Attentive Neural Controlled Differential Equations for Time-series Classification and Forecasting

Sheo Yon Jhin¹, Heejoo Shin¹, Seoyoung Hong¹, Minju Jo¹, Solhee Park¹, Noseong Park¹, Seungbeom Lee², Hwiyoung Maeng², Seungmin Jeon²
Yonsei University¹, Seoul, South Korea
Socar Co. Ltd.², Seoul, South Korea
{sheoyonj,xenos961,seoyoung,allfslowl12,leadwish11,noseong}@yonsei.ac.kr, {marcus,colde,Simon}@socar.kr

Abstract—Neural networks inspired by differential equations have proliferated for the past several years, of which neural ordinary differential equations (NODEs) and neural controlled differential equations (NCDEs) are two representative examples. In theory, NCDEs exhibit better representation learning capability for time-series data than NODEs. In particular, it is known that NCDEs are suitable for processing irregular time-series data. Whereas NODEs have been successfully extended to adopt attention, methods to integrate attention into NCDEs have not yet been studied. To this end, we present Attentive Neural Controlled Differential Equations (ANCDEs) for time-series classification and forecasting, where dual NCDEs are used: one for generating attention values, and the other for evolving hidden vectors for a downstream machine learning task. We conduct experiments on three real-world time-series datasets and ten baselines. After dropping some values, we also conduct experiments on irregular time-series. Our method consistently shows the best accuracy in all cases by non-trivial margins. Our visualizations also show that the presented attention mechanism works as intended by focusing on crucial information.

Index Terms—time-series data, neural controlled differential equations, attention

I. INTRODUCTION

Time-series data is a popular and important data type in the field of data mining and machine learning [1]–[10]. From an ordered set of multi-dimensional values, which are tagged with their occurrence time-point information, we typically extract useful information, forecast future values, or classify class labels. Many different techniques, ranging from classical statistical methods to recent deep learning-based methods, have been developed to tackle this type of data.

Among those deep learning-based methods, recurrent neural networks (RNNs) are considered the standard way to deal with time-series data. Long short-term memory (LSTM) networks and gated recurrent unit (GRU) networks are representative RNNs that are known to be suitable for processing time-series data [11]–[13].

Owing to the recent advancement of deep learning technology, however, neural ordinary differential equations (NODEs [14]) and neural controlled differential equations (NCDEs [15]) show state-of-the-art accuracy in many time-series tasks and datasets.

In NODEs, a multi-variate time-series vector \( z(t_1) \) at time \( t_1 \) can be either a raw multi-variate time-series vector or a hidden vector created from the raw input.

In other words, NODEs have continuous depth.

In NCDEs, the time variable \( t \) corresponds to the layer of a neural network. Whereas NODEs, NCDEs assume more complicated evolutionary processes and use \( z(t) = z(t_0) + \int_{t_0}^{t} f(z(t); \theta_f) dX(t) \), where the integral is a Riemann–Stieltjes integral [15]. NCDEs reduce to NODEs if and only if \( X(t) = t \), i.e., the identity function w.r.t. time \( t \). In other words, NODEs are controlled by time \( t \) whereas NCDEs are controlled by another time-series \( X(t) \). In this regard, NCDEs can be understood as a generalization of NODEs.

Fig. 1. The overall architecture of NCDE and our proposed attentive NCDE (ANCDE). In our method, the bottom NCDE produces attention values and the top NCDE produces the last hidden vector. This figure assumes classification but we also support regression.
Another point worth mentioning is that \( f(z(t); \theta_f) dX(t) \) indicates a matrix-vector multiplication, which can be performed without spending large computational costs. In [19], it has been demonstrated that NCDEs significantly outperform RNN-type models and NODEs in standard benchmark problems.

Despite their promising performance, NCDEs are still a fairly novel concept in their early phase of development. In this work, we greatly enhance the accuracy of NCDEs by adopting attention. Attention is a widely-used approach to enhance neural networks by letting them focus on a small set of useful information only, which is also effective in time-series modeling [20]. To the best of our knowledge, no other works have ventured to apply attention to NCDEs.

