# Extrapolative Continuous-time Bayesian Neural Network for Predictive Streaming Domain Adaptation

**Anonymous Author(s)**
Affiliation
Address
email

## Abstract

This work is inspired by internal predictive modeling in neuroscience. Such a mechanism allows humans to process non-stationary streaming sensory data with remarkably low latency and high precision by continuously and instantly generating internal predictive models. This leads us to rethink why existing approaches in unsupervised domain adaptation (UDA) fail to achieve what humans are capable of when handling the streaming data. In this paper, we explore the possibility of introducing such a mechanism in UDA. We propose to formulate predictive modeling as a continuous-time Bayesian filtering problem. We aim to infer latent states of a dynamical system that describes the dynamics of model parameters of a UDA model progressively receiving non-stationary streaming data. To this end, we then propose extrapolative continuous-time Bayesian neural networks (ECBNN), which generalize existing Bayesian neural networks to represent temporal dynamics and allow us to extrapolate the distribution of model parameters over the time axis when only historical streaming data is available. Remarkably, our empirical results show that ECBNN is capable of continuously generating better distributions of model parameters along the time axis given historical data only, thereby achieving (1) gradually improved alignment between the source and target features, and (2) gradually improved model performance over time during the real-time testing stage.

## 1 Introduction

Human consciousness requires building an internal predictive model that describes the external world [1, 2]. Numerous neuroscience studies have suggested that humans continually update their internal predictive model to represent the statistics of the fluctuating environments accurately [3, 4, 5, 6]. Internal predictive modeling is crucial for humans perceiving continuous stream of sensory inputs (e.g. sound and sight) in real time given constraints imposed by neural transmission delays [7]. For example, to catch a fast-moving object, the internal predictive model is required to forecast its trajectory ahead to compensate for the response latency of neurons; to recognize words from lip movements at various view angels in real time, again, the internal predictive model is required to ensure that decision outcomes are not delayed [8].

Such a mechanism allows humans to process non-stationary streaming sensory data with remarkably low latency and high precision by continuously and instantly generating internal predictive models. In contrast, most existing UDA algorithms either require collecting additional data from a large amount of increasingly more domains [9] or using streaming data batches for fine-tuning [10, 11], which is impractical for real-time streaming situations. This leads us to rethink why existing approaches in

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

unsupervised domain adaptation (UDA) fail to achieve what humans are capable of when handling the streaming data.

This work aims to introduce "internal predictive modeling" to augment traditional UDA algorithms in handling non-stationary streaming data. UDA augmented with such a mechanism should be able to instantly extrapolate the model parameters on the fly throughout the real-time testing stage, rather than merely performing offline adaptation or online adaptation with unacceptable latency due to data collection and training or fine-tuning.

To achieve this, we propose to formulate predictive modeling as a continuous-time Bayesian filtering problem. More specifically, we model the neural network (NN) parameters of UDA evolving with non-stationary streaming data as a continuous-time stochastic dynamical system. We aim to infer model distributions over time given the historical streaming data.

To our knowledge, such Bayesian treatment of time-evolving NN parameters is mostly unexplored. Existing continuous-time Bayesian filtering approaches, e.g. CTBN [12], assume discrete-state for the dynamical system; and thus cannot scale to NNs. Furthermore, most of Bayesian neural networks, including SDE-BNN [13] are designed to reason about static distributions of NN parameters. In this paper, we propose extrapolative continuous-time Bayesian neural network (ECBNN), which generalizes existing static BNNs to represent temporal dynamics and allows us to extrapolate the distribution of model parameters over the time axis when only historical streaming data is available. Specifically, by adopting ECBNN as encoder of UDA for streaming data, the new model is capable of efficiently inferring multi-step ahead model distributions before observing the incoming data, therefore effectively reducing the latency. The multi-step ahead Bayesian inference is achieved by developing a particle-filter-like probabilistic differential equation, dubbed Particle Filter Differential Equation (PFDE), which allows reasoning about time-evolving model distributions suitable for handling each incoming data point. Such distributions are produced by efficiently solving PFDE. Furthermore, PFDE assumes a continuous-time Markov chain for the dependency of model distributions, which can be evaluated in real-time, allowing us to align source and target temporal domains in the entire timeline by optimizing an upper bound for temporal-domain-invariant representation learning.

## 2 Related Work

**Unsupervised Domain Adaptation with Streaming Data.** There is a vast body of prior work [14, 15, 16, 17, 18, 19] exploring domain-invariant representation learning for the purposes of unsupervised domain adaptation (UDA). These works' key idea is harnessing expressive power of neural networks to transform the input data to latent representations that are further aligned across static domains. However, in real-world streaming applications, streaming data usually follows non-stationary distributions. Consequently, these work and a few other offline adaptation methods [20, 21, 22] mostly fail to handle such data due to either the assumption of static domains or the requirements of low latency.

The UDA for streaming data with non-stationary distribution have been studied before but in different problem settings with different focuses. For example, continuous manifold adaption (CMA) [9] addresses this problem by adapting the model with the additional unlabeled data from a fixed number of pre-defined intermediate domains; predictive domain adaptations [23, 10] attempt to solve a specific scenario where target data is unavailable but the meta data that characterizes the target domain is provided. However, these methods cannot be directly applied to our problem settings due to the unavailability of the meta-data, e.g. absolute time stamps of each data sample. The closest work to ours is [11], in which a meta adaptation framework is introduced to extract the temporal-domain-invariant representation from the streaming data. However, [11] requires collecting partial streaming data for fine-tuning, greatly limiting the applicability to real-time streaming situations. In contrast, our work derive a new PF-based differential equation, allowing to continuously infer distributions of NN parameters from historical streaming data and achieve much lower latency and temporal-domain-invariant feature learning during real-time testing.

