Room Occupancy Detection using Liquid Time-Constant Network

Aryamaan Rajesh Chaurasia
Department of Electronics and Communication Engineering
National Institute of Technolgy Karntaka
Surathkal, India
aryamaanrc.211ec106@nitk.edu.in

Ganta Yashwanth Kumar
Department of Electronics and Communication Engineering
National Institute of Technolgy Karntaka
Surathkal, India
gantayashwanthkumar.211ec112@nitk.edu.in

Puligadda Shashanka Mouli
Department of Electronics and Communication Engineering
National Institute of Technolgy Karntaka
Surathkal, India
puligaddashashankamouli.211ec138@nitk.edu.in

Vishal Marwade
Department of Electronics and Communication Engineering
National Institute of Technolgy Karntaka
Surathkal, India
vishalmarwade.211ec164@nitk.edu.in

***Abstract*— we present the implementation of networks comprising linear first-order dynamical systems, modulated through nonlinear interlinked gates. The resultant models portray dynamical systems characterized by variable (liquid) time-constants coupled to their hidden state. Outputs are computed utilizing numerical differential equation solvers. These neural networks demonstrate stable and bounded behavior, offering enhanced expressivity within the realm of neural ordinary differential equations. Moreover, they exhibit improved performance on time-series prediction tasks.**

Subsequently, we leverage these networks to develop a Room Occupancy Detection model, showcasing the approximation capabilities of Liquid Time-Constant Networks(LTCs) in contrast to classical and contemporary Recurrent Neural Network(RNN) such as Long-Short-Term Memory(LSTM) and Continuous-Time Recurrent Neural Network(CT-RNN).

Keywords*— *Liquid Time-Constant Networks(LTC), LSTM, CT-RNN, Gradient Descent, Time Series Prediction, Neural Networks

I. INTRODUCTION

The Liquid Time Constant (LTC) network is a reservoir computing model that leverages the dynamics of a recurrent reservoir to process temporal information in a manner inspired by biological neural networks, particularly the neocortex. The reservoir, composed of interconnected neurons, exhibits rich nonlinear dynamics crucial for capturing complex temporal patterns in input data. What sets LTC apart is its incorporation of time constants, which govern the rates of neuronal integration and transmission. These time constants play a pivotal role in shaping the network's response dynamics, influencing its ability to encode and process temporal dependencies effectively. By retaining memory of past inputs through the echo state property, LTC networks facilitate tasks such as time-series prediction and sequence learning. During training, the readout layer learns to map the reservoir states to desired outputs using supervised learning techniques, optimizing its parameters to minimize prediction error. This approach offers several technical advantages, including robustness to noise, scalability to high-dimensional input spaces, and efficient handling of complex, real-world datasets. From a technical standpoint, LTC networks provide a biologically plausible framework for studying neural computation, offering insights into how biological systems may process and represent temporal information. Their versatility and effectiveness make them valuable tools for a wide range of applications in computational neuroscience and machine learning research.

Recurrent neural networks incorporating continuous-time hidden states governed by ordinary differential equations (ODEs) represent powerful algorithms for modeling time series data, widely applied across medical, industrial, and business domains.The state of a neural ODE, $x(t) \in R\ D$, is defined by the solution of this equation (Chen. 2018): $dx(t)/dt = f(x(t),I(t), t, \theta)$, with a neural network f parametrized by θ. The state can be calculated using a numerical ODE solver, and the network can be trained through reverse-mode automatic differentiation. (Rumelhart, Hinton, and Williams 1986), by gradient descent through the solver (Lechner. 2019).

Instead of directly defining the derivatives of the hidden state through a neural network f, one can opt for a more stable approach, such as determining a continuous-time recurrent neural network (CT-RNN) by the following equation (Funahashi and Nakamura 1993):
$dx(t)/dt = - x(t)/\tau + f(x(t),I(t), t, \theta)$, in which the term $- x(t)/\tau$ assists the autonomous system to reach an equilibrium state with a time-constant τ . x(t) is the hidden state, I(t) is the input, t represents time, and f is parametrized by θ.