In our proposed Attentive Neural Controlled Differential Equations (ANCDEs), there are two NCDEs as shown in Fig. 1 for generating attention values and the other for evolving \( z(t) \). We support two different attention methods: i) time-wise attention \( a(t) \in [0, 1] \) and ii) element-wise attention \( a(t) \in [0, 1]^D \), where \( D = \text{dim}(X(t)) \). In both cases, we feed the (element-wise) multiplication of the attention and \( X(t) \), denoted by \( Y(t) \), into the second NCDE, therefore the second NCDE’s input is comprised of values selected from \( X(t) \) by the first NCDE. We also propose a training algorithm that guarantees the well-posedness of our training problem.

We conduct experiments on three real-world datasets and tasks: i) hand-written character recognition, ii) mortality classification, and iii) stock price forecasting. We compare our method with RNN-based, NODE-based, and NCDE-based models. In particular, we consider the state-of-the-art NODE model with attention. Our proposed method outperforms them in all cases. Our contributions can be summarized as follows:

1) We extend NCDEs by adopting attention.
2) To this end, we use dual NCDEs. The bottom NCDE produces the attention value of the path \( X(t) \). The top NCDE reads the (element-wise) multiplication of the attention and \( X(t) \) and produces the last hidden vector for a downstream machine learning task.
3) We propose a specialized training algorithm that guarantees the well-posedness of our training problem.
4) Our method, called ANCDE, significantly outperforms existing baselines.

II. RELATED WORK & PRELIMINARIES

In this section, we review NODEs and NCDEs in detail. We also introduce attention.

A. Neural ODEs

NODEs solve the following initial value problem (IVP), which involves an integral problem, to calculate \( z(t_1) \) from \( z(t_0) \) [14].

\[
z(t_1) = z(t_0) + \int_{t_0}^{t_1} f(z(t); \theta_f) dt,
\]

where \( f(z(t); \theta_f) \), which we call ODE function, is a neural network to approximate \( \dot{z} \overset{\text{def}}{=} \frac{dz(t)}{dt} \). To solve the integral problem, NODEs rely on ODE solvers, such as the explicit Euler method, the Dormand–Prince (DOPRI) method, and so forth [21]. Fig. 2 shows the typical architecture of NODEs. There is a feature extraction layer which produces the initial value \( z(0) \) and the NODE layer evolves it to \( z(1) \). This architecture is much simpler than that of NCDEs in Fig. 1.

In general, ODE solvers discretize time variable \( t \) and convert an integral into many steps of additions. For instance, the explicit Euler method can be written as follows in a step:

\[
z(t + h) = z(t) + h \cdot f(z(t); \theta_f),
\]

where \( h \), which is usually smaller than 1, is a pre-determined step size of the Euler method. The DOPRI method uses a much more sophisticated method to update \( z(t + h) \) from \( z(t) \) and dynamically controls the step size \( h \). However, those ODE solvers sometimes incur unexpected numerical instability [22]. For instance, the DOPRI method sometimes keeps reducing the step size \( h \) and eventually, an underflow error is produced. To prevent such unexpected problems, several countermeasures were also proposed.

Instead of the backpropagation method, the adjoint sensitivity method is used to train NODEs for its efficiency and theoretical correctness [14]. After letting \( a_z(t) = \frac{dL}{dz(t)} \) for a task-specific loss \( L \), it calculates the gradient of loss w.r.t model parameters with another reverse-mode integral as follows:

\[
\nabla_{\theta_f} L = \frac{dL}{d\theta_f} = -\int_{t_m}^{t_0} a_z(t)^T \frac{\partial f(z(t); \theta_f)}{\partial \theta_f} dt.
\]

\( \nabla z(0) L \) can also be calculated in a similar way and we can propagate the gradient backward to layers earlier than the ODE if any. It is worth of mentioning that the space complexity of the adjoint sensitivity method is \( O(1) \) whereas using the backpropagation to train NODEs has a space complexity proportional to the number of DOPRI steps. Their time complexities are similar or the adjoint sensitivity method is slightly more efficient than that of the backpropagation. Therefore, we can train NODEs efficiently.