**Bayesian Neural Networks with Neural Differential Equations.** Bayesian neural networks (BNNs) [24] aim to infer the posterior distribution of the NN parameters given observations and a prior imposed on these parameters. With a different aim, early work in Neural differential equations, e.g. neural ordinary differential equation (ODE) [25], focuses on adopting non-Bayesian NNs to parameterize the dynamics involved in differential equations over depth or layers. Bayesian Neural ODE [26, 27] improve upon Neural ODE by using BNNs to achieve a similar goal. Unlike the above studies, this work aims to develop a BNN that captures the evolution of NN parameters over time. The closest work to ours is [13], which focuses on building a continuous-depth BNN using stochastic differential equations. However, their model assumes a static distribution for NN parameters; and thus cannot scale to capture the time-varying distribution of NN parameters. Addressing this issue is one of our model's focuses. (Please refer to the Supplement for related work on particle filtering with neural networks)

# 3 Method

In this section, we first review the background of particle filtering, formalize the problem of predictive streaming adaptation for non-stationary streaming data, and then describe our methods for addressing the problem.

## 3.1 Background: Particle Filtering

Particle filtering (PF) provides an effective solution for state estimation problem of dynamical systems, in which sequential importance sampling (SIS) is used to approximate the evolution of the posterior distribution of the state given observations, $p(\theta_t|x_{1:t})$, where $\theta_t$ represents the latent state at time $t$ and $x_{1:t}$ denotes observations from time 1 to $t$. PF adopts a weighted set of particles $\{(w_t^{(j)}, \theta_t^{(j)})\}_{j=1}^k$ to construct an approximate posterior:

$$q(\theta_t|x_{1:t}) = \sum_{j=1}^{k} w_t^{(j)} \delta(\theta_t - \theta_t^{(j)}),\ \theta_t^{(j)} \sim q(\theta_{1:t}), \tag{1}$$

where $\delta$ denotes the Dirac-delta function; $\theta_t^{(j)}$ is the $j$-th particle (sample) drawn from an approximate distribution called importance distribution $q(\theta_{1:t})$, which can be assumed to have a simple parametric form, e.g. a Gaussian; $w_t^{(j)}$ is the corresponding importance weight.

By assuming a Markov chain for both the importance distribution and the state space as well as using transition distribution $p(\theta_t|\theta_{t-1})$ as the importance distribution, one can construct a bootstrap filter [28] with simplified importance weights as: $w_t^{(j)} \propto w_{t-1}^{(j)} p(x_t|\theta_t)$, where $p(x_t|\theta_t)$ is the likelihood and $\sum_j w_t^{(j)} = 1$.

## 3.2 Problem Setting: Predictive Streaming Domain Adaptation

**Problem.** We consider the unsupervised domain adaptation setting for non-stationary streaming data. More specifically, given a source stream $\{\mathbf{B}_i^s\}_{i=1}^I$, a target stream $\{\mathbf{B}_i^a\}_{i=1}^I$ and a testing stream $\{\mathbf{B}_i^e\}_{i=1}^I$, where $I$ denotes the number of data batches; the source data batches $\mathbf{B}_i^s = \{\mathbf{x}_t^s, \mathbf{y}_t^s\}_{t=(i-1)\cdot N}^{i\cdot N-1}$ are labeled, while the target and testing data batches $\mathbf{B}_i^a = \{\mathbf{x}_t^a\}_{t=(i-1)\cdot N}^{i\cdot N-1}$, $\mathbf{B}_i^e = \{\mathbf{x}_t^e\}_{t=(i-1)\cdot N}^{i\cdot N-1}$ are unlabeled, where $N$, the number of frames in each batch, is usually small for machine learning applications that require streaming inputs, e.g. streaming video processing systems. Note that the target and test stream share the same categorical domain, while source stream differs from them. We first train a UDA based on the source and target streams $\{\mathbf{B}_i^s\}_{i=1}^I$ and $\{\mathbf{B}_i^a\}_{i=1}^I$. Based on such a model, we aim to perform adaptation as each testing stream data batch $\mathbf{B}_i^e$ arrives.

**Low Latency and Timeline.** Notably, during the real-time testing stage, our problem setting has strict requirements for the latency caused by adaptation. Therefore, existing offline adaptation methods or online adaptation methods that require training or fine-tuning, are not applicable to our problem setting. Furthermore, the source stream, target stream and testing stream may not necessarily share the same timeline and the absolute time stamps of the data are unknown; and thus DA methods [18, 29] that require absolute time stamps fail to handle our problem setting as well.

### 3.3 Overview

Below we provide an overview of our method (Fig. 1).

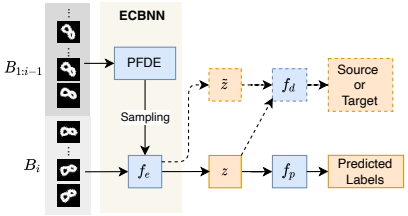


Figure 1: General framework of predictive streaming domain adaptation; the components marked with dotted line are removed during testing.

