ABSTRACT

Title of thesis:	Estimation of the Temporal Response Function and Tracking Selective Auditory Attention using Deep Kalman Filter
	Yexin Cao Master of Science, 2020
Thesis directed by:	Professor Babadi Behtash Department of Electrical and Computer Engineering

The cocktail party effect refers to the phenomenon that people can focus on a single sound source in a noisy environment with multiple speakers talking at the same time. This effect reflects the human brain's ability of selective auditory attention, whose decoding from non-invasive electroencephalogram (EEG) or magnetoencephalography (MEG) has recently been a topic of active research. The mapping between auditory stimuli and their neural responses can be measured by the auditory temporal response functions (TRF). It has been shown that the TRF estimates derived with the envelopes of speech streams and auditory neural responses can be used to make predictions that discriminate between attended and unattended speakers. l_1 regularized least squares estimation has been adopted in previous research for the estimation of the linear TRF model. However, most real-world systems exhibit a degree of non-linearity. We thus have to use new models for complex, realistic auditory environments. In this thesis, we proposed to estimate TRFs with the deep Kalman filter model, for the cases where the observations are a noisy, non-linear

function of the latent states. The deep Kalman filter (DKF) algorithm is developed by referring to the techniques in variational inference. Replacing all the linear transformations in the classic Kalman filter model with non-linear transformations makes the posterior distribution intractable to compute due to the non-linearity. Thus, a recognition network is introduced to approximate the intractable posterior and optimize the variational lower bound of the objective function. We implemented the deep Kalman filter model with a two-layer Bidirectional LSTM and a MLP. The performance is first evaluated by applying our algorithm to simulated MEG data. In addition, we also combined the new model for TRF estimation with a previously proposed framework by replacing the dynamic encoding/decoding module in the framework with a deep Kalman filter to conduct real-time tracking of selective auditory attention. This performance is validated by applying the general framework to simulated EEG data.

Estimation of the Temporal Response Function and Tracking Selective Auditory Attention using Deep Kalman Filter

by

Yexin Cao

Thesis submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Master of Science
2020

Table of Contents

Chapter 1: Introduction

1.1 Cocktail Party Effect

The cocktail party effect is a phenomenon that a person focuses his/her auditory attention on a particular sound source and filters out all other stimuli, just like people focusing on a single conversation on a noisy cocktail party [1]. This effect was first defined and named the cocktail party problem by Colin Cherry in 1953, who conducted the dichotic listening task. In Cherry's attention experiments, the participants had to separate two different messages they heard from a loudspeaker in each ear. According to the cocktail party effect, people are able to segregate multiple stimuli into different streams, and subsequently decide which streams are most pertinent to them. This ability to identify a specific source amid sounds emanating from other sources, is actually an essential function of human brain. Because of the importance of selective attention, the mechanisms underlying the real-time process of target tracking have been a topic of interest for a long time, although most of the underlying process are still unknown.

From a neuroscience perspective, the central auditory system has to perceptually segregate and group the acoustic input into sequences of distinct auditory objects [2]. Although the target stream and interfering streams are processed in the same pathway within the left hemisphere, fMRI results show that target streams are treated with more attention than competing streams [3]. As the acoustic signals propagate through the auditory pathway, they are decomposed into spectrotemporal features at different stages, and a rich representation of the complex auditory environment reaches the auditory cortex. It has been hypothesized that the perception of an auditory object is the result of adaptive binding as well as discounting of these features [4]. There are various hypotheses and models with regarding to the neural underpinnings of perceptual organization in the central auditory system, especially the auditory cortex. For example, one of the popular hypotheses, known as the "population-separation hypothesis", states that sound elements segregate into separate "streams" whenever they activate well-separated populations of auditory neurons that are selective to frequency or any other sound attributes that have been shown to support stream segregation [5]. Another influential hypothesis is that streams are formed automatically or pre-attentively, in or below the primary auditory cortex [5].

From a computational modeling perspective, researchers have tried to design several different kinds of attention decoders, so as to reliably decode the attentional focus of a listener in a multi-speaker environment using non-invasive neuroimaging techniques like electroencephalography (EEG) and magnetoencephalography (MEG) [4]. The previous approaches have two major problems, while most of them are able to reliably decode the attentional focus. Since these methods are typically based on linear regression, which requires large datasets for training, the application of those attention decoders to real world is limited. In addition, the decoding accuracy drops significantly when operating at temporal resolutions that humans are able to switch attention from one speaker to another. Therefore, we aim at using an alternative approach that overcomes the aforementioned limitations.

1.2 Temporal Response Function

Temporal Response Function (TRF) describes a mapping between some features of a sensory stimulus and the auditory neural response. There are other mapping functions between the stimuli and the neural response, such as the event related potentials (ERPs). While ERPs require many repetitions of the same stimulus to be computed, the TRFs can be computed using continuous stimuli such as speech. Previous research has proved that if the TRF estimates are derived with the envelopes of speech streams and auditory neural responses, the TRF-driven predictions can be used to determine which speaker is attended in a multi-speaker scenario. Therefore, we will focus on the estimation of TRF in order to keep track of the selective auditory attention.

One important thing to notice is that the temporal response function can be described as a sparse kernel. Therefore, we model the TRF over a Gaussian dictionary with time-varying coefficients, where the coefficients are assumed to be sparse [6].

1.3 Kalman Filter

Kalman filter, one of the most influential algorithms for tracking time-varying phenomena, estimates unknown states given observations over time. In classical Kalman filters, the latent state evolution, the emission distribution, and the transition functions are all modelled as linear functions perturbed by Gaussian noise [7]. The classical Kalman filter has the following state space model given the observation sequence $x_1, ..., x_T$:

$$\begin{aligned} z_t &= G_t z_{t-1} + B_t u_{t-1} + \epsilon_t \ (action - transition) \\ x_t &= F_t z_t + \eta_t \ (Observations) \end{aligned} \tag{1.1}$$

where $\epsilon_t \sim \mathcal{N}(0, \Sigma_t), \eta_t \sim \mathcal{N}(0, \Sigma_t)$ are zero-mean i.i.d. normal random variables, with covariance matrices which may vary with t. As stated above, this classic Kalman Filter model assumes linear latent space evolution, treats the control signal u_t as linear transformation of the latent state, and generates the observations linearly from the latent state via the observation matrix [7].

However, the linear transition and emission distribution do not apply to the complicated real world applications. Since the non-linearities make learning much more challenging, the researchers proposed multiple modifications to the functional form of Kalman filters to make it non-linear, including the Extended Kalman Filter, the Unscented Kalman Filter, and the Deep Kalman Filter used in this thesis [7]. In the non-linear situation, the posterior distribution $p(z_1, ..., z_T | x_1, ..., x_T, u_1, ..., u_T)$

becomes intractable to compute. Thus, the techniques in variational inference, an approach to approximate Bayesian posterior inference, are adopted in deep kalman filter model. The variational Bayes approximates a full posterior distribution with a factorized set of distributions by maximizing a lower bound on the marginal likelihood of the variational objective function, which is equivalent to minimizing the Kullback-Leibler divergence between the true posterior and a predefined factorized distribution on the same variables [8]. The most important innovation in the deep kalman filter model is to introduce a recognition network that could approximate the intractable posterior [7].

1.4 General Framework

The general framework we used in this thesis is the same as what used in *Real-Time Tracking of Selective Auditory Attention From M/EEG: A Bayesian Filtering Approach* [4], which contains three main modules, i.e., dynamic encoder/decoder estimation, attention marker extraction, and the real-time state space estimator. The dynamic encoder/decoder estimation module is used to estimate dynamic encoding/decoding models that could be fitted to the neural data in real-time. In this part, we utilizes the deep kalman filter model, which fit a generative model to a sequence of observations and actions. In this model, we suppose that the observations are a noisy, non-linear function of the latent state which evolves over time. We also assume that we can observe the actions that may affect the latent state in a possibly non-linear manner [7]. The attention marker extraction module, as we can see from its name, is employed to extract attention marker features that are functions of the M/EEG recordings, the estimated encoding/decoding coefficients, and the auditory stimuli. We have to choose the attention marker features that could help us separate the contributions of attended and unattended speakers in the neural response. Those extracted attention marker features would be passed into the last module, namely, the real-time state space estimator. Based on Bayesian fixed-lag smoothing, the state-space estimator operates with controllable delay and translates the attention marker features into probabilistic, robust, and dynamic measures that could be used in real-time application.

Although the general framework are the same, the difference between this work and the one from *Miran et.al* [4] is that we employ the deep Kalman filter, a probabilistic generative non-linear model, instead of the linear encoding and decoding models used in previous paper for the first module. *Miran et.al* [4] utilizes the forgetting factor mechanism of the Recursive Least Squares (RLS) algorithm together with the l_1 regularization penalty from Lasso to capture the dynamics in the data while preventing overfitting, and the real-time inference is then efficiently carried out using a Forward-Backward Splitting (FBS) procedure.[4] RLS is an adaptive filter algorithm that recursively finds the coefficients that minimize a weighted linear least squares cost function relating to the input signals. The final encoding/decoding coefficients in encoding context, also known as Temporal Response Function (TRF)s, are estimated dynamically using the RLS algorithm with the neural response and envelopes of speech as inputs. However, as a special case of the classic Kalman Filter model, the RLS algorithm is not optimal in complicated real world applications.

Therefore, in this thesis, we produce the time-varying estimates of TRFs using the deep Kalman filter model, which employing deep neural networks as building blocks. With the deep neural networks, the Kalman filter model can account for complex transition dynamics and emission distributions, which can be used to model the real world problem.

Chapter 2: Deep Kalman Filter

2.1 Overview

As stated in previous chapter, in classical Kalman filter models, the latent state is assumed to evolve linearly and the relationship between the latent space, observed space and actions are expressed in the form of a linear dynamical system. For real world applications, we have to replace linear transformations with non-linear transformations, which largely increases the complexity of the problem as the posterior distribution $p(z_1, ..., z_T|x_1, ..., x_T, u_1, ..., u_T)$ becomes intractable to compute. In order to approximate this intractable posterior, we referred to the variational encoder [9][10] to optimize a variational lower bound on the marginal likelihood of the variational objective function [7].

In section 2.2, we will give a overlook of the whole deep Kalman filter model, including the model setup and related equations. Then in section 2.3, we will demonstrate the learning process, i.e., the optimization of the lower bound of the marginal log-liklihood, using the stochastic backpropagation. In this section, we will first derive the general equations and algorithm, then illustrate the detailed example of variational autoencoder, with the specific technique used, i.e., the reparameterization trick.

2.2 Deep Kalman Filter Model

In this case, we assume the observation sequence $\vec{x} = (x_1, ..., x_T)$ is a nonlinear function of the corresponding latent state $\vec{z} = (z_1, ..., z_T)$, where the latent state itself evolves over time. Also define the actions to be $\vec{u} = (u_1, ..., u_T)$. Here, $x_t \in \mathbb{R}^d, u_t \in \mathbb{R}^c, z_t \in \mathbb{R}^s$. Then the generative model is:

$$\begin{aligned} z_1 &\sim \mathcal{N}(\mu_0; \Sigma_0) \\ z_t &\sim \mathcal{N}(G_\alpha(z_{t-1}, u_{t-1}, \Delta_t); S_\beta(z_{t-1}, u_{t-1}, \Delta_t)) \\ x_t &\sim \Pi(F_\kappa(z_t)) \end{aligned} \tag{2.1}$$

That is to say, we assume that the latent state z_t has a Gaussian distribution, whose mean and variance are nonlinear functions of the previous latent state z_{t-1}, the previous actions u_{t-1}, and the time difference Δ_t. The observations x_t is correlated to the latent state z_t since its distribution depends on the distribution Π, whose parameters are a function of the corresponding latent state z_t. In addition, we set $\mu_0 = 0, \Sigma_0 = I_d$, so the parameters of this generative model are $\theta = \{\alpha, \beta, \kappa\}$. [7]

Since the functions $G_\alpha, S_\beta, F_\kappa$ could be of any form, the Equation 2.1 actually includes a large family of latent space models. In other words, we could train various kinds of Kalman filters using this general equation, as long as we modify the functional forms of $G_\alpha, S_\beta, F_\kappa$. Moreover, S_β should be a diagonal covariance matrix, and we have to make sure it is positive definite by log-parameterization. To better understand this generative model, we can take a look at the classic Kalman filter model. Compared to Equation 1.1, it's easy to see that $G_\alpha(z_{t-1}, u_{t-1}, \Delta_t) = G_t z_{t-1}, S_\beta(z_{t-1}, u_{t-1}, \Delta_t) = B_t u_{t-1}, F_\kappa(z_t) = F_t z_t$ in the case of classic Kalman filter. In more complicated situation, i.e., when $G_\alpha, S_\beta, F_\kappa$ are nonlinear, all these function are parameterized by deep neural networks which will be explained in detail later.