B. Neural CDEs

One limitation of NODEs is that \( z(t_1) \) is, given \( \theta_f \), decided by \( z(t_0) \), which degrades the representation learning capability of NODEs. To this end, NCDEs create another path \( X(t) \) from underlying time-series data and \( z(t_1) \) is decided by both \( z(t_0) \) and the NODE layer evolves it to \( z(1) \). This architecture is much simpler than that of NCDEs in Fig. 1.
and \( X(t) \) \[^{[15]}\]. The initial value problem (IVP) of NCDEs is written as follows:

\[
\begin{align*}
    z(t_1) &= z(t_0) + \int_{t_0}^{t_1} f(z(t); \theta_f) dX(t), \\
    z(t_0) &= z(t_0) + \int_{t_0}^{t_1} f(z(t); \theta_f) \frac{dX(t)}{dt} dt,
\end{align*}
\]

where \( X(t) \) is a natural cubic spline path of underlying time-series data. In particular, the integral problem is called the Riemann–Stieltjes integral (whereas NODEs use the Riemann integral). Differently from NODEs, \( f(z(t); \theta_f) \), which we call the CDE function, is to approximate \( \frac{dz(t)}{dt} \). Whereas other methods can be used for \( X(t) \), the original authors of NCDEs recommend the natural cubic spline method for its suitable characteristics to be used in NCDEs: i) it is twice differentiable, ii) its computational cost is not much, and iii) \( X(t) \) becomes continuous w.r.t. \( t \) after the interpolation.

C. Attention

Attention, which is to extract key information from data, is one of the most influential concepts in deep learning. It has been widely studied for computer vision and natural language processing \[^{[20]–[23]}\]. After that, it was quickly spread to other fields \[^{[26]–[28]}\].

The attention concept can be well explained by the visual perception system of human, which focuses on strategic parts of input while ignoring irrelevant signals \[^{[29]}\]. Several different types of attention and their applications have been proposed. A self-attention method was proposed for language understanding \[^{[30]}\]. Kiela et al. proposed a general purpose text representation learning method with attention in \[^{[31]}\]. For machine translation, a soft attention mechanism was proposed by Bahdanau et al. \[^{[22]}\]. A co-attention method was used for visual question answering \[^{[32]}\]. We encourage referring to survey papers \[^{[33], [34]}\] for more detailed descriptions. Like this, there exist many different types and applications of attention.

To our knowledge, attention has not yet been actively studied for NCDEs. This is likely due to the fact that they are a relatively new paradigm of designing neural networks, and that it is not straightforward how to integrate attention into them. On the other hand, several attention-related designs exist for NODEs. ETN-ODE \[^{[35]}\] uses attention in its feature extraction layer before the NODE layer. It does not propose any new NODE model that is internally combined with attention. In other words, they use attention to derive \( z(0) \) in their feature extraction layer, and then use a standard NODE layer to evolve \( z(0) \). As the concept of attention is only applied to the feature extraction layer, it is difficult to classify ETN-ODE as an attention-based NODE model.

As another example, ACE-NODE proposed to augment \( z(t) \) with another ODE producing attention values. In NODEs, \( z(t) \) means a hidden vector at time \( t \). To help a downstream task with attention, \( z(t+s) \) should be derived from \( z(t) \) aided by the attention at time \( t \). To this end, its ODE state is extended to \( [z(t), z'(t)] \), where \( z(t) \) is the main ODE and \( z'(t) \) is the attention ODE. They evolve both ODEs simultaneously and, while updating \( z(t) \), it refers to the attention value. They naturally co-evolve rather than evolving independently. As of now, ACE-NODE is the work most similar to our ANCDE. However, the two works differ in that ACE-NODE proposes a novel NODE architecture, while our work extend NCDEs with attention.

III. PROPOSED METHOD

We propose the method of Attentive Neural Controlled Differential Equation (ANCDE), where we adopt two NCDEs: the bottom NCDE for calculating attention values and the top NCDE for time-series classification/forecasting.

A. Overall Workflow

Fig. 3 shows the detailed architecture of our proposed ANCDE. Given a time-series data \( \{(x_0, t_0), (x_1, t_1), \ldots \} \) annotated with observation time, we first construct a continuous path \( X(t) \) with the natural cubic spline algorithm. We then feed \( X(t) \) into the bottom NCDE which produces attention values for each time-point \( t \). Another path \( Y(t) \) is created by the (element-wise) multiplication of the attention and \( X(t) \) in Eq. (15) and fed into the top NCDE which produces the last hidden vector. There is one more classification layer in Eq. (17) which produces the final prediction.

Even though each raw data point \( (x_i, t_i) \) is observed in a discrete and irregular fashion, the path \( X(t) \) is continuous and we note that \( X(t_i) = x_i \), where \( t_i \) is the time-point when \( x_i \) is observed. For other non-observed time-points, the natural cubic spline algorithm interpolates nearby observed data.

