Informatics Institute of Technology

In Collaboration With

The University of Westminster, UK

The University of Westminster, Coat of Arms

A Novel Approach to Time Series Forecasting using Liquid Time-constant Networks

A Project Methodology by

Mr. Ammar Raneez

W1761196 | 2019163

Supervised by

Mr. Torin Wirasingha

November 2022

This Project Proposal is submitted in partial fulfilment of the requirements for the BSc (Hons) Computer Science degree at the University of Westminster.

Contents

List of Tables

List of Figures

Acronyms

AI	Artificial Intelligence.
API	Application Programming Interface.
ARIMA	Autoregressive Integrated Moving Average.
BPTT	Back-Propagation Through Time.
BTC	Bitcoin.
CT-GRU	Continuous-time Gated Recurrent Unit.
CT-RNN	Continuous-time Recurrent Neural Network.
DL	Deep Learning.
GPU	Graphics Processing Unit.
LSTM	Long Short-Term Memory.
LTC	Liquid Time-constant.
ML	Machine Learning.
(s)MAPE	Symmetric Mean Absolute Product Error.
MASE	Mean Absolute Scaled Error.
MSE	Mean Squared Error.
N-BEATS	Neural Basis Expansion Analysis for interpretable Time Series
NLP	Natural Language Processing.
ODE	Ordinary Differential Equations.
POC	Proof-Of-Concept.
REST	Representational State Transfer.
RMSE	Root Mean Squared Error.
RNN	Recurrent Neural Network.
TS	Time Series.
SDE	Stochastic Differential Equations.

1. METHODOLOGY

1.1 Research methodology

Methodologies suitable for the research project have been evaluated and chosen from the predefined Saunders Research Onion Model (Saunders, Lewis and Thornhill, 2007, p102).

Table 1: Research Methodology

Philosophy	The **positivism** philosophy was chosen: although the data collected will be a collection of nominal and ordinal data, the data collected will be encoded into numbers. Additionally, as the outcome of this research, it is expected to validate/invalidate hypothesis **H**01 using necessary evaluation comparisons.
Approach	The **deductive** approach was chosen over the inductive since the final analysis and evaluation will be quantitative, which aims to deduce hypothesis **H**01.
Strategy	**Archival research** and a**ction research** were chosen as the data collection strategy. Since the research topic is modern, the principal source of data collection would be research documents. Action research will also be included since the research process will likely be an iterative approach of diagnosis, planning, action & evaluation. To assist the supplementary research taking place, **survey** was chosen to obtain insights from end users.
Choice	**Multi-method** will suit the proposed research project most since qualitative analysis would complement to the primary quantitative approach. However, it will not be used as a combination.
Time Horizon	The **cross-sectional** time horizon was chosen over the longitudinal time horizon. Even though the latest available data will have to be obtained often to update the model, there will be no interlinking between the times when the data is gathered as they will be independent.
Techniques and procedures	As a form of **data collection & analysis**, as many sources as possible will be used since there are finite resources. The primary mediums will be statistics, reports, journals, articles, and observations.

1.2 Development methodology

1.2.1 Life cycle model

Prototyping was chosen as the life cycle development methodology since heavy iterative building development and evaluation are required.

1.2.2 Requirement elicitation methodology

The author will utilize **surveys**, **interviews**, and the knowledge obtained from available **literature** to gather requirements to implement the associated software produced.

1.2.3 Design methodology

Structured System Analysis & Design Method (SSADM) was chosen as the design methodology since it is easier to understand and maintain. These are essential factors given that the project will be developed over a considerably long period. Additionally, the selected software requirements (Python & React) do not promote Object Oriented Programming (OOP) in nature, but rather, functions and modules. It is also worth mentioning that it will have little discernible benefit in the case of a data science project.

1.2.4 Software development methodology

Structured programming will accompany the chosen design methodology to facilitate development using modules and functions.

1.2.5 Evaluation methodology

A specialized version of the K-fold cross-validation: cross-validation on a rolling basis (Shrivastava, 2020), will be used as a means of evaluation. Validation is required to ensure the model is robust and does not overfit.