Figure 2.1 shows the learning process for the deep Kalman filter. The solid lines here denote the generative model $p_0(z)p_\theta(x|z)$, the dashed lines denote the variational approximation $q_\phi(z|x)$ to the intractable posterior $p(z|x)$. $q_\phi(\vec{z}|\vec{x}, \vec{u})$ is the parametric approximation to $p_\theta(\vec{x}|\vec{z}, \vec{u})$.

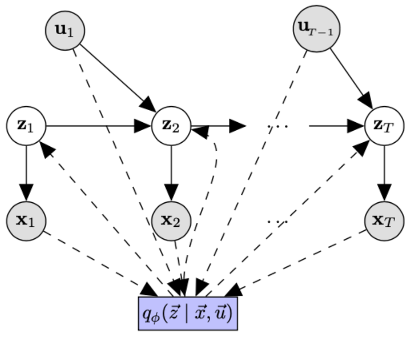


Figure 2.1: Deep Kalman Filter structure (figure from [1])

2.3 Stochastic Backpropagation

Given the generative equation stated in previous section, the core problem is to compute the intractable posterior inference, i.e., $p(z_1, ..., z_T|x_1, ..., x_T, u_1, ..., u_T)$, where we employ the variational inference technique. The variational Bayes approximates the posterior distribution by attempting to optimize a variational lower bound on the marginal log-liklihood of the observation $\vec{x}$, which is completed by a recognition neural network.

Let $p(x, z) = p_0(z)p_\theta(x|z)$, where $p_0(z)$ is the prior on z, and $p_\theta(x|z)$ is a generative model parameterized by θ. This equation is a generative model for the observations x, whose posterior distribution $p_\theta(x|z)$ is typically intractable. Therefore, according to the variational inference techniques, we have to introduce a new distribution $q_\phi(z|x)$ to approximate the actual posterior distribution. We can derive the lower bound on the marginal likelihood as following (using Jensen's inequality):

$$\begin{aligned} log\ p_\theta(x) &= log\ \int_z \frac{q_\phi(z|x)}{q_\phi(z|x)} p_\theta(x|z)p_0(z)dz \\ &\geq \int_z q_\phi(z|x) log \frac{p_\theta(x|z)p_0(z)}{q_\phi(z|x)})dz \\ &= \mathbb{E}_{q_\phi(z|x)}[log\ p_\theta(x|z)] - KL(q_\phi(z|x)\|p_0(z)) \\ &= \mathcal{L}(x; (\theta, \phi)) \end{aligned} \tag{2.2}$$

In the implementation, $q_\phi(z|x)$ will be parameterized by a neural network so that ϕ is the parameter of this network. Equation 2.2 is difficult to calculate directly due to two reasons. First, the expectation term $\mathbb{E}_{q_\phi(z|x)}[log\ p_\theta(x|z)]$ is unknown in most situation. Second, there is an indirectly dependency on the parameter of the neural network ϕ. The way to solve this problem is the stochastic backpropagation [9].

If we assume the latent state to be a K-dimensional Gaussian distribution, i.e., $q_\phi \sim \mathcal{N}(\boldsymbol{\mu}_\phi(\boldsymbol{\xi}), \boldsymbol{\Sigma}_\phi(\boldsymbol{\xi}))$, the required gradients of the expectation term can be computed using the Gaussian gradient identities:

$$\begin{aligned}\nabla_{\mu_i}\mathbb{E}_{\mathcal{N}(\boldsymbol{\mu},\boldsymbol{\Sigma})}[f(\boldsymbol{\xi})] &= \mathbb{E}_{\mathcal{N}(\boldsymbol{\mu},\boldsymbol{\Sigma})}[\nabla_{\xi_i}f(\boldsymbol{\xi})] \\ \nabla_{\Sigma_{ij}}\mathbb{E}_{\mathcal{N}(\boldsymbol{\mu},\boldsymbol{\Sigma})}[f(\boldsymbol{\xi})] &= \frac{1}{2}\mathbb{E}_{\mathcal{N}(\boldsymbol{\mu},\boldsymbol{\Sigma})}[\nabla^2_{\xi_i,\xi_j}f(\boldsymbol{\xi})]\end{aligned}$$

To be more specific, the above identity is derived as following:

$$\begin{aligned}\nabla_\phi E_q[f(\theta)] &= \nabla_\phi \int_\theta f(\theta)q(\theta|\phi)d\theta \\ &= \int_\theta f(\theta)\nabla_\phi q(\theta|\phi)d\theta = \int_\theta f(\theta)q(\theta|\phi)\nabla_\phi log\ q(\theta|\phi)d\theta \\ &= E_q[f(\theta)\nabla_\phi log\ q(\theta|\phi)]\end{aligned}$$

Then employing Monte Carlo integration, we can further compute:

$$\nabla_\phi E_{q_\phi(\boldsymbol{\xi})}[f(\boldsymbol{\xi})] \approx \frac{1}{L}\sum_{l=1}^{L} f(\boldsymbol{\xi}^{(l)})\nabla_{q_\phi(\boldsymbol{\xi}^{(l)})} log\ q_\phi(\boldsymbol{\xi}^{(l)})$$

where L is the number of samples used to approximate the expectation. Similarly, the second term (KL divergence) in Equation 2.2 can be estimated in the same way since the KL divergence is also an expectation.

However, the lower bound in Equation 2.2 only works for simple transition models and has a high variance when estimating the gradient of the KL term. Therefore, we have to extend the original equation and factorize the KL term in a new way to achieve more stable gradients.

$$\begin{aligned} log\ p_\theta(\vec{x}|\vec{u}) &\geq \int_{\vec{z}} q_\phi(\vec{z}|\vec{x},\vec{u}) log \frac{p_\theta(\vec{x}|\vec{z},\vec{u})p_0(\vec{z}|\vec{u})}{q_\phi(\vec{z}|\vec{x},\vec{u})}) d\vec{z} \\ &= \mathbb{E}_{q_\phi(\vec{z}|\vec{x},\vec{u})}[log\ p_\theta(\vec{x}|\vec{z},\vec{u})] - KL(q_\phi(\vec{z}|\vec{x},\vec{u})\|p_0(\vec{z}|\vec{u})) \\ &= \sum_{t=1}^{T} \mathbb{E}_{z_t \sim q_\phi(z_t|\vec{x},\vec{u})}[log\ p_\theta(x_t|z_t,u_{t-1})] - KL(q_\phi(\vec{z}|\vec{x},\vec{u})\|p_0(\vec{z}|\vec{u})) \\ &= \mathcal{L}(x;(\theta,\phi)) \end{aligned} \tag{2.3}$$

The new factorization of the KL term is:

$$\begin{aligned} KL(q_\phi(\vec{z}|\vec{x},\vec{u})\|p_0(\vec{z})) &= \int_{z_1} \cdots \int_{z_T} q_\phi(z_1|\vec{x},\vec{u}) .... q_\phi(z_T|z_{T-1}\vec{x},\vec{u}) \\ &* \ log \frac{p_0(z_1,...,z_T)}{q_\phi(z_1|\vec{x},\vec{u}) .... q_\phi(z_T|z_{T-1}\vec{x},\vec{u})} d\vec{z} \\ &= KL(q_\phi(z_1|\vec{x},\vec{u})\|p_0(z_1)) \\ &+ \sum_{t=2}^{T} \mathbb{E}_{z_t \sim q_\phi(z_t|\vec{x},\vec{u})}[KL(q_\phi(z_t|z_{t-1},\vec{x},\vec{u})\|p_0(z_t|z_{t-1},u_{t-1}))] \end{aligned}$$

Substitute this new factorization in the Equation 2.3, we can get the lower bound as following:

$$\begin{aligned} log\ p_\theta(\vec{x}|\vec{u}) &\geq \mathcal{L}(x;(\theta,\phi)) \\ &= \sum_{t=1}^{T} E_{q_\phi(z_t|\vec{x},\vec{u})}[log\ p_\theta(x_t|z_t)] - KL(q_\phi(z_1|\vec{x},\vec{u})\|p_0(z_1)) \\ &- \sum_{t=2}^{T} E_{q_\phi(z_{t-1}|\vec{x},\vec{u})}[KL(q_\phi(z_t|z_{t-1},\vec{x},\vec{u})\|p_0(z_t|z_{t-1},u_{t-1}))] \end{aligned} \tag{2.4}$$

Then we can still use the Monte Carlo estimation to evaluate the marginal liklihood:

$$
\begin{aligned}
p(\vec{x}) &\approx \frac{1}{S}\sum_{s=1}^{S}[\frac{p(\vec{x}|\vec{z}^{(s)})p(\vec{z}^{(s)})}{q(\vec{z}^{(s)}|\vec{x})}],\ \ \vec{z}^{(s)} \sim q(\vec{z}|\vec{x}) \\
log\ p(\vec{x}) &\approx log\ \frac{1}{S}\sum_{s=1}^{S} exp(log[\frac{p(\vec{x}|\vec{z}^{(s)})p(\vec{z}^{(s)})}{q(\vec{z}^{(s)}|\vec{x})}])
\end{aligned} \tag{2.5}
$$

2.3.1 KL divergence computation

Suppose we have two multivariate Gaussians $q \sim \mathcal{N}(\mu_q, \Sigma_q), p \sim \mathcal{N}(\mu_p, \Sigma_p)$. The KL divergence between them can be written as:

$$KL(q\|p) = \frac{1}{2}(log\frac{|\Sigma_p|}{|\Sigma_q|} - D + Tr(\Sigma_p^{-1}\Sigma_q) + (\mu_p - \mu_q)^T\Sigma_p^{-1}(\mu_p - \mu_q))$$

The output of variational model provides us μ_q, Σ_q, where μ_p, Σ_p depends on the generative model with $\mu_{p1} = 0, \Sigma_{p1} = \mathbf{1}, \mu_{pt} = G_{t-1}, \Sigma_{p1} = \Delta\vec{\sigma}$.

If $t = 1$, then

$$
\begin{aligned}
log\frac{|\Sigma_{p1}|}{|\Sigma_{q1}|} &= -log|\Sigma_{q1}| \\
Tr(\Sigma_{p1}^{-1}\Sigma_{q1}) &= Tr(\Sigma_{q1}) \\
(\mu_{p1} - \mu_{q1})^T\Sigma_{p1}^{-1}(\mu_{p1} - \mu_{q1}) &= \|\mu_{q1}\|^2
\end{aligned}
$$

If $t > 1$, then

$$\begin{aligned}
log\frac{|\Sigma_{pt}|}{|\Sigma_{qt}|} &= log|\Sigma_{pt}| - log|\Sigma_{qt}| \\
&= Dlog(\Delta) + log|\vec{\sigma}| - log|\Sigma_{qt}| \\
Tr(\Sigma_{pt}^{-1}\Sigma_{qt}) &= \frac{1}{\Delta}Tr(diag(\vec{\sigma})^{-1}\Sigma_{qt}) \\
(\mu_{pt} - \mu_{qt})^T\Sigma_{pt}^{-1}(\mu_{pt} - \mu_{qt}) &= \Delta(G_{t-1} - \mu_{qt})^T diag(\vec{\sigma})^{-1}(G_{t-1} - \mu_{qt})
\end{aligned}$$

Therefore, we can rewrite the KL divergence as:

$$\begin{aligned}
KL(q(z_1, ..., z_T)\|p(z_1, ..., z_T)) &= \frac{1}{2}((T-1)Dlog(\Delta)log|\vec{\sigma}| - \sum_{t=1}^{T} log|\Sigma_{qt}| + Tr(\Sigma_{q1}) \\
&+ \Delta\sum_{t=2}^{T} E_{z_{t-1}}[(G_{t-1} - \mu_{qt})^T diag(\vec{\sigma})^{-1}(G_{t-1} - \mu_{qt})] \\
&+ \frac{1}{\Delta}\sum_{t=2}^{T} Tr(diag(\vec{\sigma})^{-1}\Sigma_{qt}) + \|\mu_{q1}\|^2) \qquad (2.6)
\end{aligned}$$

With Equation 2.5, we can take gradients with respect to μ_{qt}, Σ_{qt}, and $G(z_{t-1}, u_{t-1})$. Since the KL divergence could be evaluated analytically, the resulting objective function has stable analytic gradients [11].