NCDE vs. NODE: The key difference between NCDEs and NODEs lies in the existence of the path \( X(t) \). When calculating \( X(t) \) in NCDEs, in particular, the natural cubic spline considers some selected future observations \( \{x_{t'}\} \), where \( t' \geq t \), in conjunction with its current and past observations, which is not the case in NODEs. Therefore, NCDEs can be considered as a more general method than NODEs \[^{[19]}\]. In particular, NCDEs reduce to NODEs when \( X(t) = t \), i.e., the identity function w.r.t. time.

B. Attentive NCDE

1) Bottom NCDE for attention values: The bottom NCDE can be written as follows:

\[
\begin{align*}
    h(t_1) &= h(t_0) + \int_{t_0}^{t_1} f(h(t); \theta_f) dX(t), \\
    h(t_0) &= h(t_0) + \int_{t_0}^{t_1} f(h(t); \theta_f) \frac{dX(t)}{dt} dt,
\end{align*}
\]

where \( h(t) \) means a hidden vector of attention. We later derive attention at time \( t \) from \( h(t) \). We note that this bottom NCDE is equivalent to the original NCDE setting.

We support two different attention concepts: i) time-wise attention \( a(t) \in \mathbb{R} \) and ii) element-wise attention \( a(t) \in \mathbb{R}^{\text{dim}(X(t))} \). In the former type, the output size of the fully-connected layer \( FC_1 \) is 1 and \( a(t) = \sigma(FC_1(h(t))) \) is a scalar value. In the latter type, \( a(t) = \sigma(h(t)) \) is a vector.
where the temperature $\tau$ with a large enough to control the slope of the sigmoid function. We note that, of the rounding function. Therefore, we keep increasing different definitions of $\sigma$ after initializing it to 1 at the beginning.

With the straight-through estimator, i.e., the hard attention with the original sigmoid followed by the rounding function, iii) attention with the original sigmoid, ii) the hard attention with the soft attention. However, this does not theoretically break the straight-through estimator. The hard attention generates values in 0 or for the element-wise attention, whereas values in 1 are produced by the soft attention.

We also support three different types of $\sigma$: i) the soft attention with the original sigmoid, ii) the hard attention with the original sigmoid followed by the rounding function, iii) the straight-through estimator, i.e., the hard attention with the sigmoid slope annealing. We omit the description of the soft attention since we use the original sigmoid.

We further derive Eq. (12) into a more tractable form as follows:

$$z(t_1) = z(t_0) + \int_{t_0}^{t_1} g(z(t); \theta_g) \frac{dY(t)}{dt} dt,$$

where the temperature $\tau \geq 1.0$ is a scalar multiplicative term to control the slope of the sigmoid function. We note that, with a large enough $\tau$, the sigmoid slope becomes close to that of the rounding function. Therefore, we keep increasing $\tau$ slowly as training goes on. We add 0.12 to $\tau$ every epoch after initializing it to 1 at the beginning.

In total, therefore, we can define six attention models, i.e., six combinations from the two attention types and the three different definitions of $\sigma$, as shown in Table I.

### Table I

| Type          | Time/Element-wise | $\sigma$      |
|--------------|-------------------|---------------|
| SOFT-TIME    | Time-wise (i.e., $a(t) \in \mathbb{R}$) | Sigmoid       |
| HARD-TIME    | Time-wise (i.e., $a(t) \in \mathbb{R}$) | Hard         |
| STE-TIME     | Time-wise (i.e., $a(t) \in \mathbb{R}$) | Straight-through |
| SOFT-ELEM    | Element-wise (i.e., $a(t) \in \mathbb{R}^{dim(X(t))}$) | Sigmoid       |
| HARD-ELEM    | Element-wise (i.e., $a(t) \in \mathbb{R}^{dim(X(t))}$) | Hard         |
| STE-ELEM     | Element-wise (i.e., $a(t) \in \mathbb{R}^{dim(X(t))}$) | Straight-through |

2) Top NCDE for time-series classification/forecasting: The top NCDE can be written as follows:

$$z(t_1) = z(t_0) + \int_{t_0}^{t_1} g(z(t); \theta_g) \frac{dY(t)}{dt} dt,$$

where $\otimes$ stands for the element-wise multiplication, and $Y(t)$ is the (element-wise) multiplication of the attention and $X(t)$. Therefore, $Y(t)$ contains information selected by the bottom NCDE. In this regard, the top NCDE is able to focus on useful information only and the performance of the downstream machine learning task can be improved.