**Training.** Our approach adopts both source stream and target stream data for training: (i) it starts by adopting PFDE, taking as input the historical streaming batches $\mathbf{B}_{1:i-1}$, to infer the posterior distributions of NN parameters of the encoder $f_e$; based on such distributions, particles are sampled to update the encoder $f_e$ (updated before processing $\mathbf{B}_i$ ); (ii) it further take the updated encoder to generate $z$, the encoder output for $\mathbf{B}_i$, and $\widetilde{z}$, the encoder output for the approximated data required for calculating the proposed analytical upper bound (see Sec.3.7 for more details of the approximated data ); (iii) finally, we follow the typical adversarial UDA training procedure to evaluate the training loss given $z$ and $\widetilde{z}$ and update the whole framework (see Sec. 3.6 for more detailed description).

**Real-time Testing** During the real-time testing stage, the testing stream cannot be revisited and should be processed as each data batch arrives. The testing procedure is mostly similar to the training, except that discriminator are removed.

### 3.4 Particle Filter Differential Equation (PFDE)

Traditional PF methods, e.g. bootstrap filters, require calculating likelihood for observations at each time point, which is infeasible in our problem setting since only historical data is available when performing multi-step ahead Bayesian inference. We therefore develop, by the following theorem, a new type of differential equation (DE), dubbed Particle Filter Differential Equation (PFDE), that describes the evolution of the state in a continuous-time stochastic dynamical system.

**Theorem 1.** ***(Informal)*** *Suppose we are given a time series $\{\mathbf{x}_t\}_{t=\tau}^{n\tau}$, where $\tau$ is small constant. Assume $\{\mathbf{x}_t\}_{t=\tau}^{n\tau}$ are conditionally independent given a sequence of latent states $\{\theta(t)\}_{t=\tau}^{n\tau}$, which is itself assumed to be Markovian. Assume marginal distribution of the state $p(\theta(t))$ is stationary. Let $p(\theta(t)|\theta(t-\tau))$ be the transition distribution. Let $w(t)^{(j)}$ be the importance weight. Then we have:*

$$\left(\log w(t)^{(j)}\right)'' = \lim_{\tau \to 0} \left[ C \left( \log p(\theta(t)^{(j)} | \theta(t-\tau)^{(j)}) \right)'' \right],$$

where $(\cdot)'$ and $(\cdot)''$ denote the first-order and second-order temporal derivatives respectively. The derivation of PFDE in Theorem 1 (See the Supplement for the proof) is based on the bootstrap filter. By assuming the marginal distribution of the state $p(\theta(t))$ as a stationary distribution (which is a mild assumption as conditional distribution can still be non-stationary), we successfully build the *link between the transition distribution and the importance weight*, whose calculation no longer requires calculating the intractable likelihood term.

### 3.5 Extrapolative Continuous-time Bayesian Neural Networks (ECBNN)

At ECBNN's core is a PFDE, which provides a Bayesian treatment of time-evolving NN parameters and inherits several advantages from both particle filters and differential equations. For instance, PFDE enables inferring posteriors of NN parameters along the time axis by solving itself at any given time points.

Specifically, due to the intractable parametric form of the transition distribution $p(\theta_t|\theta_{t-1})$ in our setting, we use an approximate distribution $\widetilde{q}(\theta(t))$ as the importance distribution. Let $w(t)^{(j)}$ be the importance weight; let $\theta(t)^{(j)}$ be the particle and let $\widetilde{q}(\theta(t)) = \mathcal{N}(m(t), s(t))$ be the importance distribution, where $m(t)$ and $s(t)$ are the mean and variance of the Gaussian. By continuous-time Markovian assumption and Theorem 1, the PFDE can be written as:

$$f(m(t), m'(t), m''(t), s(t), s'(t), s''(t), \theta(t)^{(j)}) = (\log(w(t)^{(j)})'', \tag{2}$$

where $f(\cdot)$ is a nonlinear function, whose detailed form is described in the Supplement. Let the solution of PFDE be $G(t) = \{m(t), s(t), \{w(t)^{(j)}\}_{j=1}^{k}\}$. Then ECBNN takes the following form to approximate the posterior of the NN parameters $q(\theta(t))$:

$$q(\theta(t)) = \sum\nolimits_{j=1}^{k} w(t)^{(j)} \delta(\theta(t) - \theta(t)^{(j)}), \quad \theta(t)^{(j)} \sim \widetilde{q}(\theta(t)) = \mathcal{N}(m(t), s(t)). \tag{3}$$

We further demonstrate the inference and learning of ECBNN by applying it to address the problem of predictive streaming DA.

### 3.6 Application of ECBNN for Predictive Streaming Domain Adaptation

In this section, we demonstrate how to embed ECBNN into a typical UDA framework to perform predictive streaming DA.

**Learning.** An UDA usually contains an encoder $f_e$ to project the input to a domain-invariant space $\mathbf{z}$ with the help of an discriminator $f_d$, such that the target data can be accurately predicted by the predictor $f_p$. Let $\theta$ be the parameters of the encoder $f_e$, an UDA performs a minimax optimization with the value function $V(f_e, f_d, f_p)$ as:

$$\min_{f_e(\cdot;\theta), f_p} \max_{f_d} V_p(f_e(\cdot;\theta), f_p) - \lambda_d V_d(f_d, f_e(\cdot;\theta)), \tag{4}$$

where $V_p$ is the expectation of the prediction loss (e.g., crossentropy loss for classification tasks) over the source data batches; $\lambda_d$ is a hyperparameter balancing both losses ;$V_d$ is the expectation of the discriminator loss over source and target data batches. For simplicity, we only emphasize the parameters of the encoder $f_e$ and omit other parameters.