2.3.2 Learning with Gradient Descent

The Equation 2.4 is differentiable with respect to (θ, ϕ). If we fixed the generative model parameter θ, we can perform stochastic gradient ascent of the objective function in ϕ. We just perform the stochastic gradient ascent in both θ and ϕ. To update parameter θ, we can use backpropagation, while we can use stochastic backpropagation to estimate the gradient $\nabla_\phi q_\phi(z_t)$. Thus, the overall learning algorithm will be [11]:

Algorithm 1 Learning a Deep Kalman Filter model with Stochastic Gradient Descent. Monte-Carlo estimates are used over K samples from the recognition network during learning to evaluate expectations in the bound and gradients.

```
1: Input:  Observations sequence: $\vec{\boldsymbol{x}}$
2: Inference Model: $q_{\boldsymbol{\phi}}(\vec{\boldsymbol{z}}|\vec{\boldsymbol{x}})$
3: Generative Model: $p_{\boldsymbol{\theta}}(\vec{\boldsymbol{x}}|\vec{\boldsymbol{z}}), p_{\boldsymbol{\theta}}(\vec{\boldsymbol{z}})$
4: while  not converged do
5:     $\vec{x} \leftarrow$ sample MiniBatch
6:     Sample $\hat{\vec{\boldsymbol{z}}} \sim q_{\boldsymbol{\phi}}(\vec{\boldsymbol{z}}|\vec{\boldsymbol{x}})$
7:     Estimate $p_{\boldsymbol{\theta}}(\vec{\boldsymbol{x}}|\hat{\vec{\boldsymbol{z}}})$
8:     Compute KL divergence between posterior and prior
9:     Evaluate $\mathcal{L}(\boldsymbol{x}^{(i)}; (\boldsymbol{\theta}, \boldsymbol{\phi}))$
10:    Estimate $\nabla_{\boldsymbol{\theta}}\mathcal{L}, \nabla_{\boldsymbol{\phi}}\mathcal{L}$
11:    Update $\boldsymbol{\theta}, \boldsymbol{\phi}$ using ADAM
12: end while
```

2.3.3 ADAM

ADAM is an algorithm for first-order gradient-based optimization of stochastic objective functions, based on adaptive estimates of lower-order moments [12]. Designed specifically for training deep neural networks, this method computes individual adaptive learning rates for different parameters from estimates of first and second moments of the gradients.

Suppose we want to optimize the expectation of the objective function $f(\theta)$ that is differentiable with respect to the parameters θ. The algorithm updates exponential moving averages of the gradient (m_t) and the squared gradient (v_t) where the hyper-parameters $\beta_1, \beta_2 \in [0, 1)$ control the exponential decay rates of these movingv averages. The moving averages themselves are estimates of the first moment (the mean) and the second raw moment (the uncentered variance) of the gradient. The general process of ADAM optimization is shown below: [12]

Algorithm 2 ADAM Optimization: α is the step size, $\beta_1, \beta_2 \in (0, 1]$ are exponential decay rates for the moment estimates, and $f(\theta)$ is the stochastic objective function with parameters θ

```
 1: Initialization $t \leftarrow 0$
 2: $m_0 \leftarrow 0$
 3: $v_0 \leftarrow 0$
 4: while $\theta_t$ not converged do
 5:     $t \leftarrow t + 1$
 6:     $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$.
 7:     $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$
 8:     $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$
 9:     $\hat{m}_t \leftarrow m_t / (1 - \beta_1^t)$
10:     $\hat{v}_t \leftarrow v_t / (1 - \beta_2^t)$
11:     $\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon)$
12: end while
13: return $\theta_t$
```

2.4 Variational Autoencoder

The deep Kalman filter model is a general model that can be fitted into various scenarios with different functional forms of $G_\alpha, S_\beta, F_\kappa$. In this section, we will illustrate how to use a neural network to approximate the probabilistic encoder $q_\phi(z|x)$, so as to estimate the posterior of the generative model $p_\theta(x|z)$, and optimize the parameters ϕ and θ jointly. Figure 2.2 below gives a general idea of the learning process for the variational autoencoder.

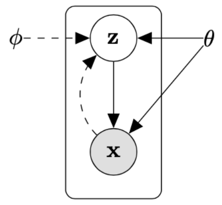


Figure 2.2: Variational Autoencoder (figure from [7])

2.4.1 Reparameterization trick

Before we demonstrate the example of variational autoencoder, we have to first introduce an essential trick for parametrization, i.e., the reparameterization trick, which makes the non-differentiable network trainable by moving the non-differentiable operations out of the network.

Suppose we want to estimate some conditional distribution $\boldsymbol{z} \sim q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$. We can express the random variable $\boldsymbol{z}$ as a deterministic variable $\boldsymbol{z} = g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}, \boldsymbol{x})$, where $\boldsymbol{\epsilon}$ is an auxiliary variable with independent marginal $p(\boldsymbol{\epsilon})$, and $g_{\boldsymbol{\phi}}()$ is some vector-valued function parameterized by $\boldsymbol{\phi}$. Depending on the types of distribution of $q_{\phi}(\boldsymbol{z}|\boldsymbol{x})$, we can choose different transformation g_{ϕ} and auxiliary variable $\boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon})$. For example, if the $q_{\phi}(\boldsymbol{z}|\boldsymbol{x})$ has tractable inverse CDF, such as exponential Cauchy, Logistic, and Reciprocal distribution, we can choose $g_{\phi}(\boldsymbol{\epsilon,x})$ to be the inverse CDF of $q_{\phi}(\boldsymbol{z}|\boldsymbol{x})$, and let $\boldsymbol{\epsilon} \sim \mathcal{U}(\boldsymbol{0}, \boldsymbol{I})$. If the distribution of $q_{\phi}(\boldsymbol{z}|\boldsymbol{x})$ belongs to 'location-scale' family, such as Laplace, Elliptical, Uniform, Triangular and Gaussian distributions, we can choose the standard distribution (with location = 0, scale = 1) as the auxiliary variable $\boldsymbol{\epsilon}$, and g_{ϕ} =location + scale$*\boldsymbol{\epsilon}$. To be more specific, suppose $z \sim p(z|x) = \mathcal{N}(\mu, \sigma^2)$. The reparameterization for this case is $z = \mu + \sigma\epsilon$, where the auxiliary variable $\epsilon \sim \mathcal{N}(0,1)$. Then we can write the expectation of the objective function as:

$$\begin{aligned}\mathbb{E}_{\mathcal{N}(z;\mu,\sigma^2)}[f(z)] &= \mathbb{E}_{\mathcal{N}(\epsilon;0,1)}[f(\mu+\sigma\epsilon)] \\ &\approx \frac{1}{L}\sum_{l=1}^{L} f(\mu+\sigma\epsilon^{(l)}),\ \epsilon^{(l)} \sim \mathcal{N}(0,1)\end{aligned}$$

In addition, if the distribution of $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ is the composition of multiple distributions, such as Log-Normal, Gamma and Chi-Squared distributions, we can always express random variables as different transformations of auxiliary variables [12].

2.4.2 Variational Autoencoder Derivation

For variational autoencoder, we assume that the prior distribution latent variables is a multivariate Gaussian, i.e., $p_{\boldsymbol{\theta}}(\boldsymbol{z}) \sim \mathcal{N}(\boldsymbol{z}, \mathbf{0}, \boldsymbol{I})$. Another essential assumption here is that the posterior distribution $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$ is also a multivariate Gaussian. To overcome the intractability of the posterior, we will use a MLP (detailed description later in this section) to computer the distribution parameters from $\boldsymbol{z}$. As we assume the $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$ to be a multivariate Gaussian with diagonal convariance, we can write the variational model as following:

$$log\ q_\phi(\boldsymbol{z}|\boldsymbol{x}^{(i)}) = log\ \mathcal{N}(\boldsymbol{z}; \boldsymbol{\mu}^{(i)}, \boldsymbol{\sigma}^{2(i)}\boldsymbol{I})$$

The mean $\boldsymbol{\mu}^{(i)}$ and standard deviation $\boldsymbol{\sigma}^{2(i)}\boldsymbol{I}$ here are the outputs of the encoding MLP.

At data point $\boldsymbol{x}^{(i)}$, we have to estimate the posterior by the recognition network, i.e., $\boldsymbol{z}^{(i,l)} \sim q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$. Applying the reparameterization trick described above, we can write the following

$$\boldsymbol{z}^{(i,l)} = g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}^{(l)}, \boldsymbol{x}^{(i)}) = \boldsymbol{\mu}^{(i)} + \boldsymbol{\Sigma}^{(i)}\boldsymbol{\epsilon}^{(l)}$$

where $\boldsymbol{\epsilon}^{(l)} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ is also normally distributed. The differentiable transformation $g_{\boldsymbol{\phi}}()$ here is $\boldsymbol{z} = \boldsymbol{\mu} + \boldsymbol{\Sigma}\boldsymbol{\epsilon}$, while the auxiliary variable $\boldsymbol{\epsilon}$ has independent marginal $p(\boldsymbol{\epsilon}) \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$. Then we don't care anymore with the sampling process during back-propagation, as it is now outside of the network, i.e. doesn't depend on anything in the net, hence the gradient will not flow through it. Then we can further derive:

$$\begin{aligned}
-KL(q_{\phi}(\boldsymbol{z}|\boldsymbol{x})|p_{\theta}(\boldsymbol{z})) &= \int q_{\phi}(\boldsymbol{z}|\boldsymbol{x})log(\frac{p_{\theta}(\boldsymbol{z})}{q_{\phi}(\boldsymbol{z}|\boldsymbol{x})})d\boldsymbol{z} \\
&= \int q_{\phi}(\boldsymbol{z}|\boldsymbol{x})[log(p_{\theta}(\boldsymbol{z})) - log(q_{\phi}(\boldsymbol{z}|\boldsymbol{x}))]d\boldsymbol{z} \\
\int q_{\phi}(\boldsymbol{z}|\boldsymbol{x})log(p_{\theta}(\boldsymbol{z}))d\boldsymbol{z} &= \int \mathcal{N}(\boldsymbol{z};\boldsymbol{\mu},\boldsymbol{\Sigma})log(\mathcal{N}(\boldsymbol{z};\boldsymbol{0},\boldsymbol{I}))d\boldsymbol{z} \\
&= -\frac{J}{2}log(2\pi) - \frac{1}{2}\sum_{j=1}^{J}(\mu_j^2 + \sigma_j^2) \\
\int q_{\phi}(\boldsymbol{z}|\boldsymbol{x})log(q_{\phi}(\boldsymbol{z}|\boldsymbol{x}))d\boldsymbol{z} &= \int \mathcal{N}(\boldsymbol{z};\boldsymbol{\mu},\boldsymbol{\Sigma})log(\mathcal{N}(\boldsymbol{z};\boldsymbol{\mu},\boldsymbol{\Sigma}))d\boldsymbol{z} \\
&= -\frac{J}{2}log(2\pi) - \frac{1}{2}\sum_{j=1}^{J}(1 + log(\sigma_j^2)) \\
-KL(q_{\phi}(\boldsymbol{z}|\boldsymbol{x})|p_{\theta}(\boldsymbol{z})) &= -\frac{J}{2}log(2\pi) - \frac{1}{2}\sum_{j=1}^{J}(\mu_j^2 + \sigma_j^2) \\
&- [-\frac{J}{2}log(2\pi) - \frac{1}{2}\sum_{j=1}^{J}(1 + log(\sigma_j^2))]
\end{aligned}$$

$$
\begin{aligned}
-KL(q_\phi(\boldsymbol{z}|\boldsymbol{x})|p_\theta(\boldsymbol{z})) &= \frac{1}{2}\sum_{j=1}^{J}(1+log(\sigma_j^2)-\mu_j^2-\sigma_j^2) \\
L(\boldsymbol{\theta},\boldsymbol{\phi};\boldsymbol{x}^{(i)}) &\approx \frac{1}{2}\sum_{j=1}^{J}(1+log((\sigma_j^{(i)})^2)-(\mu_j^{(i)})^2)-(\sigma_j^{(i)})^2))+\frac{1}{L}\sum_{l=1}^{L} log\ p_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}|\boldsymbol{z}^{(i,l)})
\end{aligned}
$$

Then we will use a MLP as the decoder to estimate the term $log\ p_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}|\boldsymbol{z}^{(i,l)})$.

