A Universal Approach to Reduce Computation Complexity: Represent Dynamical Large-Scale High-Resolution Numerical Models by Machine Learning Models

MSc Thesis proposal

H.R.A. ten Eikelder
h.r.a.teneikelder@uu.nl

April 2023

1st supervisor: Prof. dr. D. Karssenberg
2nd supervisor: MSc. O. Pomarol Moya

Department of Graduate School of Natural Sciences
Faculty of Science

Universiteit Utrecht

Contents

1 Introduction

Large-scale high-resolution numerical models are widely used in many fields. These models are used to represent a large variety of complex systems that are governed by physical patterns. These models can, due to the complexity of the underlying physical pattern, be very slow to calculate. In this research, we will only discuss grid-based models. The models evaluated in this research are Cellular Automata for Game-of-Life and forest fire simulations, and watershed models that describe the patterns of the flow of water in an area.

Many systems that are modelled by simulation models are dynamical in nature, this means that their state varies over time. In some situations the inputs can also change over time. To represent this evolution of the system in dynamical numerical models an update function is applied. This function calculates a set of state variables for the next time step. This function is applied to all grid cells, as a function of the current state variables, parameters and input. This can be represented by the following equation,

$$Z(x, y, t+1) = f(Z(x, y, t), Z(xn, yn, t), I(x, y, t), I(xn, yn, t), p(x, y), p(xn, yn)), \forall x, y \in S \tag{1}$$

Where, Z is a vector of state variables, I is a vector of inputs (drivers, boundary conditions), p is a vector of parameters or constant inputs (elevation maps, slopes), x and y represent the index of a cell, xn and yn are the vectors that represent neighbouring cells (e.g. Neumann neighbourhood, cells in upstream area), S represents the set with all cell indices of the grid and f() is the update function (the model, e.g. cellular automata).

The goal of this research is to use these existing models to train a machine learning (ML) model with a lower computational complexity. In general the ML models can be designed to be less complex then the large-scale numerical models. The ML model can make a statistical approximation of the underlying mathematical pattern that governs the more complex large-scale simulation, see Eq 1. The ML model will be represented by the following equation,

$$Z(x, y, t+1) = F(Z(x, y, t), Z(xn, yn, t), I(x, y, t), I(xn, yn, t), p(x, y), p(xn, yn)), \forall x, y \in S. \tag{2}$$

Where Z, I, p, x, y, xn,yn and S all represent the same values as in Eq. 1, F() now represents the ML model that replaces the complex update function.

The ML model in Eq. 2 can possibly have identical input as the simulation model in Eq. 1. The input does not necessarily have to be identical, these mathematical representations are very general and can apply to all variations that may occur in this research. Some ML models will only use the static input or disregard the neighbourhood relations that are apparent in the simulation model.

State of the art

This approach has been executed in climate and weather modelling. This research shows that there have been instances where the predictions of ML methods are faster than large numerical models [1, 2]. Less complex ML models have the potential to provide faster predictions. However, it is unclear to see how decisions are made in some ML models which are considered a black box. In cases where the underlying physical model is not of interest the latter is not considered a problem.

Knowledge gap

There is no universal approach to ML representations of large-scale numerical models. This research aims to evaluate a general approach for representing grid-based models with machine learning as a stepping stone to closing this gap.

2 Main- and sub-research questions

The main research question can be formulated as follows: "How effective can machine learning models represent dynamic large-scale numerical simulation models and how does this approach influence computational complexity and execution time?"

This research question can be divided in the following sub-questions.

1. Can a ML model accurately represent a simple numerical simulation model?
2. How well does the proposed approach work on different types of simple simulation models?
3. How well does the proposed approach work on more complex simulation models?
4. Can the proposed approach significantly reduce computational complexity and increase the speed of numerical simulations?
5. How can the software be designed to allow users to choose the best machine learning model for their specific simulation application?
6. Which machine learning models are most effective at representing different types of simulation models?