We extend an UDA with ECBNN to handle predictive streaming DA by adopting ECBNN as encoder of UDA. The architecture of our model is shown in Fig. 1. Recall that ECBNN adopts an approximate distribution $\widetilde{q}(\theta(t))$ as the importance distribution due to the intractability of the transition distribution. This leads us to adopt variational inference to optimize our model, in which $\widetilde{q}(\theta(t))$ is treated as the variational distribution of the transition distribution. Let $p(\theta(t))$ be the prior. Let $l_u(\cdot;\theta, f_p, f_d)$ be the function inside the minimax optimization of Eq.(4). Since both terms in $l_u$ can be treated as negative log-likelihood, we directly combine their negation's expectation with KL terms as the evidence lower bound (ELBO), which can be written as:

$$l_b = \sum\nolimits_{t=1}^{I\times N} (-\mathrm{KL}(\widetilde{q}(\theta(t))||p(\theta(t))) - \sum\nolimits_{j=1}^{k} w(t)^{(j)} l_u(\cdot;\theta(t)^{(j)}, f_p, f_d). \tag{5}$$

We further include in the ELBO the following additional loss term $l_{de}$, which aims to learn the NN parameters involved in PFDE by attempting to satisfy the DE defined in Eq. (2):

$$l_{de} = \sum\nolimits_{t=1}^{I\times N} |f(m(t), m'(t), m''(t), s(t), s'(t), s''(t), \theta(t)^{(j)}) - (\log(w(t)^{(j)}))''|^2. \tag{6}$$

In the following subsections, we will show how to parameterize the variables and its temporal derivatives involved in DE using neural networks.

**Inference.** The key to reducing latency is to force the inference of variables involved in PFDE to depend on historical batch data only. Specifically, suppose we are given up to $i-1$-th batch $\mathbf{B}_{1:i-1}$, we aim to infer the encoder distribution for each frame of the incoming batch through solving the PFDE. However, existing ODE numerical solvers fail to give the approximate solution as the initial or boundary values of the variables are computationally intractable. Inspired by [25, 30], we adopt neural networks to approximate the solution of these variables, such that:

$$m(t) = f_{\theta_m}(t, \mathbf{B}_{1:i-1}), \quad s(t) = f_{\theta_s}(t, \mathbf{B}_{1:i-1}), \quad \log w(t)^{(j)} = f_{\theta_w}(t, \mathbf{B}_{1:i-1}, \theta(t)^{(j)}), \tag{7}$$

where $f_{\theta_*}$ are neural networks ( implementations detail in the Supplement). To process $\mathbf{B}_{1:i-1}$, $f_{\theta_*}$ contains a recurrent neural network (RNN), which maintains hidden states to represent the historical batches, without need to repeatedly process the sequence from the start. Notably, our solution doesn't require the absolute time stamps, e.g. to solve for the encoder distribution for the incoming batch

at the first frame, we can set $t = 1$. Furthermore, we specifically design these neural networks as a non-linear function of time $t$, allowing all temporal derivatives required by Eq. (6) to be computed by the chain rule and evaluated by means of automatic differentiation [31].

**Prior.** We use NNs to parameterize the mean $m_0(t)$ and variance $s_0(t)$ of the prior $p(\theta(t))$:

$$m_0(t) = f_{\theta_{m0}}(\mathbf{B}_i), \quad s_0(t) = f_{\theta_{s0}}(\mathbf{B}_i), \tag{8}$$

where $f_{\theta_*}$ are neural networks (implementations detail in the Supplement). We specifically use the current batch $\mathbf{B}_i$ to estimate the parameters of the prior during training. Consequently, KL term in the ELBO tends to encourage the posterior to extrapolate the future based on what has observed before. It's worth noting that this will not affect the latency as prior will be removed during testing.

### 3.7 Toward Temporal-domain-invariance for the Entire Time Line

Similar to particle filters, PFDE is also capable of capturing the latent time-varying distribution of the non-stationary streaming data. Therefore, the solution of PFDE $G(t) = \{m(t), s(t), \{w(t)^{(j)}\}_{j=1}^{k}\}_0^{+\infty}$ can be treated as a stochastic process that describes the generation mechanism of the non-stationary data. To impose the discriminator to consider such inductive bias and encourage our model to generalize to unseen streaming data, we give a temporal-domain-invariant loss that is the expectation of the discriminator loss $V_d$ over the stochastic process $G(t)$:

$$l_i = \int_0^{+\infty} \sum\nolimits_{j=1}^{k} w(t)^{(j)} V_d(f_d, f_e(\mathbf{x}(t); \theta(t)^{(j)}))dt, \tag{9}$$

However, the upper limit of integration in above loss function approaching infinity results in an improper integral which may not converge. To overcome above challenges, we propose the following theorem which provides an analytical upper bound for the temporal-domain-invariant loss.