2.4.3 Multi-layer Perceptron

Two MLPs are used in variational autoencoder, one for the encoding and one for decoding. The left part in Figure 2.3 is the encoder MLP, which maps the input

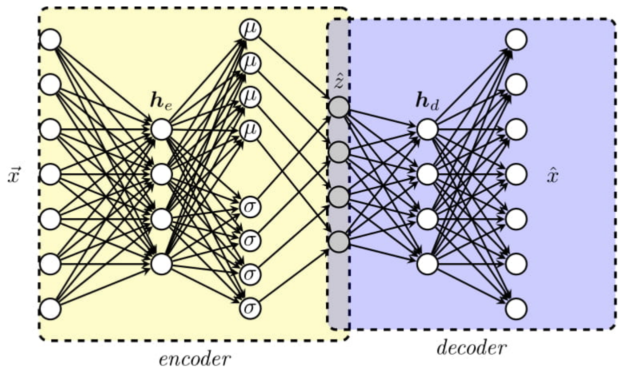


Figure 2.3: VAE structure from network perspective

$\vec{x}$ to two parameters $\boldsymbol{\mu}, \boldsymbol{\sigma}$. Then the middle part is where we sample $\hat{z} \sim q_\phi(\vec{z}|\vec{x})$ using reparameterization trick. The right part is the decoder MLP, which maps $\hat{z}$ to output $\hat{x}$, denoted as $p_\theta(\vec{x}|\hat{z})$.

Since we choose the encoder and the decoder to be a multivariate Gaussian with diagonal covariance, the MLP will have following structure:

$$\begin{aligned}
log\ p(\boldsymbol{x}|\boldsymbol{z}) &= log\ \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma I}) \\
\boldsymbol{\mu} &= \boldsymbol{W_1 h} + \boldsymbol{b_1} \\
log\ \boldsymbol{\Sigma} &= \boldsymbol{W_2 h} + \boldsymbol{b_2} \\
\boldsymbol{h} &= tanh(\boldsymbol{W_3 z} + \boldsymbol{b_3})
\end{aligned}$$

When this MLP structure is used as decoder, $\{\boldsymbol{W}_1, \boldsymbol{W}_2, \boldsymbol{W}_3\}$ are the weights and $\{\boldsymbol{b}_1, \boldsymbol{b}_2, \boldsymbol{b}_3\}$ are the biases of the MLP, while all these parameters are part of parameters $\boldsymbol{\theta}$. When used as encoder, we have to swap $\boldsymbol{z}, \boldsymbol{x}$, and $\{\boldsymbol{W}_1, \boldsymbol{W}_2, \boldsymbol{W}_3, \boldsymbol{b}_1, \boldsymbol{b}_2, \boldsymbol{b}_3\}$ are part of parameters $\boldsymbol{\phi}$.

2.5 Long-Short Term Memory Recurrent Neural Network

Since the distribution of $q_\phi(z|x)$ is unknown and we cannot assume it to be Gaussian, we only used the MLP as decoder in our real implementation. Instead of the encoder MLP, we employ a two-layer bi-directional Long-Short Term Memory Recurrent Neural Network (LSTM) as the sequential variational model.

Recurrent neural network (RNN), the network architecture that use its internal state (memory) to process variable length sequences of inputs, is widely used to process sequences related tasks, such as speech recognition [13], machine translation [14], image captioning [15], and video classification [16]. Theoretically, vanilla RNNs are capable of learning arbitrary long-term dependencies in the input sequences. We can process a sequence of vectors x by applying a recurrence formula at every time step: $h_t = f_W(h_{t-1}, x_t)$. All recurrent neural networks have the form of a chain of repeating modules of neural network. The repeating modules in classic RNNs could be very simple, such as a single hyperbolic tangent layer. Then we can write:

$$\begin{aligned} h_t &= tanh(W_{hh}h_{t-1} + W_{xh}x_t) = tanh(W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}) \\ y_t &= W_{hy}h_t \end{aligned}$$

However, in practice, the classic RNN suffers from computational challenges. When trying to train a vanilla RNN with backpropagation, the gradients may either vanish, i.e, $\nabla \to 0$, or explode, i.e., $\nabla \to \infty$. In order to solve the vanishing gradient problem, a modified RNN architecture has been proposed, i.e., Long-Short Term Memory (LSTM), although it still suffers from the exploding gradient problem.

Instead of the simple structure of the repeating module in vanilla RNNs, the repeating module of LSTM contains four interacting layers. A common LSTM unit contains a cell and three regulators, i.e., an input gate, an output gate and a forget gate. The LSTM can add or remove information to the cell state by three gates, which are composed of a pointwise multiplication operation and an activation layer (usually sigmoid layer). The cell is used to keep track of the long-term dependencies between the elements in the input sequence. The input gate determines what new information is going to flow into the cell, the forget gate determines what information will remain in the cell, and the output gate determines what information in the cell will be used to compute the output. To decide what information is going to be stored in the cell, we actually have two gates, the first one decides whether to write to the cell using a sigmoid layer, while the second one controls the extent to which a value remains in the cell using a hyperbolic tangent layer. We can denote the first gate with sigmoid layer as the input gate i, and the second tanh layer as gate g. We can also denote o to be the output gate, f to be the forget gate, and c_t to be the cell state at time step t. Then the LSTM can be written as:

$$\begin{aligned}
\begin{bmatrix} i \\ f \\ o \\ g \end{bmatrix} &= \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ tanh \end{bmatrix} W \begin{bmatrix} h_{t-1} \\ x_t \end{bmatrix} \\
c_t &= f \odot c_{t-1} + i \odot g \\
h_t &= o \odot tanh(c_t)
\end{aligned}$$

It is not surprising that the regular RNN has many limitations. Thus, a modified version of RNN, bidirectional recurrent neural network (BRNN), is proposed to improve the performance. Both past and future input of a specific time frame can be used to train the BRNN. The general idea is to split the split the state neurons of a regular RNN into two parts, one for the positive time direction (forward) and one for the negative time direction (backward) [17]. Similarly, bi-directional long-short term memory (BiLSTM) RNN is an extension of regular LSTM RNN. Taking in all available input information from both the past and the future, the BiLSTM network outperforms the unidirectional neural networks on sequence classification problem [18]. Therefore, the bi-directional LSTM RNN is used as the inference model of the deep Kalman filter in this thesis.

Chapter 3: Real-time Tracking of Selective Auditory Attention

As mentioned in Chapter 1, the general framework of the real-time tracking of selective auditory attention is composed of three modules, i.e., the Dynamic Encoder/Decoder Estimation module, the Attention Marker module, and the Dynamic State-Space Model module [4]. The first module estimates the model coefficients that fit to the neural data in real time. The output of the encoding/decoding models is passed into the second module, where the features are modulated by the instantaneous attentional state. Then the features keep going to the last module to achieve a dynamic estimation of the attentional state.

In this thesis, we mainly focus on using the deep Kalman filter to conduct the TRF estimates. Thus, we will explain the dynamic encoding/decoding module in detail as our goal is to implement this module with deep Kalman filter. This section is organized as following: In section 3.1, we will employ the deep Kalman filter algorithm to get the real time estimates of TRF. Then in section 3.2, we will define two types of attention markers, and elaborate the dynamic state-space model that outputs the estimates of the attenional states.

3.1 Dynamic Encoding & Decoding Estimation

One of the ultimate goals of auditory neuroscience is to understand the mapping between the auditory stimuli and the corresponding neural response. The neural encoding model can predict the neural response from the stimulus, while a neural decoding model is used to express the stimulus as a function of the neural response. This mapping can be measured by temporal response functions (TRFs). Thus, the purpose of this module is to estimate TRFs given the auditory neural response and the speech envelope of speakers. Previous research has shown that the TRF is a sparse kernel, which regresses auditory MEG data with respect to the envelopes of the speech streams. Therefore, the temporal response function can be modeled over a shifted Gaussian kernel with time-varying coefficients, where the coefficients are assumed to be sparse [19]. If the coefficients are expected to be sparse in a basis represented by the columns of a matrix G, such as the Haar or Gabor bases, we can multiply the stimuli by the base G, compute the TRF estimates as usual, then multiply estimates by the matrix G to get the final estimates.

In this case, the stimuli are represented by the speaker's covariate matrix composed of the speech envelopes, and the neural responses are recorded with MEG/EEG channels. Let $s_t^{(1)}, s_t^{(2)}$ to be the speech envelopes of speakers 1 and 2 at time t respectively, and e_t^c denotes the neural response recorded at time t and channel c. In addition, we divide the inputs into consecutive and non-overlapping windows with same length W. The encoding and decoding coefficients are assumed to be constant over each window due to the piece-wise constant dynamics.

For the encoding model, the stimuli, modulated by the covariate matrix, at k^{th} window is $\boldsymbol{X}_k = [\mathbf{1}_{W\times 1}, \boldsymbol{X}_k^{(1)}, \boldsymbol{X}_k^{(2)}]$, where $\boldsymbol{X}_k^{(i)} = [\boldsymbol{s}_{(k-1)W+1}^{(i)}, \boldsymbol{s}_{(k-1)W+2}^{(i)}, ..., \boldsymbol{s}_{kW}^{(i)}]^T$, and $\boldsymbol{s}_t^{(i)} = [s_t^{(i)}, s_{t-1}^{(i)}, ..., s_{t-L_e}^{(i)}]^T$, L_e here is the total lag in the encoding model. The vector $\boldsymbol{y}_k = [E_{(k-1)W+1}, E_{(k-1)W+2}, ..., E_{kW}]^T$ is defined as the neural response, where E_t is the linear combination of $e_t^1, e_t^2, ..., e_t^c$ with weights. The weights are used to select a single channel, which makes the E_t represent the dominant component of the neural response. For the decoding model, we set $\boldsymbol{y}_k = [\boldsymbol{s}_{(k-1)W+1}^{(i)}, \boldsymbol{s}_{(k-1)W+2}^{(i)}, ..., \boldsymbol{s}_{kW}^{(i)}]^T$. The decoding covariate matricx at k^{th} window is $\boldsymbol{X}_k = [\boldsymbol{\varepsilon}_{(k-1)W+1}, \boldsymbol{\varepsilon}_{(k-1)W+2}, ..., \boldsymbol{\varepsilon}_W]^T$, where $\boldsymbol{\varepsilon}_t = [1, \boldsymbol{e}_t^T, \boldsymbol{e}_{t+1}^T, ..., \boldsymbol{e}_{t+L_d}^T]^T$, and $\boldsymbol{e}_t = [e_t^{(1)}, e_t^2, ..., e_t^C]^T$, L_d here is the total lag in the decoding model, which affects the dependency between the future neural responses and the current stimuli.

With all these notations, if we use a linear encoding/decoding model, we can write the estimation problem as the following optimization problem:

$$\hat{\boldsymbol{\theta}}_k = arg\ min_{\boldsymbol{\theta}} \sum_{j=1}^{k} \lambda^{k-j} \|\boldsymbol{y}_j - \boldsymbol{X}_j \boldsymbol{\theta}\|_2^2 + \gamma \|\boldsymbol{\theta}\|_1, \quad k = 1, ..., K \tag{3.1}$$

At each window k, for $k = 1, ..., K, \hat{\boldsymbol{\theta}}_k$ are updated based on the new measurements, $\boldsymbol{y}_k, \boldsymbol{X}_k$, and previous measurements through the forgetting factor $\lambda \in (0, 1]$, while γ is a regularization parameter and $\boldsymbol{\theta}$ is the parameter vector. As we stated before, since the TRF is sparse, we have to replace $\boldsymbol{X}_j$ in Equation 3.1 by $\boldsymbol{X}_j G$, and solve for $\hat{\boldsymbol{\theta}}_k$, where the final estimates should also multiply the matrix G. In other words, the final model coefficients should be $G\hat{\boldsymbol{\theta}}_k$.