3 Methods

Due to the time limit the scope of this research will be limited to raster based models. This research will be divided into four stages. The main focus of this research will be on the universality of this approach to reproducing large-scale numerical models with machine learning.

Simultaneous to stage one, literature is researched to evaluate which ML model is best suited for stage one and which should be added to the software in stage 5.

3.1 Description of each stage in the process

Stage one - Version 1
Stage one will contain the development of a random forest model(optional, evaluate literature) to represent a simple numerical simulation model. The simplicity of the simulation model lies in its size and inputs. The inputs in this stage will be consistent throughout the simulation. The simulation model in this stage will be the game of life with a predefined start. The game of life simulation will contain a 'glider' of pixels that moves from the top left to the bottom right in the simulation. This stage in the research aims to answer sub-question 1.

Stage two - Test
In stage two, the software designed in stage one will be tested on different types of simulation models, such as random initiated game of life, simple forest fire and water drainage models. This stage of the research aims to answer sub-question 2.

Stage three - Version 2
In stage three, more complex simulation models are introduced to the software. In this stage dynamical simulation models are introduced, this entails that the input drivers that can vary over time steps during the simulation. This stage aims to answer sub-questions 3 and 4.

Stage four - Version 3
Stage four aims at the universality of the software. During this stage, the research will be focused on rewriting the software to make work with any type of grid-based simulation. The aim will also be to let the user specify the machine learning algorithm and feature of interest they need for their research. This stage aims to answer sub-question 5.

(Optional)
If time allows, the research will switch direction towards evaluating different machine learning algorithms for different simulation models. This part of the research will aim to answer sub-question 6.

3.2 Universal structure of the data

The following subsection describes how the data will be structured in the software. The simulation models can vary in several aspects such as their number of features, static and dynamic inputs or maps. For this reason a general approach is needed to structuring the data. All simulation models in this research will be restricted to raster-based models. This means that they are split up into pixels. Each feature is split up into the same raster. See Figure 1 for an illustration of a grid based system.

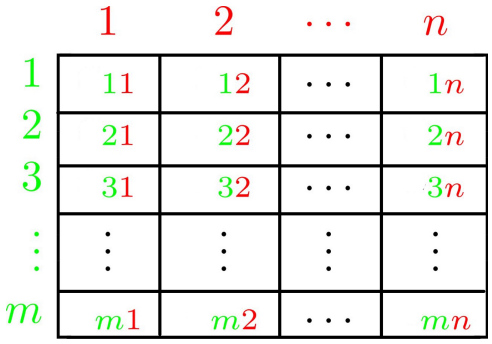

Figure 1: Grid size of simulation model is n x m pixels.

Table 1: Structural overview for the universal dataframe that represents the training data for machine learning algorithm based on any grid-based simulation model. Grid size of simulation model is n x m pixels. In this table the simulation ran from t0 to t final. The thick lines indicate a new time step in the simulated data, purely indicative in this table.