We use the following alternative formulation: let the hidden state flow of a network be declared by a system of linear ODEs of the form: $dx(t)/dt = -x(t)/\tau + S(t)$, and let $S(t) \in R^M$ represent the following nonlinearity determined by $S(t) = f(x(t),I(t), t, \theta)(A-x(t))$, with parameters θ and A. Then, by inserting in S into the hidden states equation, we get:

$$\frac{dx(t)}{dt} = -[\frac{1}{\tau} + f(x(t), I(t), t, \theta]x(t) + \quad (1)$$
$$f(x(t), I(t), t, \theta)A.$$

Equation 1 represents a novel instance of a time-continuous RNN, exhibiting numerous distinctive features and advantages:

Liquid time-constant: The concept of a liquid time-constant involves a neural network f not only determining the derivative of the hidden state x(t) but also functioning as an input-dependent varying time-constant ($\tau sys = \tau / (1 + \tau * f(x(t), I(t), t, \theta))$). This time constant serves as a parameter dictating the speed and sensitivity of coupling for an ODE. This unique property allows individual components of the hidden state to adapt to specialized dynamical systems based on incoming input features at each time step. These models are referred to as liquid time-constant recurrent neural networks (LTCs). LTCs can be implemented using various ODE solvers.

Time series modelling: Time series modeling is the analysis, prediction, and understanding of sequential data points ordered in time. It involves exploring and visualizing data, identifying stationarity and trends, applying models like ARIMA and exponential smoothing, utilizing machine learning approaches such as LSTM networks, evaluating model performance, and making forecasts for decision-making. Advanced topics include dynamic linear models, Bayesian analysis, and emerging trends in interpretability and scalability for handling large time series data.We use LTC to detect occupancy of a room and compare the performance of our model with various other models such as LSTM and CT-RNN. We observe that our model performs consistently better than the other models.

II. Related Work

In recent years, notable progress has been achieved in refining neural networks specifically designed for embedded systems. The Liquid Time-Constant (LTC) neural network, as introduced by Hasani(2021), has shown remarkable efficacy in time-series prediction tasks, maintaining consistent and constrained behavior. However, the original LTC algorithm exhibits limitations necessitating further refinement to bolster its suitability for embedded systems.

Various studies have focused on optimizing LTC neural networks alongside other cutting-edge models, such as Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM) networks. Despite their advantages, these models may encounter obstacles such as high computational overhead, memory requirements, or limited adaptability to diverse tasks and environments.

The domain of neural networks tailored for has witnessed significant advancements. One seminal contribution in this arena is the Liquid Time-Constant (LTC) neural network introduced by Hasani, which demonstrated exceptional performance in time-series prediction tasks with stable and constrained behavior.

In the subsequent sections, we elaborate on the methodology underpinning our LTC refinement efforts, elucidate the performance of our optimized LTC neural network algorithm across various machine learning tasks, and juxtapose LTC against previous iterations and other state-of-the-art models to underscore the benefits of our optimizations in terms of usability, compatibility with Keras functions, and overall code clarity. Finally, we offer insights into potential avenues for future research and applications of LTC in the domain of scalable AI.

III. Description of Dataset

For the model, We used a dataset which consists of values measured by physical sensors. The sensors used were the temperature, humidity, and CO2 concentration sensors (Candanedo and Feldheim 2016). Input data and binary labels are sampled in one-minute long intervals. The original dataset consisted of a pre-defined training and test set.

We then cut the sequences of each of the two sets into a training and test set of overlapping sub-sequences of exactly 32 time-steps. Sufficient care was taken that no item from the test set was leaking into the training set during this process. Input features of all data were normalized by the mean and standard deviation of the training set, such that the training set has zero mean and unit standard deviation. Then, we selected 10% of the training set as the validation set. All 3 datasets comprised of 5 features namely- Temperature, Humidity, Light, CO2, Humidity Ratio.

The training dataset consists of 8143 samples of data, the testing dataset consists of 9752 samples of data, and the validation dataset consists of 2665 samples of data.

IV. Problem Formulation

This paper aims to highlight the performance of LTC as compared to CT-RNN and LSTM. It also aims to highlight the improved accuracy obtained as compared to the base paper.

The LTC model predcits whether a room has been occupied or not, using the readings from 5 sensors- Temperature, Humidity, Light, CO2, and Humidity Ratio. We then compare the performance of our model with the base paper's model and other models such as CT-RNN and LSTM in terms of training loss and accuracy, validation loss and accuracy, testing loss and accuracy.

V. Methodology

To develop our LTC model we have implmented the following steps:

Implementing the algorithm from the base paper: In order to accurately compare our models, we have implemented the exact algorithm from which the model is built:

Algorithm 1 LTC update by fused ODE Solver

Parameters: $\theta = \{\tau^{(N\times 1)}$ = time-constant, $\gamma^{(M\times N)}$ = weights, $\gamma_r^{(N\times N)}$ = recurrent weights, $\mu^{(N\times 1)}$ = biases$\}$, $A^{(N\times 1)}$ = bias vector, L = Number of unfolding steps, Δt = step size, N = Number of neurons,
Inputs: M-dimensional Input $\mathbf{I}(t)$ of length T, $\mathbf{x}(0)$
Output: Next LTC neural state $\mathbf{x}_{t+\Delta t}$
Function: FusedStep($\mathbf{x}(t)$, $\mathbf{I}(t)$, Δt, θ)
$\mathbf{x}(t+\Delta t)^{(N\times T)} = \frac{\mathbf{x}(t) + \Delta t f(\mathbf{x}(t),\mathbf{I}(t),t,\theta)\odot A}{1+\Delta t\left(1/\tau + f(\mathbf{x}(t),\mathbf{I}(t),t,\theta)\right)}$
▷ $f(.)$, and all divisions are applied element-wise.
▷ $\odot$ is the Hadamard product.
end Function
$\mathbf{x}_{t+\Delta t} = \mathbf{x}(t)$
for $i = 1 \ldots L$ **do**
  $\mathbf{x}_{t+\Delta t}$ = FusedStep($\mathbf{x}(t)$, $\mathbf{I}(t)$, Δt, θ)
end for
return $\mathbf{x}_{t+\Delta t}$

Algorithm 2 Training LTC by BPTT

Inputs: Dataset of traces $[I(t), y(t)]$ of length T, RNN-cell $= f(I, x)$
Parameter: Loss func $L(\theta)$, initial param θ_0, learning rate α, Output w $= W_{out}$, and bias $= b_{out}$
for $i = 1 \ldots$ number of training steps **do**
  (I_b, y_b) = Sample training batch, $x := x_{t_0} \sim p(x_{t_0})$
  for $j = 1 \ldots T$ **do**
    $x = f(I(t), x)$, $\hat{y}(t) = W_{out}.x + b_{out}$, $L_{total} = \sum_{j=1}^{T} L(y_j(t), \hat{y}_j(t))$, $\nabla L(\theta) = \frac{\partial L_{tot}}{\partial \theta}$
    $\theta = \theta - \alpha \nabla L(\theta)$
  end for
end for
return θ

Further, we have made several improvements on this algorithm such as updating the LTCCell class and CT-RNNCell to inherit from tf.keras.layers.AbstractRNNCell for better compatibility with TensorFlow 2.x.
We have also use the following parameters for building the model which is the same paramters used in the base paper:

Parameter	Value	Description
Number of hidden units	32	
Minibatch size	16	
Learning rate	0.001 - 0.02	
ODE-solver step	1/6	relative to input sampling period
Optimizer	Adam (Kingma and Ba 2014)	
β_1	0.9	Parameter of Adam
β_2	0.999	Parameter of Adam
$\hat{\epsilon}$	1e-08	Parameter of Adam
BPTT length	32	Backpropagation through time length in time-steps
Validation evaluation interval	1	Every x-th epoch the validation metric will be evaluated
Training epochs	200	

Improving the LTC model: To achieve a better accuracy we decided to use the following techniques:

1. Hyperparameter Tuning: We have set the learning rate=0.1 as opposed to the learning rate=0.005 used in the base paper. We observed that our model converges at learning rate=0.1.
2. Gradient Descent: In our model, Gradient Descent is used to optimize the parameters of the neural network, with the learning rate specified as a hyperparameter. The model updates its parameters in the direction that reduces the loss, ultimately improving its ability to make accurate predictions on unseen data.

Even though both the models are using Adam optimizer, we have used Gradient Descent technique along with Adam optimizer. The Adam optimizer is used for training the model whereas the gradient descent is used in the fitting of the model.
The benefits of using Gradient Descent along with Adam optimizer is:
1). Fine-tuning Learning Rate: While Adam adjusts the learning rate adaptively based on the moving average of past gradients and squared gradients, using gradient descent on top allows further fine-tuning of the learning rate. This can be particularly useful in scenarios where the adaptive learning rate might not be optimal for every parameter or during certain stages of training.
2). Stability and Robustness: Sometimes, the adaptive nature of optimizers like Adam might lead to oscillations or instability in the training process, especially with certain architectures or datasets. Incorporating gradient descent alongside Adam can provide additional stability and robustness to the optimization process.
3). Exploration of Different Optima: Gradient descent, being a more straightforward optimization algorithm, might help explore different regions of the loss landscape that Adam might not efficiently explore. This can be beneficial in finding diverse or alternative minima, potentially leading to better generalization or improved performance on the validation set.

VI. Results

After running our LTC model, we have observed the following performances with respect to other models by evaluating the testing accuracy:

Table-1: Testing Accuracy of the 4 models

Evaluation Metric	LSTM	CT-RNN	LTC (original)	LTC (improved)
Accuracy	**93.01%**	**95.4%**	**94.41%**	**99.04%**

From the above table, it can be seen clearly that our LTC model performs bettter than the original LTC model and that of the LSTM and CT-RNN model. We have improved the model using gradient descent technique followed by hyperparamet tuning of the learning rate.