In previous research, the optimization problem in Equation 3.1 is solved using

Forward-Backward Splitting (FBS) method, which computes the gradient descent of the log-likelihood term, and then applies a soft-thresholding shrinkage operator. However, the linear model is limited in real-world application. Thus, in this thesis, we replace the regularized least squares (RLS) algorithm with the deep Kalman filter model as explained in previous section.

3.2 Attention Markers & Dynamic State Space Model

The attention marker is essentially a mapping function, which associates the encoding/decoding model coefficients and the covariate matrix to features. It is a measure of how well a decoder can reconstruct its envelope. Since the temporal response function is the encoding coefficients in the context of encoding model for the first module, in this case, the inputs of the attention marker extraction module would be the covariate matrix $\boldsymbol{X}_k^{(i)}$ for each sound source i, the estimated encoding model coefficients, i.e., TRF estimates, $\hat{\boldsymbol{\theta}}_k^{(i)}$, and the neural response $\boldsymbol{y}_k$ recorded from M/EEG channel at time window k, and the output of the second module would be a positive real number denoted as $m_k^{(i)}$. In the specific case of this thesis, there will be two outputs $m_k^{(1)}, m_k^{(2)}$, for speaker 1 and 2 respectively, from the attention markers. They will be further used in the dynamic state space model as measures of the attentional state.

From previous research findings, which states that a trained attended decoder results in 10% more attention decoding accuracy than a trained unattended decoder [20], we can assume that the attended speaker has more influence on the real-time auditory M/EEG response than the unattended speaker. The corresponding decoder of the attended speaker will perform better than that of the unattended speaker in the reconstruction of the speech envelope. In other words, the significant components in the auditory neural response are provided by the attended speaker, while the unattended speaker will result in small and random components. Utilizing this property, we can extract features in two different ways, i.e., correlation-based attention marker, and l_1 norm based attention marker. The correlation-based attention marker can be derived as:

$$m_k^{(i)} = f(\hat{\boldsymbol{\theta}}_k^{(i)}, \boldsymbol{X}_k, \boldsymbol{y}_k^{(i)}) := |\text{corr}(\boldsymbol{y}_k^{(i)}, \boldsymbol{X}_k \hat{\boldsymbol{\theta}}_k^{(i)})|, \text{ for } i = 1, 2, \; k = 1, ..., K$$

Therefore, the attention marker in this scenario is the correlation magnitude between the speech envelope and its reconstruction by the corresponding decoder. Since l_1 norm of the decoder is able to capture the significant components, it can also be used for feature extraction. The l_1 norm based attention marker can be calculated as following:

$$m_k^{(i)} = \|\hat{\boldsymbol{\theta}}_k^{(i)}\|_1, \text{ for } i = 1, 2, \; k = 1, ..., K$$

l_1 norm based attention marker provides smoother results than correlation based attention marker does, but the correlation based attention marker is more reliable, thus we will mainly focus on the results of the correlation based attention marker in this thesis.

However, the real-time situation is much more comnplex with lots of uncertainty and stochastic fluctuations due to limited data and integration time. Even though the attention markers are validated in batch mode analysis, we still have to introduce a state-space model to correct all the uncertainty and fluctuations. To be more specific, we assume a linear state-space model based on the attention. At instance k, we define the binary random variable so that $n_k = 1$ when speaker 1 is attended and $n_k = 2$ when speaker 2 is attended. Then we want to estimate the probability of attention on speaker 1 $p_k := P(n_k = 1), \forall\ 1 \leq k \leq K_A$, where K_A is window length [4].

$$\begin{cases} p_k = P(n_k = 1) = 1 - P(n_k = 1) = \frac{1}{1+exp(-z_k)} \\ z_k = c_0 z_{k-1} + w_k \\ w_k \sim \mathcal{N}(0, \eta_k) \\ \eta_k \sim \Gamma^{-1}(a_0, b_0) \end{cases} \tag{3.2}$$

Equation (3.2) describes the dynamic of the latent variable z_k. We also need an observation model to relate the observations $m_k^{(1)}, m_k^{(2)}$ with the state dynamics in above equation.

$$\begin{cases} \begin{cases} m_k^{(i)} | n_k = i \sim \text{Log Normal}(\rho^{(a)}, \mu^{(a)}) \\ m_k^{(i)} | n_k \neq i \sim \text{Log Normal}(\rho^{(u)}, \mu^{(u)}) \end{cases} , \ i = 1, 2 \\ \rho^{(a)} \sim \Gamma(\alpha_0^{(a)}, \beta_0^{(a)}),\ \mu^{(a)} | \rho^{(a)} \sim \mathcal{N}(\mu_0^{(a)}, \rho^{(a)}) \\ \rho^{(u)} \sim \Gamma(\alpha_0^{(u)}, \beta_0^{(u)}),\ \mu^{(u)} | \rho^{(u)} \sim \mathcal{N}(\mu_0^{(u)}, \rho^{(u)}) \end{cases} \tag{3.3}$$

Equation (3.4) states that we use two log-normal distribution with different parameters on $m_k^{(i)}$, depending on whether the corresponding speaker is attended. These log-normal distribution are approximated by Gaussian density in the implementation.

Chapter 4: Results & Discussion

We implemented the deep Kalman filter model in MATLAB with the neural network structure described in section 4.1. This neural network is first applied to an one dimensional random walk for validation. Then we apply it to a simulated MEG dataset to derive TRF estimates. The TRF estimates are discussed in section 4.2. Finally, the deep Kalman filter is used in the general framework as a dynamic encoding module on a simulated EEG dataset. The results are shown in section 4.3.

4.1 Neural Network Structure

The paper *Deep Kalman Filters* [7] compared four different choices of variational models with increasing complexity, i.e., parameterizing $q(z_t|x_t)$ by an MLP (denoted as q**INDEP** in Figure 4.1), parameterizing $q(z_t|x_{t-1}, x_t, x_{t+1})$ by an MLP (denoted as q**LR** in Figure 4.1), parameterizing $q(z_t|x_1, ..., x_t)$ by an RNN (denoted as q**RNN** in Figure 4.1), and parameterizing $q(z_t|x_1, ..., x_t)$ by a bi-directional RNN (denoted as q**BRNN** in Figure 4.1). Based on their experimental results on the healing MNIST dataset constructed by applying rotations to the hand-written digits, the bi-directional RNN outperforms the other recognition models as we can see from Figure 4.1. It is not surprising since the Bi-Directional RNN, similar to the

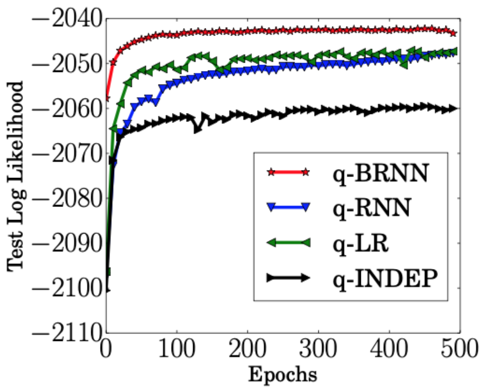


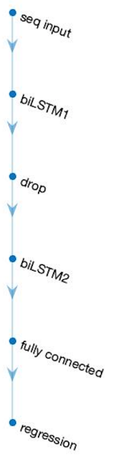
Figure 4.1: Test log likelihood on four recognition models (figure from [7])

Forward-Backward algorithm, takes in the information from both the past and the future at every time step to form the most effective approximation to the posterior distribution of z_t. Therefore, we chose the bi-directional RNN as the sequential variational model for the deep Kalman filter. To be more specific, the inference model is a two-layer Bi-directional Long-Short Term Memory (Bi-LSTM) Recurrent Neural Network to look at the input sequence in both forward and backward directions, while Multi-layer Perceptrons (MLP) is chosen to implement the generative model. The overall structure is shown as in Figure 2.1 in previous chapter. The observations $\boldsymbol{x}_t$ will first pass through the two-layer BiLSTM RNN in order to sample $\hat{z}_t \sim q_\phi(\vec{z}|\vec{x}, \vec{u})$, and the estimates of $\hat{z}_t$ will get into the decoder MLP to approximate $\hat{x} \sim p_\theta(\vec{x}|\hat{z})$. In this case, $\hat{z}_t$ is the TRF estimates we calculated in previous section.

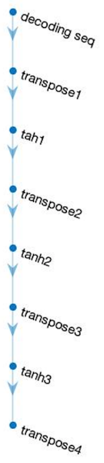
Figure 4.2 shows the detailed composition of layers for each network. For the two-layer bi-directional LSTM RNN, the first BiLSTM layer with 200 hidden units will process the sequence input by mapping it to 200 features and generating an output sequence. Then a dropout layer with 0.2 probability is used to regularize the model and prevent over-fitting. Another BiLSTM layer with 200 hidden units first maps its input into 200 features and then prepares the output for the fully connected layer afterwards. The final output of the recognition model is achieved by passing the previous output through a regression layer. The input of the decoder MLP will go through 2-D transposed convolution layers that up-sample feature maps. There are four 2-D transposed convolution layers in total, where each of them are the transpose of convolution. There is also a hyperbolic tangent (tanh) activation layer between each two transposed convolution layers. The first two 2-D transposed convolution layers have 64 filters with size [10,10] and [3,3] respectively, the third one has 32 filters with size [3,3], while the last one only has 1 filter with size [3,3] to generate the prediction of the observation sequence.

The input to the BiLSTM RNN is the sequence of observations, with size [1 $\times$ number of samples], while the output of RNN is the estimates of TRF, with size [number of estimates $\times$ number of samples]. Then this output goes into the MLP to generate the predictions of the observations. Since the TRF is represented in a Garbor basis due to its sparsity, we have to choose a proper distance between the adjacent Gabor atoms in the lag domain. For example, if the window length of TRF estimates is 250ms, and we want to make each atom shifted by 50ms. Then we will have 5 atoms to cover this 250 ms window. Thus, the number of estimates for each speaker will be 5 in this case.

We first apply the above neural network architecture with the one-dimensional random walk with different number of steps to examine its functionality. We con-



(a) BiLSTM RNN



(b) decoder MLP

Figure 4.2: Network Architecture

struct the system as following:

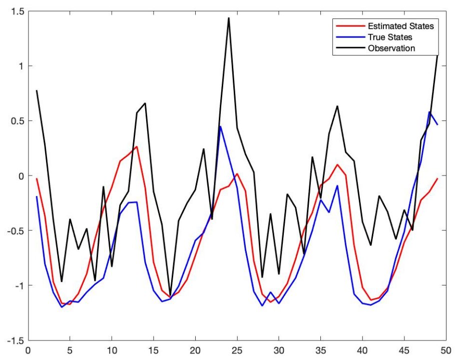
$$x_k = x_{k-1} + w_k$$

$$y_k = x_k + v_k$$

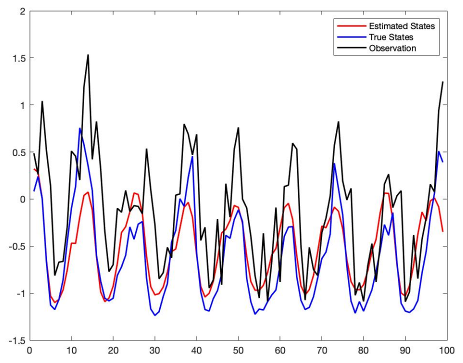
where y_k is the observation sequence, and the x_k is the states, i.e., TRF in this case. As shown in Figure 4.3, the estimated results generally follow the trend of true TRF x_k, which proves the feasibility of the network above.

4.2 TRF Estimates with MEG

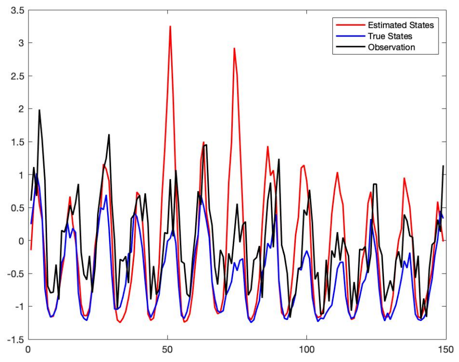
In this section, we use a dataset with simulated MEG data of the multiple speakers situation, and compare the estimation results of four different models, i.e., recursive least square (RLS), linear Gaussian state-space model (classic Kalman filter), LSTM with a linear generative model, and the deep Kalman filter model.