**Theorem 2.** *Suppose we are given an arbitrary function, $\pi(t)$ with $t \in [0, +\infty)$. Let $\beta = -e^{-t^2/\alpha^2}$. Let $\epsilon$ be a positive real constant and let $h : [-1, 0) \to R$ be a continuous function. There exists an $H : [-1, 0) \to [0, +\infty)$ that satisfies an initial value problem:*

$$H'(\beta) = h(\beta), \;\; H(-1) = 0, \; where \; h(\beta) = \frac{-\alpha}{2\beta\sqrt{-\log(-\beta)}}\pi(\alpha\sqrt{-\log(-\beta)})$$

*Such that as $\epsilon \to 0$, we have:*

$$\int_0^{+\infty} \pi(t)dt \leqslant \lim_{\epsilon \to 0}(H(-\epsilon) + \|H(-2\epsilon) - H(-\epsilon)\|)$$

Let $\pi(t)$ be the integrand of the right-hand-side of the Eq. (9), Theorem 2 (proof given in the Supplement) states that we can construct a $H : [-1, 0) \to [0, +\infty)$ that satisfies an initial value problem (IVP):

$$H'(\beta) = h(\beta), \;\; H(-1) = 0$$

where $h(\beta)$ can be written as:

$$h(\beta) = \frac{-\alpha}{2\beta\sqrt{-\log(-\beta)}} \sum\nolimits_{j=1}^{k} [w(\phi)^{(j)} V_d(f_d, f_e(\widetilde{\mathbf{x}}(\phi); \theta(\phi)^{(j)}))], \tag{10}$$

where $\phi = \sqrt{-\log(-\beta)}$; $\widetilde{\mathbf{x}}(\phi)$ is the approximated data. Note that we use $\widetilde{\mathbf{z}}$ to denote the encoder output for the approximated data. Then by Theorem 2, the upper bound of the temporal-domain-invariant loss, $\widetilde{l}_i$, can be written as:

$$l_i \leqslant \widetilde{l}_i = \lim_{\epsilon \to 0}(H(-\epsilon) + \|H(-2\epsilon) - H(-\epsilon)\|) \tag{11}$$

where $H(-\epsilon)$ and $H(-2\epsilon)$ can be obtained by solving the IVP. For instance, these two terms can be approximated using the ODE numerical solvers such as the Euler method and the Runge-Kutta methods [25]. In this work, we adopt Euler method to calculate these two terms, in which $\epsilon$ is set as the step size of the Euler method. Notably, by setting a suitable step size, $\widetilde{\mathbf{x}}(\phi)$, where $\phi \in (0, N)$ are required to calculate the bound; and thus $\widetilde{\mathbf{x}}(\phi)$ can be approximated by simply using linear

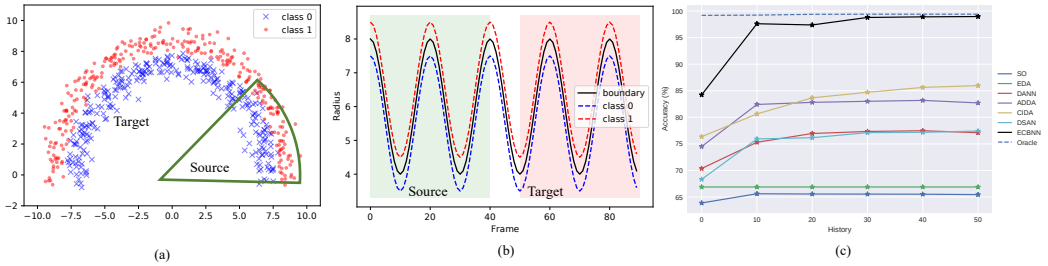


Figure 2: (a) Initial data samples at 0-th frame, where red dots and blue crosses are data samples from class 0 and class 1 respectively; Green sector indicates data samples from source domain. (b) Visualization of motion dynamics of decision boundary and data, where Black solid line represents the decision boundary, red hash line represents data for class 0 and blue hash line for class 1. (c) The accuracy of different methods by varying the length of historical stream data on Growing Circle

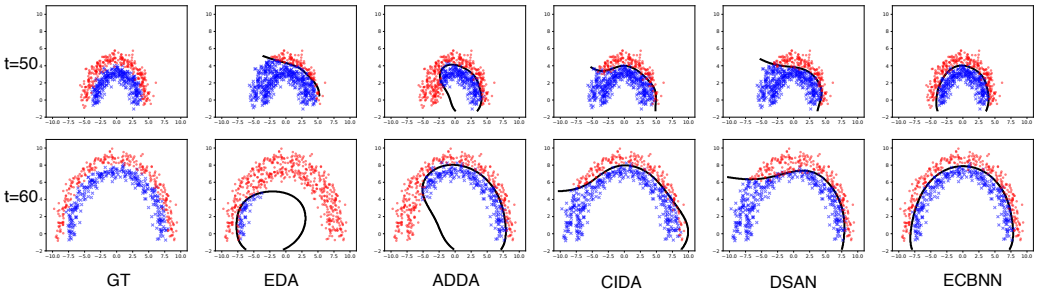


Figure 3: Results on the Growing Circle testing set. Black lines represent the decision boundaries generated according to model predictions.

interpolation over the hidden representation of the current batch. In summary, the final loss of our model can be written as a minimax optimization problem:

$$\min_{f_\theta, f_{\theta_0}, f_p} \max_{f_d} \Big( -l_b(\cdot; f_\theta, f_{\theta_0}, f_p, f_d) + \lambda_{de} l_{de}(\cdot; f_\theta) - \lambda_i \widetilde{l}_i(\cdot; f_d, f_\theta) \Big), \tag{12}$$

where $\lambda_{de}$ and $\lambda_i$ are importance weights; $f_\theta = \{f_{\theta_m}, f_{\theta_s}, f_{\theta_w}\}$, $f_{\theta_0} = \{f_{\theta_{m0}}, f_{\theta_{s0}}\}$.