The evaluation metrics: MSE, RMSE, (s)MAPE, and MASE (Hyndman et al., 2021) will evaluate the system to produce adequate comparisons against existing solutions and validate or invalidate hypothesis **H**01.

Benchmarking

As no previous system utilizing an LTC architecture exists, the author will not be able to conduct a comparative benchmarking analysis on the proposed system. However, conducting baselining to capture performance for future research benchmarking is feasible and will be carried out.

1.3 Solution methodology

As mentioned, a BTC forecasting prototype will be built to create justification.

A summarized workflow that will be followed when creating the model is depicted below.

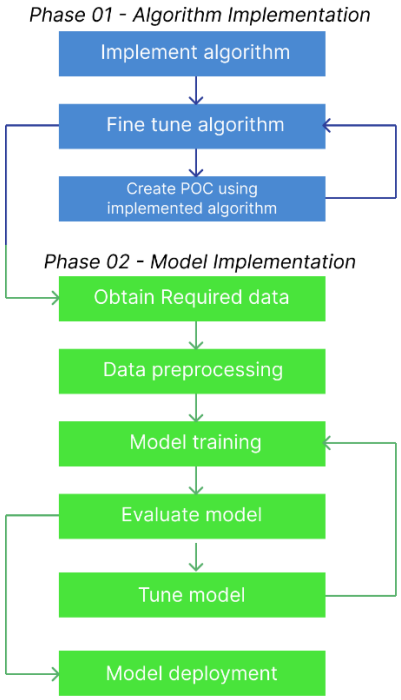

Figure 1: Model creation workflow (*Self-Composed*)

Phase 01

1.3.1 Implement the algorithm

The first and most crucial step is to implement the LTC algorithm. This step is critical since it will give the author an idea of whether the development is feasible, whether pivoting is necessary, or even if the project must change entirely. The paper by Hasani et al. (2020) will be used as a guide alongside the latent SDE talk presented by Duvenaud (2021).

1.3.2 Fine-tune the algorithm

Once satisfactory progress has been made, the code must be cleaned and fine-tuned to be scalable and generalizable.

1.3.3 Create a POC

An example POC must be implemented to validate whether the supplemental forecasting application is feasible. This step is also essential since it will give the author an idea of how the software will be built.

Creating the POC and fine-tuning will be an iterative process since minor tweaks will be done while implementing.

Phase 02

1.3.4 Obtain the required data

As identified in the literature: existing systems had been trained on data that are outdated now. To address this limitation, the data used in this project will be scraped using APIs, which will be the most up-to-date.

Furthermore, to keep the model as updated as possible, the model will be retrained periodically with the existing new data.

1.3.5 Data preprocessing

Once the data has been fetched, it must be cleaned. The APIs return redundant & unneeded columns (ex: repeated features with different names) that must first be removed. The author must apply NLP techniques to extract sentiment scores and related information extracted. Once cleaned, they can be combined, creating a single set.

Data processing for TS forecasting applications is not the same as classification or regression problems since the data is temporal – therefore, the order must be given prominence.

Creating the train and test sets is unlike other problems, as random splits will not work. The data will be split sequentially, at a point in time such that the observations before it is the train data and after it the test data, a ‘pseudo future.’ Therefore, there is no ‘leakage’ between the two sets (Hyndman et al., 2021): past data must forecast the future.

Finally, the data must be ‘windowed’ to convert into a supervised learning problem and split into features and labels (BI4ALL, 2021), which is required since past windows will predict the future.

1.3.6 Model training

Once the data windows are ready, the model can be created. Here, the developed cell will be used within an RNN layer to provide a fair comparison against other existing cells like the LSTM.

1.3.7 Evaluation

Once the model has been trained, sufficient evaluation must be conducted to shed light on the model's performance. The model will be evaluated by the metrics discussed in the evaluation methodology.

1.3.8 Tuning

If the performance obtained is subpar, the model hyperparameters must be tuned (ex: number of epochs, batch size, learning rate, optimizer, activation function, number of units & layers). Tuning mentioned hyperparameters could cause a significant change in performance – even worsen the performance. However, this is an important step that must be carried out, as it could drastically improve performance.