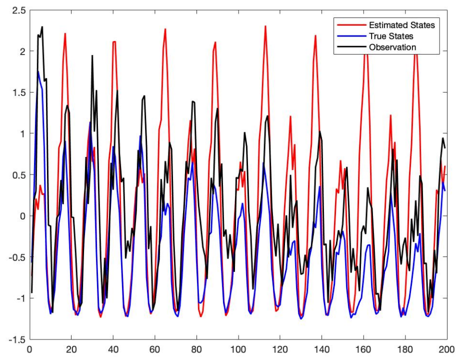
(a) 50 time steps



(b) 100 time steps



(c) 150 time steps



(d) 200 time steps

Figure 4.3: Estimates of 1D random walk

The simulated data are taken from [21]. For this MEG dataset, we use a sampling frequency of 100, the window length is 0.3 seconds, and we set the length for TRF to be 0.25 seconds, i.e., 25 samples for TRF and 30 samples for a window. As we explained in section 4.1, we want the distance between the adjacent Gabor atoms in the lag domain to be 0.05 seconds, so we cover 0.25s with 5 Gabor atoms for each speaker. Therefore, the total number of states for the TRF estimates is 10 (two speakers together).

4.2.1 Model Choice

For RLS algorithm, we map the one-dimensional observation x_k at time k to a 10 dimensional representation b_k, then normalize b_k to produce the estimate of the states. The whole process can be written as:

$$\begin{aligned} A_k &= \lambda A_{k-1} + C_k^\top C_k \\ b_k &= \lambda b_{k-1} + y_k C_k^\top \\ \theta_k &= (A_k + \gamma I)^{-1} b_k \end{aligned}$$

where $C_k = [E_k^{1\top} G, E_k^{2\top} G]$, and E_k^i is the speech envelop of i^{th} speaker, γ is l_2 regularization parameter, which is chosen to be 1 in this case. The classic Kalman filter algorithm will make the prediction as following:

$$\begin{aligned} x_k &= F_k x_{k-1} + u_k, u_k \sim \mathcal{N}(0, Q_k) \\ y_k &= C_k x_k + v_k, v_k \sim \mathcal{N}(0, R_k) \\ \hat{x}_{k|k-1} &= F_k \hat{x}_{k-1} \\ P_{k|k-1} &= F_k P_{k-1} F_k^T + Q_k \\ \hat{x}_k &= \hat{x}_{k|k-1} + P_{k|k-1} c_k^T (R_k + C_k P_{k|k-1} C_k^T)^{-1} (y_k - C_k \hat{x}_{k|k-1}) \\ P_k &= P_{k|k-1} - P_{k|k-1} C_k^T C_k P_{k|k-1} (R_k + C_k P_{k|k-1} C_k^T)^{-1} \end{aligned}$$

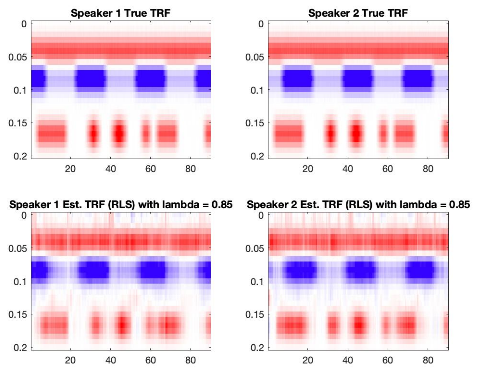
The deep Kalman filter model we used for this simulated MEG dataset is the network architecture shown in Figure 4.2, i.e., a two-layer BiLSTM RNN followed by a MLP as decoder. We also simplify the network by replacing the decoder MLP with a linear generative model $y_k = E_k^T G\theta_k + w_k$, where E_k is the vector of speech envelopes at time k, G is the Gabor matrix, and θ_k is the multi-dimensional (number of dimension depends on the number of Gabor atoms) states, so that $G\theta_k$ is the TRF estimates. In this way, we train a network that maps y_k to $\hat{\theta}_k$ and then for sampling from p_θ, we just generate $y_k = E_k^\top G\hat{\theta}_k + w_k$.

4.2.2 Estimation Results

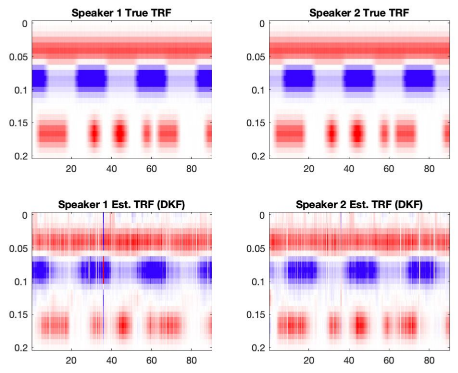
For the deep Kalman filter model and the LSTM with linear generative model, we initialize the LSTM network with the TRF estimates using RLS, and the estimates using classic Kalman filter, respectively. The TRF estimates with both initialization looks similar, so we only put the results using the TRF estimates by linear Gaussian state-space model as the initialization.

Figure 4.4 shows the results of TRF estimates using each of four models. Figure 4.4(a) is the TRF estimates made with RLS algorithm; (b) is the TRF estimates achieved with deep Kalman fitler model using the network architecture in Figure 4.2; (c) is the TRF estimates derived with two-layer BiLSTM RNN and linear generative model; and (d) is the TRF estimates derived with linear Gaussian state space model (classic Kalman filter).

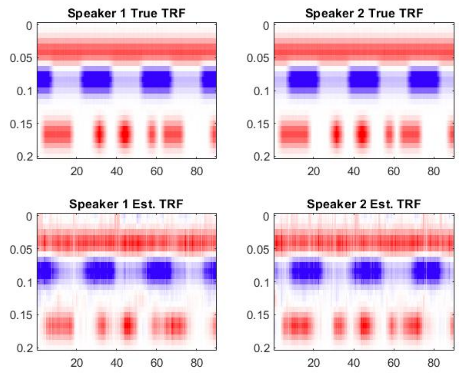
From Figure 4.4, we can see that all these methods successfully reproduce the desired encoding coefficients, as they generate the TRF estimates close to the true TRF. The complete deep Kalman filter produces the TRF estimates less smoother



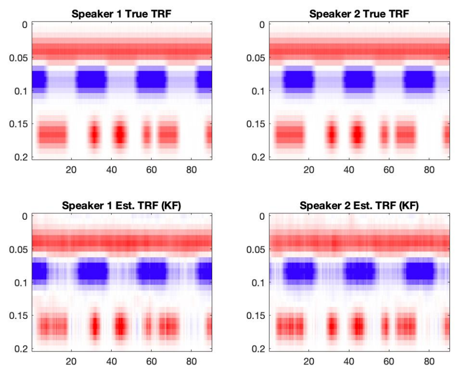
(a) TRF estimates with RLS



(b) TRF estimates with DKF



(c) TRF estimates with LSTM and linear generative model

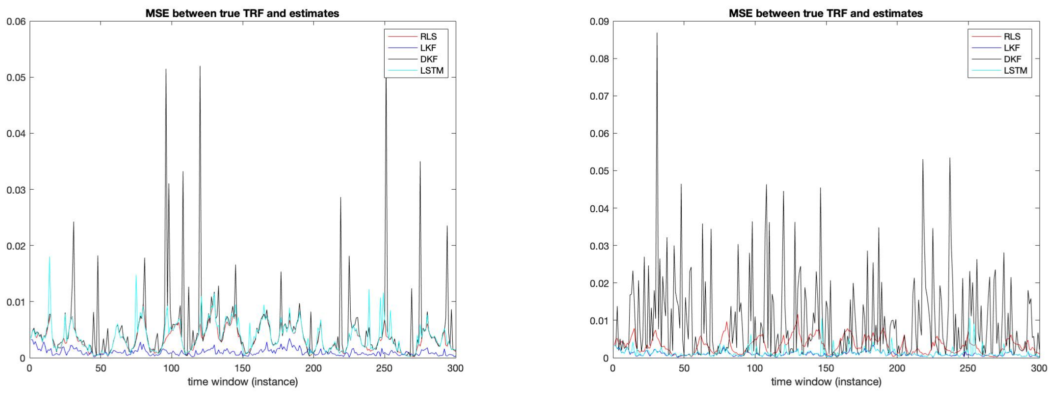


(d) TRF estimates with LKF

Figure 4.4: TRF estimates of MEG with various models

than the other three approaches. By interpreting smoothness as the Lipschitz continuity of the cost function and its gradient, a recent study proves that there is an underlying relationship between the smoothness and the long-term information retention, i.e., the larger the Lipschitz constant, the slower the decay of information retention [22].

Figure 4.5 shows the mean squared error(MSE) between the true TRF and its estimates using different methods across the time window. The left one is the MSE of estimates initialized by RLS estimates, while the right one is the results initialzied by classic Kalman filter estimates. From Figure 4.5(a), we can see that the linear



(a) Initilizaed with TRF estimates by RLS (b) Initilizaed with TRF estimates by KF

Figure 4.5: MSE between true TRF and its estimates

Kalman filter has the smallest mean squared error among all four methods. The LSTM with linear generative model has MSE almost the same as the method used to calculate its initialization, i.e., RLS algorithm. The deep Kalman filter has the highest MSE. Similarly, in Figure 4.5(b), The LSTM with linear generative model again has MSE close to the approach that produces the initialization, i.e., linear Gaussian state-space model. The complete deep Kalman filter still has the largest mean-squared error among all four models.

4.3 General Framework Simulation

4.3.1 Experiment Setup

The following generative model is employed to simulated the EEG data in dual-speaker situation:

$$e_t = w_t^{(1)}(s_t^{(1)} * h_t) + w_t^{(2)}(s_t^{(2)} * h_t) + \mu_t + u_t \tag{4.1}$$

where $s_t^{(1)}, s_t^{(2)}$ are the speech envelopes of speakers 1 and 2 at time t respectively, which are chosen from two 60s long speech segments from experiment with a sampling rate $f_s = 200\text{Hz}$. h_t in the equation represents the TRF as the impulse response of the neural process resulting in e_t, and the final impulse response is smoothed using a Gaussian kernel with a standard deviation of 10ms as shown in Figure 4.6. μ is chosen to be 0.02 as a constant mean, and u_t is a zero-mean $i.i.d$ Gaussian noise. $w_t^{(1)}, w_t^{(2)}$ are weight functions that determine the relative effect of the two speeches on the neural response. The attention will be on speaker 1 for the first half, and on speaker 2 for the second half, i.e., the weight functions are chosen to favor speaker 1 in the (0 s, 30 s) interval and speaker 2 in the (30 s, 60 s) interval. We tested on three different scenarios with decreasing separation between the attended and unattended speeches in the neural response as shown in Figure 4.7. For the first case, the high value is 1, the low value is 0.5, so this case represents the well-separated situation. In case 2, the high value is 0.9, the low value is 0.6. The separation between the attended and unattended speaker decreases, but still large enough to distinguish. In case, the high value is 0.8, the low value is 0.7. The separation further decreases, and this simulates the situation that the two speakers are almost mixed together.

Figure 4.8 shows the simulation of the auditory response of speaker 1, speaker 2, and mixed speakers respectively. For the decoder estimation, consecutive non-overlapping windows of length 0.25s are used, which leads to K = 240 windows of length W = 50 samples. The effective data length is 5s for decoder estimation. The forward lags of the neural response have been limited to a 0.4s window, i.e., $L_d = 80$ samples.