# 4 Experiments

We evaluate ECBNN on both synthetic and real-world streaming data. Specifically, we conduct preliminary experiments on low dimensional synthetic toy data (Growing Circle), and then study the generalization of ECBNN in handling high dimensional synthetic video dataset based on handwritten digits (Streaming Rotating MNIST→USPS). We finally evaluate our model on OuluVS2, a realistic multi-view lip reading dataset. (See the Supplement for the detailed data configuration of datasets)

## 4.1 Baselines

We compare ECBNN with state-of-the-art UDA methods including **DANN** [15], **ADDA** [16], **CIDA** [18], and **DSAN** [17]. Since these UDAs assume static domains, we extend them to handle streaming data in our experiments through adopting recurrent neural network (RNN) in the encoder. For example, we extend CIDA by using both the frame index and categorical domain label as the input to the encoder and as target of the discriminator. Notice that all above methods can only perform offline adaptation, while ECBNN is able to perform online adaptation by continuously producing encoder distributions. We also compare our model with a typical online UDA method, (**EDA**) [11], which is able to perform online adaptation but requires collecting data batches for fine-tuning. For fair comparisons, all models have similar numbers of weight parameters, with the similar neural network architecture of encoder $f_c$, predictor $f_p$, and discriminator $f_d$. Following [18], $\lambda_d$ is chosen from $\{0.2, 0.5, 1.0, 2.0, 5.0\}$ (see implementations details in the Supplement).

### 4.2 Toy Dataset

**Growing Circle.** We conduct a preliminary study on a toy dataset, Growing Circle, for a binary classification task. Fig. 2 shows the initial data samples at the 0-th frame, in which a green sector separates the data on a circle into source and target domain. Fig. 2 (b) shows the motion dynamics of decision boundary and data, in which time ranges of source and target domain are indicated by different colors. We include 0-th to 39-th frame to source domain and 50-th to 89-th to target domain.

**Results and Analysis.** We visualize the predicted classification boundaries generated by different methods at frames $t = 50$ and $t = 60$, shown in Fig. 3 (more results in the the Supplement). We can see that ECBNN almost captures the ground truth (GT) decision boundaries at different time stamps, while other methods perform poorly when handling data with time-varying distributions. To evaluate ECBNN's capability of continuously generating better distribution of model parameters from historical stream data in real-time, we test our model by varying the length of the historical stream data ranging from 0 to 50 frames on the testing set. Fig. 2(c) shows the results for ECBNN and baselines. ECBNN consistently outperforms baselines. Interestingly, ECBNN's performance grows increasingly better with the accumulation of observed history, demonstrating ECBNN's potential in continuously learning from streaming data during real-time testing.

Table 1: **Accuracy (%) for various DA methods on Streaming Rotating MNIST→USPS Dataset**. We report the accuracy at the source domain, target domain and out-of-domain.

| Method | MNIST | USPS | |
|---|---|---|---|
| | Source | Target | OOD |
| Source-Only | 97.8 | 27.7 | 27.7 |
| DANN [15] | 97.7 | 45.9 | 33.0 |
| ADDA [16] | 97.4 | 45.8 | 32.2 |
| CIDA [18] | 97.3 | 37.1 | 31.5 |
| DSAN [17] | 96.1 | 46.8 | 33.0 |
| EDA [11] | **97.9** | 45.5 | 31.9 |
| ECBNN (Ours) | 97.1 | **60.9** | **38.5** |

### 4.3 Streaming Rotating MNIST→USPS

**Streaming Rotating MNIST→USPS.** We further evaluate ECBNN on high-dimensional streaming data. To do this, we construct a synthetic video dataset based on MNIST and USPS: Streaming Rotating MNIST→USPS. The video sequences are generated from the rotating MNIST and USPS handwritten digits with sinusoidal angular velocity. We take the first 40 frames of the Streaming Rotationg MNIST video sequence as source data, whose rotating angles range from $0°$ to $80°$; we take the 50-th to 90-th frames of the Streaming Rotationg USPS video sequence as target data, whose rotating angles range from $110°$ to $190°$; we also take the 90-th to 110-th frames of the Streaming Rotationg USPS video sequence to construct an additional Out-of-Domain (OOD) testing set.

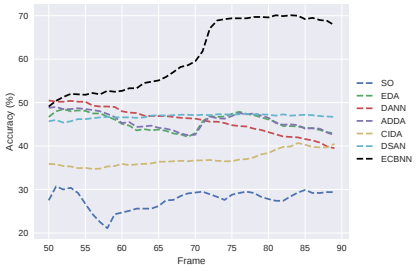


Figure 4: The frame-wise accuracy of different methods on Streaming Rotating USPS testing set.

**Results and Analysis.** We report the frame-level classification accuracy on the source-testing set, the target-testing set and the OOD-testing set, from the source domain, the target domain and out-of-domain respectively, as shown in Table 1. We can see that Source-Only achieves accuracy of 27.7%, which indicates this task is very challenging. ECBNN achieves the best accuracy of 60.9%. It dramatically outperforms other methods, e.g. it outperforms the state-of-the-art domain adaptation method (DSAN) by 14.1% absolute. For domain generalization, ECBNN outperforms the best-performing baselines by 5.5% absolute, demonstrating ECBNN's potential of temporal-domain-invariant representation learning in the entire timeline.

