with virtual agents. In *International workshop on intelligent virtual agents*, pages 125–138. Springer, 2007.

[14] Alex Graves, Santiago Fernández, and Jürgen Schmidhuber. Multi-dimensional recurrent neural networks. In *International Conference on Artificial Neural Networks*, pages 549–558. Springer, 2007.

[15] Tayeb Jamali and GR Jafari. Method for generating two coupled gaussian stochastic processes. *arXiv preprint arXiv:1602.04697*, 2016.

[16] Nal Kalchbrenner, Ivo Danihelka, and Alex Graves. Grid long short-term memory. *arXiv preprint arXiv:1507.01526*, 2015.

[17] John C Kircher and David C Raskin. Human versus computerized evaluations of polygraph data in a laboratory setting. *Journal of Applied Psychology*, 73(2):291, 1988.

[18] F Andrew Kozel, Kevin A Johnson, Qiwen Mu, Emily L Grenesko, Steven J Laken, and Mark S George. Detecting deception using functional magnetic resonance imaging. *Biological psychiatry*, 58(8):605–613, 2005.

[19] Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. Modelling interaction of sentence pair with coupled-lstms. *arXiv preprint arXiv:1605.05573*, 2016.

[20] Weixin Luo, Wen Liu, and Shenghua Gao. A revisit of sparse coding based anomaly detection in stacked rnn framework. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 341–349, 2017.

[21] Romero Morais, Vuong Le, Truyen Tran, Budhaditya Saha, Moussa Mansour, and Svetha Venkatesh. Learning regularity in skeleton trajec-

tories for anomaly detection in videos. *arXiv preprint arXiv:1903.03295*, 2019.

[22] Iead Rezek, Peter Sykacek, and Stephen J Roberts. Learning interaction dynamics with coupled hidden markov models. *IEE Proceedings-Science, Measurement and Technology*, 147(6):345–350, 2000.

[23] David E Rumelhart, Geoffrey E Hinton, Ronald J Williams, et al. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1, 1988.

[24] Lin Sun, Kui Jia, Yuejia Shen, Silvio Savarese, Dit Yan Yeung, and Bertram E Shi. Coupled recurrent network (crn). *arXiv preprint arXiv:1812.10071*, 2018.

[25] Linda Tickle-Degnen and Robert Rosenthal. The nature of rapport and its nonverbal correlates. *Psychological inquiry*, 1(4):285–293, 1990.

[26] Philip Tsui and Gail L Schultz. Failure of rapport: Why psychotherapeutic engagement fails in the treatment of asian clients. *American Journal of Orthopsychiatry*, 55(4):561–569, 1985.

[27] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention

is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

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