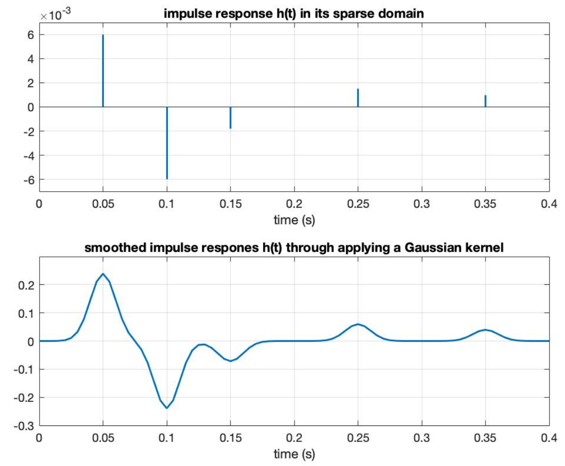


Figure 4.6: TRF as the Impulse response h_t

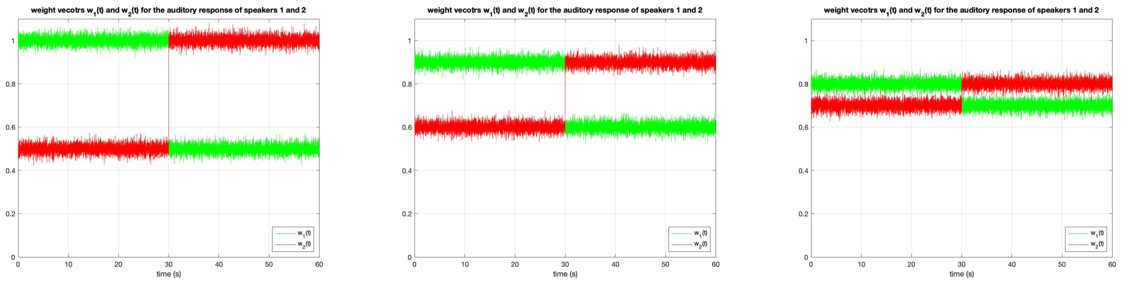


Figure 4.7: Weight functions

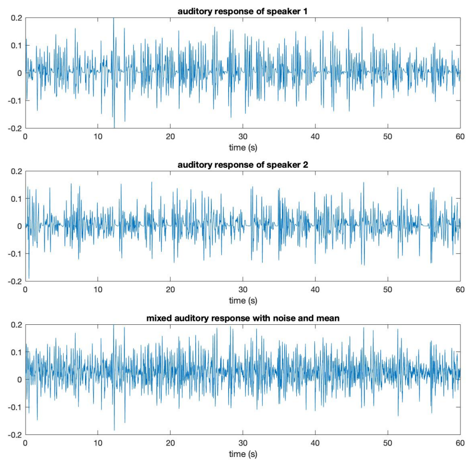
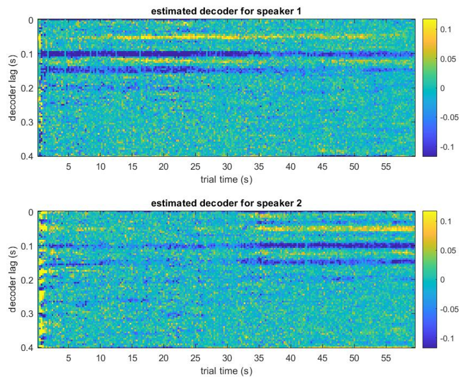


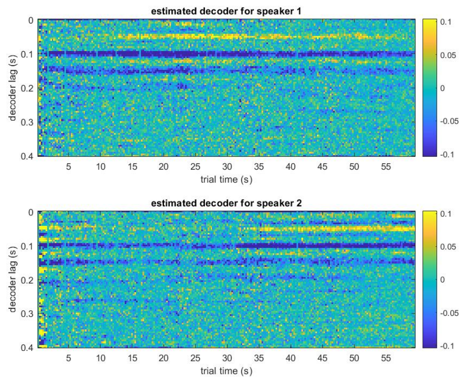
Figure 4.8: auditory response

4.3.2 Estimation Results

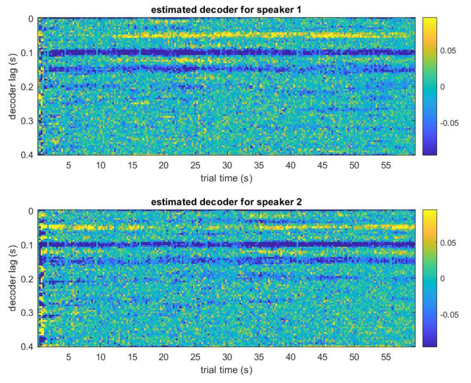
The TRF estimates in Figure 4.9 are trained with max epoch 10 for the BiLSTM RNN, while the estimates in Figure 4.10 are trained with max epoch 100.



(a) Case 1: large separation distance



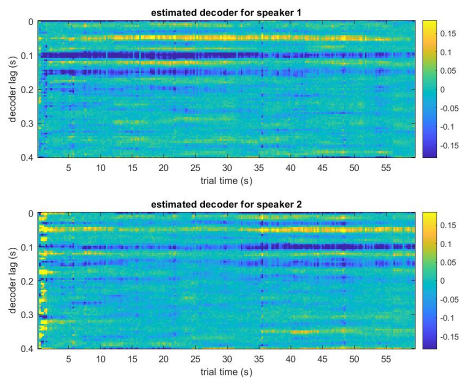
(b) Case 2: medium separation distance



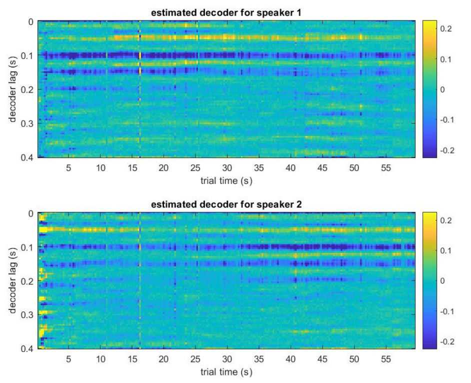
(c) Case 3: small separation distance

Figure 4.9: TRF estimates using Deep Kalman Filter with max epoch 10

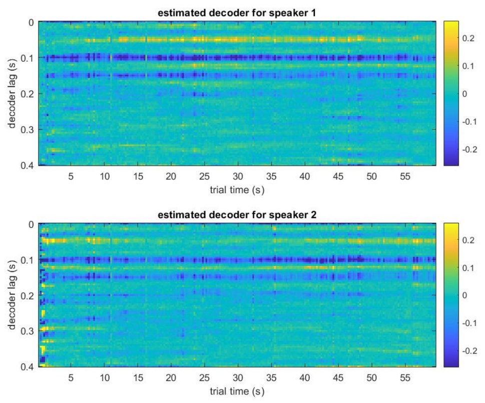
If we trained the network with more epochs, i.e, 100 epochs in this case, the TRF estimates are more clear and robust as shown in the above figures. Thus, all the following discussions and results are based on the network trained with max epoch 100.



(a) Case 1: large separation distance



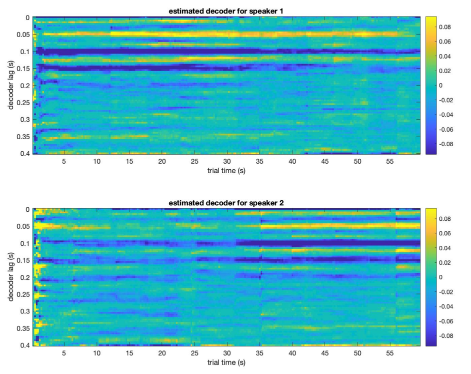
(b) Case 2: medium separation distance



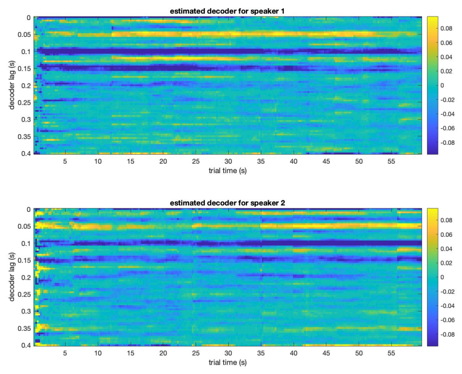
(c) Case 3: small separation distance

Figure 4.10: Decoder estimate using Deep Kalman Filter with max epoch 100

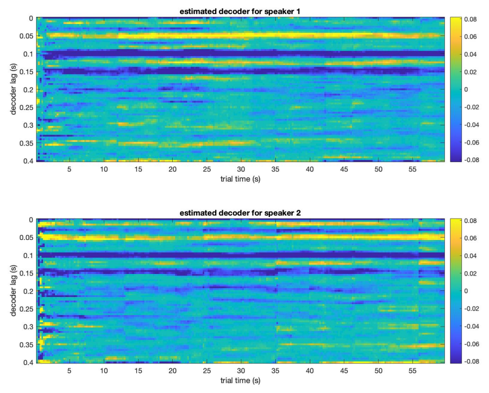
As stated in the experiment setup, the simulated attention is on speaker 1 for the first 30 seconds, and on speaker 2 from 30s to 60s. According to our simulation setup, there are supposed to be significant components near the 50ms, 100ms, 150ms lag. In case 1, where the influences of attended and unattended speakers in the neural response are well separated, we can see that the significant components are obvious in decoder estimate of speaker 1 for first 30 seconds, and become less significant in last 30 seconds as the attention switches from speaker 1 to speaker 2. Similarly, after the attention switches to speaker 2, the significant components of speaker 2 decoder estimates become larger and visible. Therefore, these significant components of decoder estimates are modulated by the attentional states, where the effects of modulation decrease as the separation between the attended and unattended speaker decreases. In case 2, one can still see the weakened influence of the modulation around 30s where the attention is switched, while in case 3 the modulation effect almost disappears.



(a) Case 1: large separation distance



(b) Case 2: medium separation distance

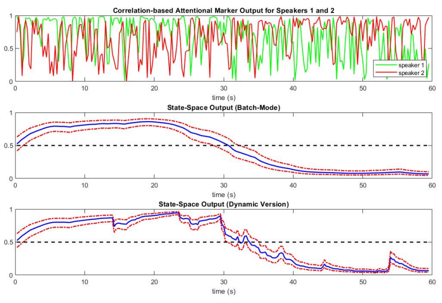


(c) Case 3: small separation distance

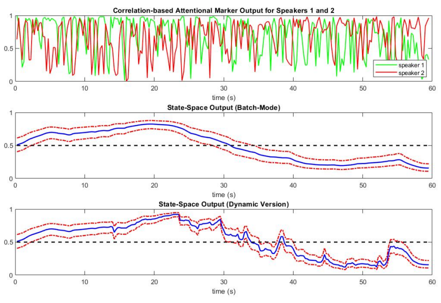
Figure 4.11: decoder estimates using RLS

Figure 4.11 is the estimation of the decoder with RLS algorithm. The decoder estimates using RLS are almost the same as that using deep Kalman filter model. In case 2, the RLS based decoder estimate of speaker 1 doesn't have an obvious change around 30s, where the speaker 1 estimate using deep Kalman filter reflects the attention switch. The estimate of speaker 2 using deep Kalman filter also outperforms that using RLS a little bit around the significant components at 100ms lag for first 30 seconds. In addition, the RLS based estimator is hard to distinguish two speakers in case 3, while we can still see some pattern of attention switch using deep Kalman filter. Therefore, we can say that the deep Kalman filter improves the TRF estimates in vague cases when the separation between attended and unattended speakers is not that large. This is very useful in the real world settings as the sound sources may only have small separation.