Training, evaluation & tuning will be an iterative process, as it is unlikely to obtain the best-performing model in the first experiment. It will also be unexpectedly long since there exists no reference implementation. Therefore, common hyperparameter values are not documented.

1.3.9 Deployment

The final step in the implementation is to deploy the forecasting model so that it can be accessed from anywhere, in this case, especially the client application.

In addition, a deployment pipeline must be built to facilitate future automatic deployments whenever the model is updated periodically.

1.4 Project management methodology

The author will follow a combination of PRINCE2 and Agile. The project will require many iterations and improvements since the implementation is novel and no reference exists. Alongside multiple iterations, it is best implemented by being divided into multiple chunks and focusing on each chunk at a time with a plan-based approach.

1.4.1 Schedule

Gantt chart

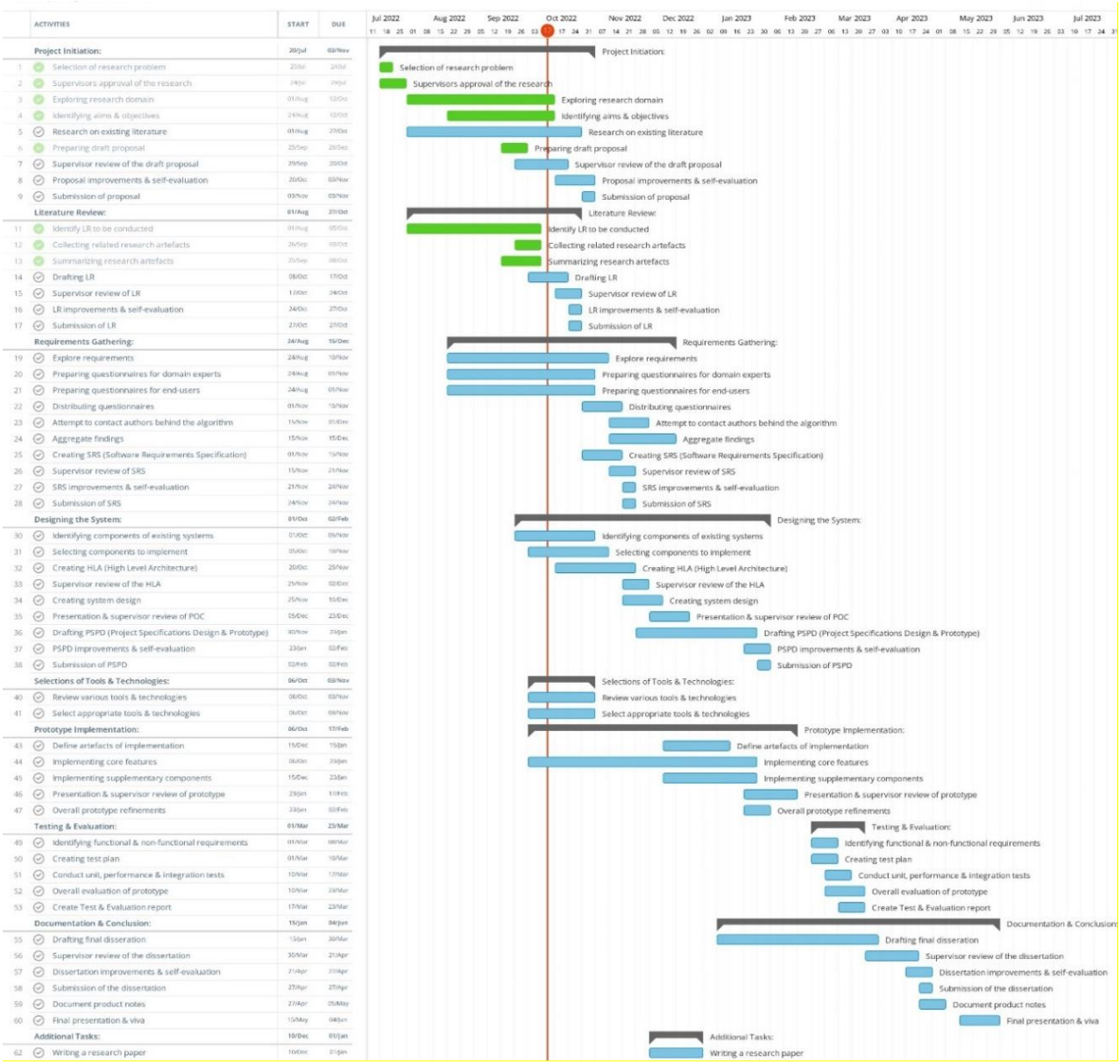

Figure 2: Gantt Chart (*Self-Composed*)