We further derive Eq. (12) into a more tractable form as follows:

$$z(t_1) = z(t_0) + \int_{t_0}^{t_1} g(z(t); \theta_g) \frac{dY(t)}{dt} dt,$$

where for the time-wise attention,

$$\frac{dY(t)}{dt} = \frac{dX(t)}{dt} a(t) + X(t) a(t)(1 - a(t)) \frac{dFC_1}{dh(t)} \frac{dh(t)}{dt},$$

or for the element-wise attention,

$$\frac{dY(t)}{dt} = \frac{dX(t)}{dt} a(t) + X(t) a(t)(1 - a(t)) \frac{dh(t)}{dt}.$$
Algorithm 1: How to train ANCDE

Input: Training data $D_{train}$, Validating data $D_{val}$, Maximum iteration number $max\_iter$

1. Initialize $\theta_f, \theta_g$, and other parameters $\theta_{\text{others}}$ if any, e.g., the parameters of the feature extractor and the classifier;
2. $k \leftarrow 0$;
3. while $k < max\_iter$ do
4. Train $\theta_{\text{others}}$ after fixing $\theta_f$ and $\theta_g$;
5. Train $\theta_f$ after fixing $\theta_{\text{others}}$ and $\theta_g$;
6. Train $\theta_g$ after fixing $\theta_f$ and $\theta_{\text{others}}$;
7. Validate and update the best parameters, $\theta_f^*, \theta_g^*$, and $\theta_{\text{others}}^*$, with $D_{val}$;
8. $k \leftarrow k + 1$;
9. return $\theta_f^*, \theta_g^*$, and $\theta_{\text{others}}^*$;

Therefore, the hard attention is still in the valid range. For instance, suppose the time-wise attention. When $a(t) = 0$ (resp. $a(t) = 1$) by the hard attention, $\frac{dy(t)}{dt} = 0$ (resp. $\frac{dy(t)}{dt} = dx(t)$), which exactly corresponds to our attention motivation, i.e., the top NCDE focuses on the values selected by the bottom NCDE. The straight-through estimator is considered as a variant of the hard attention with the temperature annealing and it also produces values in $\{0, 1\}$. Therefore, there are no problems in Eqs. (15), (16) to be used for all the three attention types.

C. Training Algorithm

We use the adjoint backpropagation method [14], [30–38] to train NCDEs, which requires a memory of $O(L+H)$ where $L = t_l - t_0$, i.e., the integral time space, and $H$ is the size of the vector field defined by an NCDE. Therefore, the training at Lines 4 to 6 of Alg. 1 is done with the adjoint method. However, our framework has two NCDEs and as a result, it needs a memory of $O(2L + H_f + H_g)$ where $H_f$ and $H_g$ are the sizes of the vector fields created by the bottom and the top NCDEs, respectively.

Well-posedness of the Problem: The well-posedness of NCDEs, given a fixed path, was already proved in [39] Theorem 1.3 under the mild condition of the Lipschitz continuity. We show that our ANCDEs are also well-posed problems. Almost all activations, such as ReLU, Leaky ReLU, SoftPlus, Tanh, Sigmoid, ArcTan, and Softsign, have a Lipschitz constant of 1. Other common neural network layers, such as dropout, batch normalization and other pooling methods, have explicit Lipschitz constant values. Therefore, the Lipschitz continuity of $f$ and $g$ can be fulfilled in our case. Therefore, it is a well-posed training problem for the bottom NCDE producing attention values given other fixed parts (Line 5). Given a fixed path $Y(t)$, then it becomes well-posed again for the top NCDE (Line 6). Therefore, our training algorithm solves a well-posed problem so its training process is stable in practice.

In our experiments, we solve time-series classification problems and use the standard cross-entropy loss with the final hidden vector $z(t_1)$, followed by a classification layer as follows:

$$\hat{y} = \phi(FC_2(z(t_1))),$$  \hspace{1cm} (17)

where $\hat{y}$ is a predicted class label and $\phi$ is softmax activation. The output size of $FC_2$ is the same as the number of classes and we use the standard cross-entropy loss. For regression, the output size of $FC_2$ is the same as the size of target prediction and we utilize mean squared error loss.