We further evaluate ECBNN's ability of continuously generating better distribution of model parameters from high-dimensional stream data in the target domain. We evaluate ECBNN and baselines on Rotating USPS testing set and report the accuracy at each frame in Fig. 4. Remarkably, while handling the real-time stream, ECBNN achieves gradually better performance over time. This demonstrates that ECBNN is capable of continuously generating better encoder distribution over time based on historical data, whereby it becomes better and better at handling the data from target domain.

### 4.4 Multi-veiw Lip Reading

**OuluVS2.** OuluVS2 dataset was originally proposed by [32], which is, as far as we know, the only publicly available dataset that contains multi-view lip motion data. It includes lipreading data recorded by 52 speakers from 5 different views. The train, valid and test sets include 5250, 750 and 1800 sequences, respectively. We adopt experiment settings similar to [33], e.g. the sequence-level class label is voted by class labels of all frames. To formulate the streaming domain adaptation scenes, we consider the sequences from $0^\circ$ view as the source domain and sequences from $[30^\circ, 45^\circ, 60^\circ, 90^\circ]$ as target domains. Besides, each data batch contains 10 frames, and each frame is a greyscale image of size $44 \times 50$.

Table 2: **Run time (s) and Accuracy (%) for various DA methods on OuluVS2 Dataset** We report run time per batch necessary to generate NN parameters and the sequence-level accuracy at the target domain.

| Method | Run time | Target |
|---|---|---|
| Source-Only | - | 46.0 |
| DANN [15] | - | 71.5 |
| ADDA [16] | - | 69.3 |
| CIDA [18] | - | 56.9 |
| DSAN [17] | - | 61.8 |
| EDA [11] | 0.23 | 70.3 |
| ECBNN (Ours) | **0.01** | **75.3** |

**Quantitative evaluation of ECBNN on OuluVS2.** Table 2 shows the sequence-level classification accuracy of the baseline models and our new ECBNN on Test of OuluVS2. We can see that ECBNN performs the best among all models in terms of accuracy in both source and target domains. ECBNN outperforms EDA, the online adaptation baseline, by $5.0\%$ absolute, while having remarkably lower latency in terms of generating model parameters. We also report the detailed accuracy of various DA methods on different views in the Supplement.

**Qualitative evaluation of ECBNN on temporal-domain-invariant feature learning.** The production of lip motion data is governed by highly nonlinear dynamics over time, e.g. the movements of various articulators. Therefore, aligning such data in terms of highly nonlinear temporal domain dynamics is very challenging. Here, we aim to qualitatively evaluate whether ECBNN can produce encoder distributions that achieves temporal-domain-invariant feature learning for real data. We randomly select data from 4 classes for both the source domain ($0^\circ$) and the target domain ($30^\circ$), and use t-SNE to visualize the corresponding encoder output $\mathbf{z}$ at time $t = 9, 15$, and $21$. Fig. 5 shows the visualization. Surprisingly, the alignment quality increases over time and features at $t = 21$ align much better than features at $t = 9$. This suggests that ECBNN can gradually improve the alignment given sequentially arriving data streams, thereby gradually improving accuracy on the target streaming data over time.

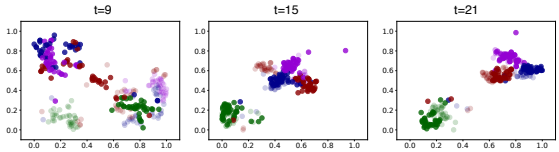


Figure 5: t-SNE visualizations of features at source domain (in *light* colors) and target domain (in *dark* colors) sampled at time $t = 9, 15$, and $21$. We use different colors to mark different classes.

## 5 Conclusion

Inspired by internal predictive modeling in neuroscience, this paper builds on the connection between Bayesian neural networks and continuous-time stochastic dynamical system. We derive a novel particle-filter-based differential equation, thereby proposing extrapolative continuous-time Bayesian neural network (ECBNN), which generalizes existing Bayesian neural networks to represent temporal dynamics. We then augment the unsupervised domain adaptation framework by adopting ECBNN as its encoder for handling non-stationary streaming data. We further provide an upper bound with theoretical guarantee for ECBNN to achieve temporal-domain-invariant representation learning. We show that ECBNN gradually improves model performance over time with low latency during the real-time testing stage. Our current approach assumes continuous dynamics of NN parameters; we leave it for future work to be able to learn more complex discrete dynamics introduced by discrete events.

# References


[1] Michael SA Graziano and Sabine Kastner. Human consciousness and its relationship to social neuroscience: a novel hypothesis. *Cognitive neuroscience*, 2(2):98–113, 2011.

[2] Artur Luczak and Yoshimasa Kubo. Predictive Neuronal Adaptation as a Basis for Consciousness. *Frontiers in Systems Neuroscience*, 15, 2022.

[3] Karl Friston. The free-energy principle: a unified brain theory? *Nature reviews neuroscience*, 11(2):127–138, 2010.

[4] Jill X O'reilly. Making predictions in a changing world—inference, uncertainty, and learning. *Frontiers in neuroscience*, 7:105, 2013.

[5] Jill X O'Reilly, Urs Schüffelgen, Steven F Cuell, Timothy EJ Behrens, Rogier B Mars, and Matthew FS Rushworth. Dissociable effects of surprise and model update in parietal and anterior cingulate cortex. *Proceedings of the National Academy of Sciences*, 110(38):E3660–E3669, 2013.