A clearer version can be found [Here]

Deliverables

Table 2: Deliverables & Dates (*Self-Composed*)

Deliverable	**Date**
Literature Review Critical analysis of related work & solutions.	27th October 2022
Project Proposal & Ethics Forms	3rd November 2022

The initial proposal of the research to be conducted.	
Software Requirement Specification Defines the requirements that must be met to prototype and collect data.	24th November 2022
Proof Of Concept & Implementation Presentation An initial implementation of the proposed system.	23rd December 2022
Project Specifications Design & Prototype A prototype of the system with the core features and an accompanying document specifying the design followed & an overview of the implemented algorithm.	8th February 2023
Test & Evaluation Report Documentation of test findings and evaluations conducted on the prototype.	23rd March 2023
Draft Project Report A draft submission of the final thesis to get evaluations.	30th March 2023
Final Thesis Final submission of the thesis with complete documentation of the project's journey.	27th April 2023

1.4.2 Resource requirements

Software requirements

- **Operating System (Windows /** Linux **/** macOS**) -** Windows will be the default since it provides easy access to the required development environments and tools. Besides, the author does not have access to other operating systems.

- **Python /** R - to create the network & the respective model. Python, since it has a much simpler learning curve, provides easy integration with other mentioned software and is optimized for large-scale ML software. Meanwhile, R is more suitable for statistical data analysis (Python vs. R: What's the Difference?, 2021)
- **TensorFlow /** Torch – provides libraries that facilitate DL in Python & R. TensorFlow, due to its large developer community and seamless integration with Keras for higher-level API development. Additionally, multiple visualizations will be required, which is made simple by TensorBoard (Dubovikov, 2018).
- **Flask /** Node / Golang – for seamless communication between the client and the model. Flask will be the primary choice since the ML component will use Python, it is incredibly lightweight, and there is only a requirement for a minimal REST API. Node and Golang are secondary options if there are requirements to add additional features, such as authentication, that are not directly relevant to the research.
- **React** / Vue / Svelte – required to develop the client-side. A fast performant library is required to prevent lags and other performance issues. React will be the option because of the author's familiarity and large community. Svelte and Vue will be the second options if React is not performant enough. Angular is not considered since it is less performant and more suitable for larger-scale applications, which is not required. (React vs Vue vs Angular vs Svelte, 2020).
- **VSCode /** PyCharm – environment to facilitate application development. VSCode is the primary choice since it provides a general-purpose yet lightweight development experience with multiple plugins making it more developer-friendly. PyCharm will be a secondary option if there are issues with the Python environment or a need for a dedicated python development environment.
- **Jupyter Notebook /** Google Colab – development environment for building the forecasting model. Jupyter will be the primary choice as it has less risk: it runs locally. Therefore, in case of power failures, training would not be interrupted. Colab will be the backup choice if there is a requirement for a GPU to train the model.
- **Zotero** / Mendeley – manage references and research artefacts. Zotero is chosen due to the author's preference and is easy to use.

- **MS Office | Overleaf** – tools to create primary research reports. A combination of MS Office and Overleaf will facilitate the development of the project submission documents and the creation of professional research papers/surveys/reviews, if necessary.
- **Google Drive** / OneDrive – to backup research artefacts. Google Drive will be the primary option due to the unlimited storage availability provided by the university.
- **Figma | Canva | Draw.io –** tools to create figures and diagrams. Combining all three will streamline the design process, as each has its advantages. Figma - to design & prototype; Draw.io – to draw flow charts and other associated diagrams; Canva - to prepare attractive presentation slides.
- **GitHub** / Bitbucket / Gitlab – track, version & manage development code & research documents. GitHub will be the choice due to the author's familiarity, integrations with the development environments, and email notifications that could be significant.