IV. EXPERIMENTS

In this section, we describe our experimental environments and results. We conduct experiments on image classification and time-series forecasting. All experiments were conducted in the following software and hardware environments: UBUNTU 18.04 LTS, PYTHON 3.6.6, NUMPY 1.18.5, SCIPY 1.5, MATHPLOTLIB 3.3.1, PYTORCH 1.2.0, CUDA 10.0, NVIDIA Driver 417.22, i9 CPU, and NVIDIA RTX TITAN. We repeat the training and testing procedures with five different random seeds and report their mean and standard deviation accuracy. Our source code and data can be found at https://github.com/sheoyon-jhin/ANCDE.

A. Datasets

Character Trajectories: The Character Trajectories dataset from the UEA time-series classification archive [40] contains 2,858 pen tip trajectory samples of Latin alphabets written on a tablet. They contain information about the X and Y-axes position and pen tip force with a sampling frequency of 200Hz. All data samples were truncated to a time-series length of 182, the shortest length in the dataset. Each sample is represented by a time-series of 3 dimensional vectors, i.e., X and Y-axes...
positions and pen tip force. The goal is to classify each sample into one of the 20 different class labels (‘a’, ‘b’, ‘c’, ‘d’, ‘e’, ‘g’, ‘h’, ‘i’, ‘m’, ‘n’, ‘o’, ‘p’, ‘q’, ‘r’, ‘s’, ‘u’, ‘v’, ‘w’, ‘y’, ‘z’). Note that some alphabet characters are missing in this dataset.

**PhysioNet Sepsis:** The PhysioNet 2019 challenge on sepsis prediction [41], [42] is also used for our time-series classification experiments. Sepsis is a life-threatening condition caused by bacteria or bacterial toxins in the blood. In the U.S., about 1.7 million people develop sepsis and 270,000 die from sepsis in a year. More than a third of patients who die in U.S. hospitals have sepsis. Early prediction of sepsis could potentially save lives, so this experiment is especially meaningful. The dataset consists of 40,335 cases of patients staying in intensive care units. There are 34 time-dependent variables, such as heart rate, oxygen saturation, body temperature, and so on. The last variable is the class label, which indicates the onset of sepsis according to the sepsis-3 definition, where “1” indicates sepsis and “0” means no sepsis, and “nan” means no information. The goal is to classify whether each patient has sepsis or not after removing the “nan” class.

**Google Stock:** The Google Stock dataset includes trading volumes in conjunction with the high, low, open, close, and adjusted closing prices from 2004 to 2019 [43]. This dataset is used for time-series forecasting experiments. CDEs were originally developed in the field of mathematical finance to predict various financial time-series values [44]. Therefore, we expect that our ANCDE shows the best appropriateness for this task. The goal is to predict, given past several days of time-series values, the high, low, open, close, and adjusted closing prices at the very next day.

### B. Experimental Methods

We conduct classification and forecasting (regression) experiments to verify the efficacy of the proposed attention-based method. Our experiments can be classified into the following two types: i) regular time-series experiments, and ii) irregular time-series experiments. For the regular time-series experiments, we use all the data provided in each dataset without dropping any observations. For the irregular time-series experiments, however, we randomly drop 30%, 50%, and 70% of observations from each individual time-series sample and then try to predict, e.g., we read the stock price observations in past random 70% days for Google and predict next prices.

The only exception is the PhysioNet Sepsis dataset, where 90% of the observations were randomly removed by the data collectors in order to protect the privacy of the people who had contributed to the dataset. Hence, we only perform irregular time-series classification with this data.

Different performance metrics are utilized for each task. For classification with balanced classes we use accuracy, whereas AUCROC is utilized in the case of imbalanced classification. For forecasting problems, we consider mean absolute error (MAE) or mean squared error (MSE).

### C. Baselines

For regular time-series forecasting and classification experiments, we compare our method with the following methods:

1) RNN, LSTM [11], GRU [12] are all recurrent neural network-based models which are able to handle sequential data. LSTM is designed to learn long-term dependencies and GRU uses gating mechanisms to control the flow of information.

2) GRU-ODE [16], [45] is a NODE analogous to GRU. This model is a continuous counterpart to GRU.

3) ODE-RNN [46] is an extension of NODE by adopting a GRU between observations. This combination of NODE and GRU is known as a jump [47].