[6] Ezgi Kayhan, Marlene Meyer, Jill X O'Reilly, Sabine Hunnius, and Harold Bekkering. Nine-month-old infants update their predictive models of a changing environment. *Developmental cognitive neuroscience*, 38:100680, 2019.

[7] Hinze Hogendoorn. Perception in real-time: predicting the present, reconstructing the past. *Trends in Cognitive Sciences*, 26(2):128–141, 2022.

[8] Prem C Pandey, Hans Kunov, and Sharon M Abel. Disruptive effects of auditory signal delay on speech perception with lipreading. *Journal of Auditory Research*, 1986.

[9] Judy Hoffman, Trevor Darrell, and Kate Saenko. Continuous manifold based adaptation for evolving visual domains. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 867–874, 2014.

[10] Massimiliano Mancini, Samuel Rota Bulo, Barbara Caputo, and Elisa Ricci. Adagraph: Unifying predictive and continuous domain adaptation through graphs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6568–6577, 2019.

[11] Hong Liu, Mingsheng Long, Jianmin Wang, and Yu Wang. Learning to adapt to evolving domains. In *NeurIPS*, 2020.

[12] Uri Nodelman, Christian R Shelton, and Daphne Koller. Continuous time bayesian networks. *arXiv preprint arXiv:1301.0591*, 2012.

[13] Winnie Xu, Ricky TQ Chen, Xuechen Li, and David Duvenaud. Infinitely deep bayesian neural networks with stochastic differential equations. In *International Conference on Artificial Intelligence and Statistics*, pages 721–738. PMLR, 2022.

[14] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael Jordan. Learning transferable features with deep adaptation networks. In *International conference on machine learning*, pages 97–105. PMLR, 2015.

[15] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, Fran**c**cois Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *The journal of machine learning research*, 17(1):2096–2030, 2016.

[16] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7167–7176, 2017.

[17] Petar Stojanov, Zijian Li, Mingming Gong, Ruichu Cai, Jaime Carbonell, and Kun Zhang. Domain adaptation with invariant representation learning: What transformations to learn? *Advances in Neural Information Processing Systems*, 34, 2021.

[18] Hao Wang, Hao He, and Dina Katabi. Continuously indexed domain adaptation. *arXiv preprint arXiv:2007.01807*, 2020.

[19] Min-Hung Chen, Zsolt Kira, Ghassan AlRegib, Jaekwon Yoo, Ruxin Chen, and Jian Zheng. Temporal attentive alignment for large-scale video domain adaptation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6321–6330, 2019.

[20] Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno A Olshausen, and Trevor Darrell. Fully test-time adaptation by entropy minimization. 2020.

[21] Viraj Prabhu, Shivam Khare, Deeksha Kartik, and Judy Hoffman. Sentry: Selective entropy optimization via committee consistency for unsupervised domain adaptation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8558–8567, 2021.

[22] Aurick Zhou and Sergey Levine. Training on test data with bayesian adaptation for covariate shift. *arXiv preprint arXiv:2109.12746*, 2021.

[23] Yongxin Yang and Timothy M Hospedales. Multivariate regression on the grassmannian for predicting novel domains. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5071–5080, 2016.

[24] Jouko Lampinen and Aki Vehtari. Bayesian approach for neural networks—review and case studies. *Neural networks*, 14(3):257–274, 2001.

[25] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

[26] Cagatay Yildiz, Markus Heinonen, and Harri Lahdesmaki. Ode2vae: Deep generative second order odes with bayesian neural networks. *Advances in Neural Information Processing Systems*, 32, 2019.

[27] Raj Dandekar, Karen Chung, Vaibhav Dixit, Mohamed Tarek, Aslan Garcia-Valadez, Krishna Vishal Vemula, and Chris Rackauckas. Bayesian neural ordinary differential equations. *arXiv preprint arXiv:2012.07244*, 2020.

[28] Neil J Gordon, David J Salmond, and Adrian FM Smith. Novel approach to nonlinear/non-gaussian bayesian state estimation. In *IEE Proceedings F-radar and signal processing*, volume 140, pages 107–113. IET, 1993.

[29] Anshul Nasery, Soumyadeep Thakur, Vihari Piratla, Abir De, and Sunita Sarawagi. Training for the future: A simple gradient interpolation loss to generalize along time. *Advances in Neural Information Processing Systems*, 34, 2021.

[30] Hengguan Huang, Hongfu Liu, Hao Wang, Chang Xiao, and Ye Wang. Strode: Stochastic boundary ordinary differential equation. In *International Conference on Machine Learning*, pages 4435–4445. PMLR, 2021.

[31] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[32] Iryna Anina, Ziheng Zhou, Guoying Zhao, and Matti Pietikäinen. Ouluvs2: A multi-view audiovisual database for non-rigid mouth motion analysis. In *2015 11th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG)*, volume 1, pages 1–5. IEEE, 2015.

[33] Stavros Petridis, Yujiang Wang, Zuwei Li, and Maja Pantic. End-to-end multi-view lipreading. In *British Machine Vision Conference, BMVC 2017*, 2017.

## Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] The Supplement contains detailed instructions to reproduce the main experimental results and the code will be released when the paper is accepted.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] Please see more details in the Supplement.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] Please see more details in the Supplement.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] Please see more details in the Supplement.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]