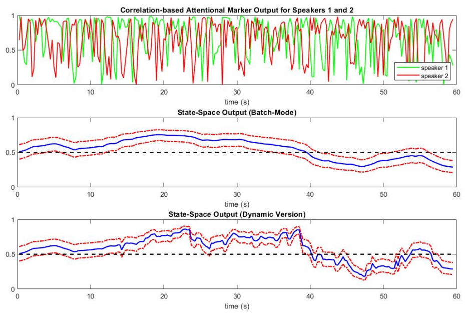
Figure 4.12 displays the correlation-based attention marker output for both speakers, and corresponding state space output of them. The first row in Figure 4.12 is the output of a correlation-based attention marker for speaker 1 and 2, which is calculated as $m_k^{(i)} = |corr(\boldsymbol{y}_k^{(i)}, \boldsymbol{X}_k \hat{\boldsymbol{\theta}}_k)^{(i)}|$, where $i = 1, 2$ denotes the speaker, $k = 1, ..., K$ is the index of time window. The second row is the output of the batch mode state-space estimator of the correlation-based attention marker, while the third row shows the results of the real-time state-space estimator. As expected, the correlation-based attention marker output of speaker 1 is higher than that of the speaker 2 in (0s,30s) time interval, while the output of speaker 2 is higher in (30s,60s) time window. In other words, the behaviour of correlation-based attention marker can be used to represent the attentional state. However, the accuracy also decreases as the separation decreases, which reflects the necessity of state-space model. Comparing the second and third row in Figure 4.12, it is not surprising to see that the batch mode state-space model gives more robust outputs than the dynamic state space estimator. The batch-mode estimator has access to all the



(a) Case 1: large separation distance



(b) Case 2: medium separation distance

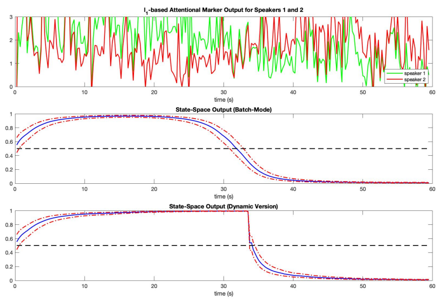


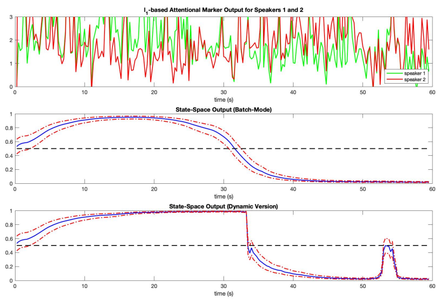
(c) Case 3: small separation distance

Figure 4.12: Correlation-based attention marker (DKF)

attention markers, while the real-time estimator only refers to a limited number of attention markers. Therefore, the stochastic fluctuation affects the output of the real-time state-space model more than that of the batch-mode estimator. However, the general performance of two state-space models matches each other, except the fluctuation's influence on the real-time estimator. The classification accuracy also decreases as the separation between two speakers decreases. Moreover, the real-time estimator misclassifies more than the batch-mode state-space does.

Figure 4.13 shows the results of l_1 norm based attention marker. In this case, the attention marker is calculated as the l_1 norm of the decoder, i.e., $m_k^{(i)} = \|\hat{\boldsymbol{\theta}}_k^{(i)}\|_1$, where $i = 1, 2$ is the speaker, and $k = 1, .., K$ denotes the time window. The attended TRF is supposed to exhibit significant and informative components of the neural response, while the unattended decoder coefficients is expected to be small and randomly distributed since it shouldn't contain much information. The first row


(a) Case 1: large separation distance


(b) Case 2: medium separation distance

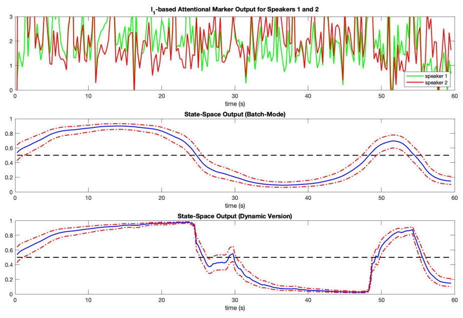
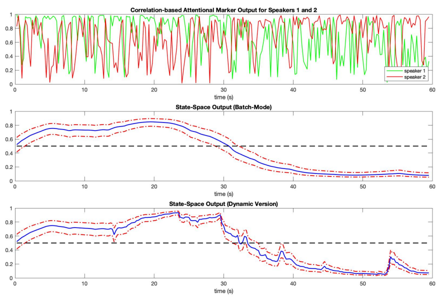

(c) Case 3: small separation distance

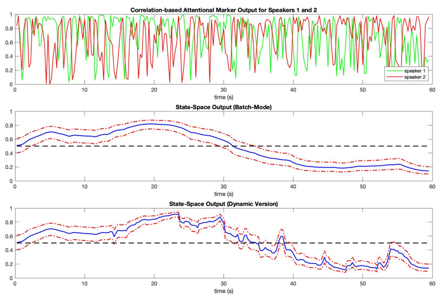
Figure 4.13: l_1 norm-based attention marker (DKF)

in Figure 4.12 displays the l_1 norm attention marker output, the second row is the batch-mode state space model output, and the last row is the dynamic state-space output. Same as the results of correlation based attention marker, the batch-mode estimator produces more robust output than the real-time estimator does. In addition, the real-time estimator performs similar to the batch-mode estimator, despite the stochastic fluctuations of the dynamic state-space model. Also the classification accuracy decreases from case 1 to case 3, i.e., as the separation between the attended and unattended speaker decreases. In general, the correlation based attention marker and the l_1 norm based attention marker have similar performance, but the estimates are smoother using the l_1 norm based attention marker.

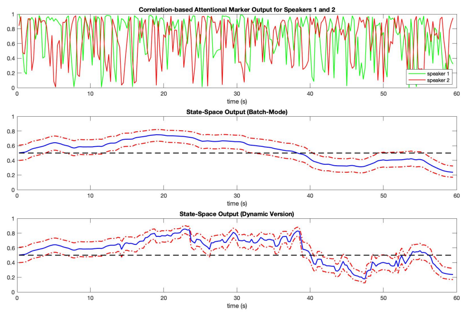
All these traits shown in the above results also appear in the results produced by RLS algorithm as shown in Figure 4.14. The batch-mode state space output and the real-time estimator output in all three cases, as well as the attention marker output in the first case, are nearly the same using RLS and deep Kalman filter. The attention marker output, as we discussed above, can be treated as a representative of attention state, where the output would be higher when attended. With RLS algorithm, two speakers have similar output in case 3, but the averaged output of speaker 2 is higher in last 30 seconds if we use the deep Kalman filter.



(a) Case 1: large separation distance



(b) Case 2: medium separation distance



(c) Case 3: small separation distance

Figure 4.14: Correlation-based attention marker (RLS)

Chapter 5: Discussion

In this thesis, we first discussed the deep Kalman filter model with the key technique it employed, i.e., a recognition network used as the variational inference model. Then we gave an overview of the estimation of temporal response functions and a general framework utilizing the TRF estimates in the real-time tracking of the selective auditory attention [4].We proposed a new model that replaces the first module in the general framework with the deep Kalman filter model. The replaced module is used to estimate the TRFs, while the deep Kalman filter model in our case is implemented with a two-layer bi-directional LSTM (BiLSTM) RNN and a multi-layer perceptron with four convolution layers. The performance is validated on both simulated MEG and EEG data, for the accuracy of TRFs estimates and the overall classification, respectively.

In the simulated MEG case, the deep Kalman filter implemented in this work has the highest mean-squared error (MSE), while the conventional Kalman filter results in the lowest mean-squared error. The RLS algorithm leads to higher MSE than the linear Kalman filter. The BiLSTM RNN with a linear generative model will have a MSE curve close to the initialization of its BiLSTM network. Thus, if we initialized the BiLSTM RNN with the estimation results of the classic Kalman filter, then this model will have the second lowest MSE that is almost the same as that of the linear Kalman filter. Similarly, if we initialized the BiLSTM RNN with the estimation results of the RLS algorithm, then this model will have a mean-squared error curve almost the same as that of the RLS algoirthm. The only difference between the deep Kalman filter and the Bi-LSTM RNN with linear generative mdoel is the prediction step utilizing the TRF estimates. The lower mean-squared error may due to the stability of the linear mapping. With linear prediction of the observation, the Bi-LSTM RNN enhances the TRF estimation results compared to RLS if we initialize the network with TRF estimates from linear Gaussian state-space model. In the future, we have to further fine-tune the decoding part of the deep Kalman filter for better estimation.

In the simulated EEG case, the general traits of the output are the same using RLS algorithm and the deep Kalman filter. More explicitly, both models produce outputs that are more robust in batch-mode estimator than in real-time estimator, while the outputs of both estimators have the same trends. In addition, the effect of attention switch decreases as the separation between the attended and unattended speaker decreases. However, the modulation effect is clearer in deep Kalman filter results than in RLS results when the speakers are not well-separated. This result suggests that utilizing the deep Kalman filter in real-time tracking of the selective auditory attention may lead to a better performance in the real-world scenarios.

Bibliography

[1] S. Getzmann, J. Jasny, and M. Falkenstein, "Switching of auditory attention in "cocktail-party" listening: ERP evidence of cueing effects in younger and older adults," *Brain and Cognition*, vol. 111, 02 2017.

[2] S. Getzmann and R. Näätänen, "The mismatch negativity (MMN) as a measure of auditory stream segregation in a simulated "cocktail-party" scenario: Effect of age," *Neurobiology of aging*, vol. 36, 07 2015.

[3] S. Evans, C. McGettigan, Z. K. Agnew, S. Rosen, and S. K. Scott, "Getting the cocktail party started: Masking effects in speech perception," *Journal of Cognitive Neuroscience*, vol. 28, no. 3, pp. 483–500, 2016.

[4] S. Miran, S. Akram, A. Sheikhattar, T. Zhang, and B. Babadi, "Real-time tracking of selective auditory attention from M/EEG: A bayesian filtering approach," 11 2017.

[5] S. Shamma, M. Elhilali, and C. Micheyl, "Temporal coherence and attention in auditory scene analysis," *Trends in neurosciences*, vol. 34, pp. 114–23, 12 2010.

[6] N. Ding, "Emergence of neural encoding of auditory objects while listening to competing speakers," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 109, pp. 11854–9, 07 2012.

[7] R. G. Krishnan, U. Shalit, and D. A. Sontag, "Deep Kalman Filters," *ArXiv*, vol. abs/1511.05121, 2015.

[8] J. Paisley, D. Blei, and M. Jordan, "Variational bayesian inference with stochastic search," *Proceedings of the 29th International Conference on Machine Learning, ICML 2012*, vol. 2, 06 2012.

[9] D. Jimenez Rezende, S. Mohamed, and D. Wierstra, "Stochastic backpropagation and approximate inference in deep generative models," 01 2014.

[10] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *CoRR*, vol. abs/1312.6114, 2014.

[11] R. G. Krishnan, U. Shalit, and D. Sontag, "Structured inference networks for nonlinear state space models," in *AAAI*, 2017.

[12] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *International Conference on Learning Representations*, 12 2014.

[13] A. Amberkar, P. Awasarmol, G. Deshmukh, and P. Dave, "Speech recognition using recurrent neural networks," pp. 1–4, 03 2018.

[14] I. Sutskever, O. Vinyals, and Q. Le, "Sequence to sequence learning with neural networks," *Advances in Neural Information Processing Systems*, vol. 4, 09 2014.

[15] Q. You, H. Jin, Z. Wang, C. Fang, and J. Luo, "Image captioning with semantic attention," pp. 4651–4659, 06 2016.

[16] Z. Wu, T. Yao, Y. Fu, and Y.-G. Jiang, "Deep learning for video classification and captioning," in *Frontiers of Multimedia Research*, 2018.

[17] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Trans. Signal Process.*, vol. 45, pp. 2673–2681, 1997.

[18] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional lstm networks," in *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, vol. 4, pp. 2047–2052 vol. 4, 2005.

[19] S. Akram, J. Z. Simon, and B. åBabadi, "Dynamic estimation of the auditory temporal response function from meg in competing-speaker environments," *IEEE Transactions on Biomedical Engineering*, vol. 64, no. 8, pp. 1896–1905, 2017.

[20] J. O'Sullivan, A. Power, N. Mesgarani, S. Rajaram, J. Foxe, B. Shinn-Cunningham, M. Slaney, S. Shamma, and E. Lalor, "Attentional selection in a cocktail party environment can be decoded from single-trial EEG," *Cerebral cortex (New York, N.Y. : 1991)*, vol. 25, 01 2014.

[21] S. Miran, J. Z. Simon, M. C. Fu, S. I. Marcus, and B. Babadi, "Estimation of state-space models with gaussian mixture process noise," in *2019 IEEE Data Science Workshop (DSW)*, pp. 185–189, 2019.

[22] A. H. Ribeiro, K. Tiels, L. A. Aguirre, and T. B. Schön, "Beyond exploding and vanishing gradients: analysing rnn training using attractors and smoothness," in *Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics (AISTATS).*, vol. 108, PMLR, 2020.