4) Latent-ODE [14] is a time-series model in which the latent state follows a NODE. The work in [40] uses the ODE-RNN to replace the recognition network of the original Latent-ODE model. We also use the ODE-RNN as a recognition network.
5) **ACE-NODE** [48] is the state-of-the-art attention-based NODE model which has dual co-evolving NODEs.

For the irregular time-series forecasting and classification experiments, we compare our method with the following methods:

1) **GRU-∆t** [19] is the GRU with the time difference between observations additionally used as an input.
2) **GRU-D** [49] is a modified version of GRU-∆t with a learnable exponential decay between observations.
3) **GRU-ODE, ODE-RNN, Latent-ODE, and ACE-NODE** can also be utilized in irregular time-series experiments as they were developed with consideration of irregular time-series patterns.

### D. Hyperparameters

We test the following common hyperparameters for our method and other baselines:

1) In Character Trajectories, we train for 200 epochs with a batch size of 32, a learning rate of $\{1.0 \times 10^{-4}, 5.0 \times 10^{-4}, 1.0 \times 10^{-3}, 5.0 \times 10^{-3}\}$ and a hidden vector dimension of $\{10, 20, 40, 50, 60\}$ and set the ODE function to have $\{3, 4, 5\}$ layers;
2) In PhysioNet Sepsis, we train for 150 epochs with a batch size 1024, a learning rate of $\{1.0 \times 10^{-5}, 5.0 \times 10^{-5}, 1.0 \times 10^{-4}, 5.0 \times 10^{-4}\}$, a hidden vector dimension of $\{30, 40, 50, 60\}$, and set the ODE function to have $\{2, 3, 4, 5\}$ layers;
3) In Google Stock, we train for 150 epochs with a batch size 200, a learning rate of $\{1.0 \times 10^{-2}, 1.0 \times 10^{-3}, 1.0 \times 10^{-4}\}$, a hidden vector dimension of $\{30, 40, 50, 60\}$, and set the ODE function to have $\{2, 3, 4\}$ layers;

For reproducibility, we report the best hyperparameters for our method as follows:

1) In Character Trajectories, we set the learning rate to $1.0 \times 10^{-3}$;
2) In PhysioNet Sepsis, we set the learning rate to $1.0 \times 10^{-5}$;
3) In Google Stock, we set the learning rate to $1.0 \times 10^{-3}$.
4) In Tables [III] to [VII], we also clarify the network architecture of the CDE functions, $f$ and $g$, i.e., the main networks modeling $\frac{dx(t)}{dt}$ and $\frac{dx(t)}{dt}$, respectively.

### E. Experimental Results

#### Character Trajectories (Regular): We first introduce our experimental results with regular time-series classification on Character Trajectories. Table [VIII] summarizes the results. All RNN-based methods except GRU show unreliable accuracy. In particular, RNN shows unreasonable accuracy in comparison with others due to its low representation learning capability. However, GRU shows good accuracy. In general, NODE and NCDE-based methods show good accuracy. ACE-NODE and NCDE show almost the same mean accuracy. The attention mechanism of ACE-NODE significantly increases the accuracy in comparison with other NODE-based models. ANCDE greatly outperforms other methods by a wide margin, showing a near-perfect accuracy of 99.1%.

#### Character Trajectories (Irregular): We also conduct irregular time-series classification experiments on Character Trajectories. For this, we randomly drop 30/50/70% of values in a time-series sample and classify. Recall that in this dataset, each time-series sample includes a sequence of (X-position, Y-position, Pen tip force) and its class is an alphabet character. Therefore, our irregular time-series classification corresponds to classifying with partial observations on drawing patterns in a tablet.

Table [IX] summarizes the results. One interesting point is that for some models, irregular time-series classification accuracy is higher than that of regular time-series classification,
The PhysioNet dataset is basically an irregular time-series dataset. Instead, we consider two different irregular time-series decomposition methods: i) one with observation intensity (OI), and ii) the other without observation intensity (No OI). When we consider observation intensity, we add an index number for each time-series value. Patients are typically frequently checked when their conditions are bad. Therefore, the intensity, i.e., the number of observations, somehow implies the mortality class label.