For the model, we have used 32 hidden units and we have trained the model for 75 epochs. The batch size is 16. The sparisty level of each model is 0. The size of each model is 32.

VII. Conclusion and Future Work

In this paper, we have showcased the improvement of our model as compared to the model implemented by the base paper.We have also compared our LTC model with other Neural Network models such as LSTM and CT-RNN.

This model that uses LTC to detect room occupance can be used in police and military applications to predict if a room

or a house is occupied or not. It can also be used in hotels and offices and classrooms, where the power to the room can be switched off, if the model predicts that the room is not occupied.

Addressing long-term dependencies presents a notable challenge for Liquid Time Constant (LTC) networks, primarily due to the occurrence of the vanishing gradient phenomenon encountered during training with gradient descent. Although LTC networks demonstrate promise across a variety of time-series prediction tasks, their suitability for learning long-term dependencies in their current form is not straightforward. This challenge underscores the importance of exploring alternative training strategies or architectural modifications to better handle and capture long-term dependencies within LTC networks.

The performance of time-continuous models like LTCs is heavily reliant on the choice of ordinary differential equation (ODE) solver employed. While LTC networks exhibit commendable performance with advanced variable-step solvers and the newly introduced Fused fixed-step solver, their effectiveness diminishes when conventional off-the-shelf explicit Euler methods are utilized. This highlights the critical role played by numerical implementation approaches in determining the overall performance and efficacy of LTC networks, necessitating careful consideration and experimentation with various ODE solvers to optimize their performance.

Comparing LTC networks with other time-continuous models, such as Neural ODEs, reveals trade-offs between computational efficiency and expressivity. While Neural ODEs offer faster computation, they often lack the expressive power required for capturing complex temporal patterns. In contrast, LTC networks enhance expressiveness but at the cost of increased time and memory complexity. This trade-off underscores the need for a thorough investigation into the computational requirements and scalability of LTC networks, particularly in scenarios involving large-scale datasets or real-time processing constraints.

Moreover, LTC networks inherently possess causal structures by virtue of their differential equation semantics. This property makes LTC networks particularly well-suited for exploring causality in recurrent models, suggesting exciting avenues for future research. Investigating the causality of performant recurrent models like LTCs could yield valuable insights into their underlying dynamics and inform their application in domains requiring robust causal reasoning, such as the control of robots operating in continuous-time observation and action spaces. Leveraging LTCs in such domains has the potential to significantly enhance reasoning capabilities and improve overall system performance. Thus, further research into the causal properties and application potential of LTC networks represents a compelling and promising direction for future investigation.

Future research endeavors may involve delving into supplementary strategies to enhance the scalability of LTC, including model compression, pruning, and quantization. Exploring the amalgamation of LTC with other cutting-edge AI methodologies, such as reinforcement learning and unsupervised learning, could yield valuable insights into its applicability across a broader spectrum of use cases.

VIII. Acknowledgment

This study draws upon Michael B. Khani's MSc thesis for its foundation. Acknowledgments are due to Dr. Ramin Hasani for his invaluable contributions to the initial development of the LTC algorithm, which served as a source of inspiration and groundwork for this research.

IX. References

Ramin Hasani, Mathias Lechner, Alexander Amini, Daniela Rus, Radu Grosu, "Liquid Time-Constant Networks," Thirty-Fifth AAAI Conference on Artificial Intelligence (AAAI-21), no. [Online]. Available: https://arxiv.org/abs/2006.04439, 2021.

Hasani, R., Lechner, M., Wang, T. H., Chahine, M., Amini, A., & Rus, D. , "Liquid structural state-space models," arXiv preprint , p. arXiv:2209.12951, 2022.

Hasani, R., " Interpretable recurrent neural networks in continuous-time control environments," Doctoral dissertation, Wien, 2020.

M. Courbariaux, Y. Bengio, and J.-P. David , "BinaryConnect: Training deep neural networks with binary weights during propagations," Proc. Advances in Neural Information Processing Systems, pp. 3123-3131, 2016.

A. Krizhevsky, I. Sutskever, and G. E. Hinton, " Imagenet classification with deep convolutional neural networks," Proc. Advances in Neural Information Processing Systems, pp. 1097-1105, 2012.

T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, "Neural ordinary differential equations," Proc. Advances in Neural Information Processing Systems, pp. 6571-6583, 2018.

L. M. Candanedo and V. Feldheim, "Accurate occupancy detection of an office room from light, temperature, humidity and CO2 measurements using statistical learning models," Energy and Buildings, vol. 112, pp. 28-39, 2016.