Hardware requirements

- **Core i5 Processor (8th gen) /** Ryzen 5 / M1 – for long-running intensive workloads and managing multiple development environments. The author has access to an intel processor; therefore, it will be the CPU choice. However, the other options would suffice just as much.
- **8GB Ram or above** – to manage model training, multiple development environments & multitasking. Moreover, it is required to load large datasets and multitask.
- **Disk space of approx. 20GB** – to store application code & data.

If the available hardware does not meet the required criteria, a cloud-based development environment can be used (ex: GitHub codespaces, Google Colab, Zotero web, Google Drive).

Data requirements

- **BTC price observations & block reward size** – from a financial website (ex: investing.com, cmcmarkets.com, finance.yahoo.com).
- **BTC tweets** – from the Twitter API or a respective website that provides the required data (ex: bitinfocharts.com).
- **Google trends –** from a Python API (PyTrends) that supplies Google Trends data.

Skill requirements

- Creation of TS forecasting systems.
- Knowledge of ODEs & respective solvers.
- Implementation of a raw neural ODE and SDE.
- Ability to create optimized & scalable DL models.
- Ability to develop optimized client-side charts & user interfaces that dynamically update.
- Research & academic writing skills.

1.4.3 Risk management

The following table identifies possible risks that the author could face and how they could mitigate them.

Table 3: Risk Management Plan (*Self-Composed*)

Risk Item	Severity	Frequency	Mitigation Plan
Lack of required knowledge	5	5	Get insights from domain experts and, if necessary, the author of the proposed algorithm.
Corrupted documentation	4	4	Store all necessary documentation on the cloud as well as external storage.
Lose access to development code	5	2	Backup code on source control and cloud storage.
Inability to deliver all expected deliverables	4	2	Follow a list of priorities and deliver accordingly.
Invalid hypothesis **H**01	3	2	Continue researching since the final output is a research contribution regardless.

REFERENCES

Hasani, R. et al. (2020). Liquid Time-constant Networks. Available from https://doi.org/10.48550/arXiv.2006.04439 [Accessed 25 September 2022].

Saunders, M.N.K., Lewis, P. and Thornhill, A. (2007). *Research methods for business students*, 4th ed. Harlow, England ; New York: Financial Times/Prentice Hall.

Shrivastava, S. (2020). Cross Validation in Time Series. *Medium*. Available from https://medium.com/@soumyachess1496/cross-validation-in-time-series-566ae4981ce4 [Accessed 12 October 2022].

BI4ALL. (2021). Supervised Machine Learning in Time Series Forecasting. *BI4ALL - Turning Data Into Insights*. Available from https://www.bi4all.pt/en/news/en-blog/supervised-machine-learning-in-time-series-forecasting/ [Accessed 12 October 2022].

Hyndman, R.J., & Athanasopoulos, G. (2021). *Forecasting: principles and practice*, 3rd edition, OTexts: Melbourne, Australia. Available from https://otexts.com/fpp3/. [Accessed on 30 Sep. 2022].

Dubovikov, K. (2018). PyTorch vs TensorFlow — spotting the difference. *Medium*. Available from https://towardsdatascience.com/pytorch-vs-tensorflow-spotting-the-difference-25c75777377b [Accessed 18 October 2022].

React vs Vue vs Angular vs Svelte. (2020). *DEV Community* 👩‍💻👨‍💻. Available from https://dev.to/hb/react-vs-vue-vs-angular-vs-svelte-1fdm [Accessed 18 October 2022].

Python vs. R: What's the Difference? (2021). Available from https://www.ibm.com/cloud/blog/python-vs-r [Accessed 18 October 2022].

Duvenaud, D (2021). Directions in ML: Latent Stochastic Differential Equations: An Unexplored Model Class. *YouTube*. Available from https://www.youtube.com/watch?v=6iEjF08xgBg. [Accessed on 30 Sep. 2022].