Table X contains the experimental results. With OI, many methods show good AUCROC scores. All GRU-based irregular time-series classification models show scores of 0.852 to 0.878. However, NODE-based methods outperform them. In general, NCDE-based methods are better than NODE-based models. Our ANCD shows the best AUCROC score with a little memory overhead in comparison with NCDE.

**Google Stock (Irregular):** Table XII shows irregular time-series forecasting results. The overall tendency is similar to that of the regular time-series experiment. GRU-based methods show reliable performance and NCDE performs poorly. However, ANCD successfully stabilizes the performance again.

**Abstraction Studies**

We additionally conduct ablation studies w.r.t. the six attention types in Tables XIII to XVII. We omit the ablation studies for regular time-series data due to space constraints.

In Table XIII, the soft time-wise attention shows the highest accuracy for the irregular time-series classification. For this dataset, the soft time-wise attention is preferred.

For PhysioNet Sepsis, however, the soft element-wise attention shows the highest AUCROC score as shown in Table XIV. In this dataset, the performance gap between attention types is very large, e.g., an AUCROC of 0.498 by the hard time-wise attention vs. 0.900 by the soft element-wise attention.

For Google Stock, however, the element-wise attention with STE shows the smallest errors in general. The hard time-wise attention shows the worst-case error, which is two orders of magnitude larger than that of the element-wise attention with STE.
TABLE XV
ABLATION STUDY ON REGULAR CHARACTER TRAJECTORIES

| Model     | Test Accuracy |
|-----------|---------------|
| SOFT-ELEM | 0.981         |
| HARD-ELEM | 0.792         |
| HARD-TIME | 0.604         |
| STE-ELEM  | 0.952         |
| STE-TIME  | 0.623         |
| SOFT-TIME | 0.992         |

TABLE XVI
ABLATION STUDY ON REGULAR GOOGLE STOCK

| Model     | Test MSE |
|-----------|----------|
| SOFT-ELEM | 0.001    |
| HARD-ELEM | 0.001    |
| HARD-TIME | 0.001    |
| STE-ELEM  | 0.001    |

TABLE XVII
ABLATION STUDY ON IRRREGULAR GOOGLE STOCK

| Model     | Test MSE | 30% dropped | 50% dropped | 70% dropped |
|-----------|----------|-------------|-------------|-------------|
| SOFT-ELEM | 0.001    | 0.002       | 0.001       |
| SOFT-TIME | 0.001    | 0.002       | 0.002       |
| HARD-ELEM | 0.002    | 0.001       | 0.002       |
| HARD-TIME | 0.051    | 0.048       | 0.048       |
| STE-TIME  | 0.021    | 0.022       | 0.023       |
| STE-ELEM  | 0.001    | 0.001       | 0.001       |

G. Visualization of Attention and Training Process

Figs. 4 and 5 show how our proposed attention mechanism works. Values highlighted in blue indicate crucial information. In PhysioNet Sepsis, for instance, strong attention scores are given when there are non-trivial changes on the curves.

In Google Stock, the attention mechanism works in a more aggressive fashion. It entirely (or partially) ignores the open/low/close prices — note that these three price figures are mostly white. Since future prices can be readily reconstructed from other parts of the data, our proposed method ignores these data values. Similar attention patterns are consistently observed in this dataset.

V. CONCLUSIONS

Recently, differential equation-based methods have been widely used for time-series classification and forecasting. Ever since NODEs were first introduced, many advanced concepts building upon NODEs have been proposed. However, NCDEs are still a rather unexplored field. To this end, we propose ANCDE, a novel NCDE architecture that incorporates the concept of attention. ANCDE consists of two NCDEs, where the bottom NCDE provides the top NCDE with attention scores, and the top NCDE utilizes these scores to perform a downstream machine learning task on time-series data. Our experiments with three real-world time-series datasets and ten baseline methods successfully prove the efficacy of the proposed concept. In addition, our visualization of attention intuitively delivers how our proposed method works.

ACKNOWLEDGEMENT

Noseong Park is the corresponding author. This work was supported by the Yonsei University Research Fund of 2021, and the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korean government (MSIT) (No. 2020-0-01361, Artificial Intelligence Graduate School Program (Yonsei University), and No. 2021-0-00155, Context and Activity Analysis-based Solution for Safe Childcare).

REFERENCES

[1] T.-c. Fu, “A review on time series data mining,” Engineering Applications of Artificial Intelligence, vol. 24, no. 1, pp. 164–181, 2011.