[row, column]	Feature of interest	Map 1	...	Map i	Feature 1	...	Feature j	Driver 1	...	Driver k	Train label
[1 , 1]	$Z(t0)_{1,1}$	..	..	..	..	..	..	..	..	..	$Z(t+1)_{1,1}$
[1 , 2]	$Z(t0)_{1,2}$	..	..	..	..	..	..	..	..	..	$Z(t+1)_{1,2}$
[1 , ..]	$Z(t0)_{1,..}$	..	..	..	..	..	..	..	..	..	$Z(t+1)_{1,..}$
[1 , n]	$Z(t0)_{1,n}$	..	..	..	..	..	..	..	..	..	$Z(t+1)_{1,n}$
[2 , 1]	$Z(t0)_{2,1}$	..	..	..	..	..	..	..	..	..	$Z(t+1)_{2,1}$
[2 , 2]	$Z(t0)_{2,2}$	..	..	..	..	..	..	..	..	..	$Z(t+1)_{2,2}$
[2 , ..]	$Z(t0)_{2,..}$	..	..	..	..	..	..	..	..	..	$Z(t+1)_{2,..}$
[2 , n]	$Z(t0)_{2,n}$	..	..	..	..	..	..	..	..	..	$Z(t+1)_{2,n}$
[m , 1]	$Z(t0)_{m,1}$	..	..	..	..	..	..	..	..	..	$Z(t+1)_{m,1}$
[m , 2]	$Z(t0)_{m,2}$	..	..	..	..	..	..	..	..	..	$Z(t+1)_{m,2}$
[m , ..]	$Z(t0)_{m,..}$	..	..	..	..	..	..	..	..	..	$Z(t+1)_{m,..}$
[m , n]	$Z(t0)_{m,n}$	..	..	..	..	..	..	..	..	..	$Z(t+1_{m,n}$
[1 , 1]	$Z(t1)_{1,1}$	..	..	..	..	..	..	..	..	..	$Z(t2)_{1,1}$
[1 , 2]	$Z(t1)_{1,2}$	..	..	..	..	..	..	..	..	..	$Z(t2)_{1,2}$
[1 , ..]	$Z(t1)_{1,..}$	..	..	..	..	..	..	..	..	..	$Z(t2)_{1,..}$
[1 , n]	$Z(t1)_{1,n}$	..	..	..	..	..	..	..	..	..	$Z(t2)_{1,n}$
[2 , 1]	$Z(t1)_{2,1}$	..	..	..	..	..	..	..	..	..	$Z(t2)_{2,1}$
[2 , 2]	$Z(t1)_{2,2}$	..	..	..	..	..	..	..	..	..	$Z(t2)_{2,2}$
[2 , ..]	$Z(t1)_{2,..}$	..	..	..	..	..	..	..	..	..	$Z(t2)_{2,..}$
[2 , n]	$Z(t1)_{2,n}$	..	..	..	..	..	..	..	..	..	$Z(t2)_{2,n}$
[m , 1]	$Z(t1)_{m,1}$	..	..	..	..	..	..	..	..	..	$Z(t2)_{m,1}$
[m , 2]	$Z(t1)_{m,2}$	..	..	..	..	..	..	..	..	..	$Z(t2)_{m,2}$
[m , ..]	$Z(t1)_{m,..}$	..	..	..	..	..	..	..	..	..	$Z(t2)_{m,..}$
[m , n]	$Z(t1)_{m,n}$	..	..	..	..	..	..	..	..	..	$Z(t2)_{m,n}$
[1 , 1]	$Z(t2)_{1,1}$	..	..	..	..	..	..	..	..	..	$Z(t3)_{1,1}$
[1 , 2]	$Z(t2)_{1,2}$	..	..	..	..	..	..	..	..	..	$Z(t3)_{1,2}$
[..]	$Z(t2)_{..}$	..	..	..	..	..	..	..	..	..	$Z(t3)_{..}$
[..]	$Z(t..)_{..}$	..	..	..	..	..	..	..	..	..	$Z(t..)_{..}$
[..]	$Z(tfinal-1)_{..}$	..	..	..	..	..	..	..	..	..	$Z(tfinal)_{..}$

Each feature, static or dynamic input, map etc., can be implemented as a column. Each pixel is represented by a row. For the entire map of pixels as rows, the simulated time step is represented in this data frame. See Table 1 for

an example of this method. The thick lines in this example represent one time step in the simulation. In this way the machine learning model can be trained on the label of the next time step, for all time steps simultaneously.

The structure in Table 1 is universal, it represents the training data for a machine learning algorithm based on data of any grid-based simulation model. If one would be interested in only one pixel, for example the time series of the outflow of water at one location, the outer right column can be adjusted. All the 'labels' will then only contain the information of the pixel of interest and not the rest of the pixels.

4 Planning

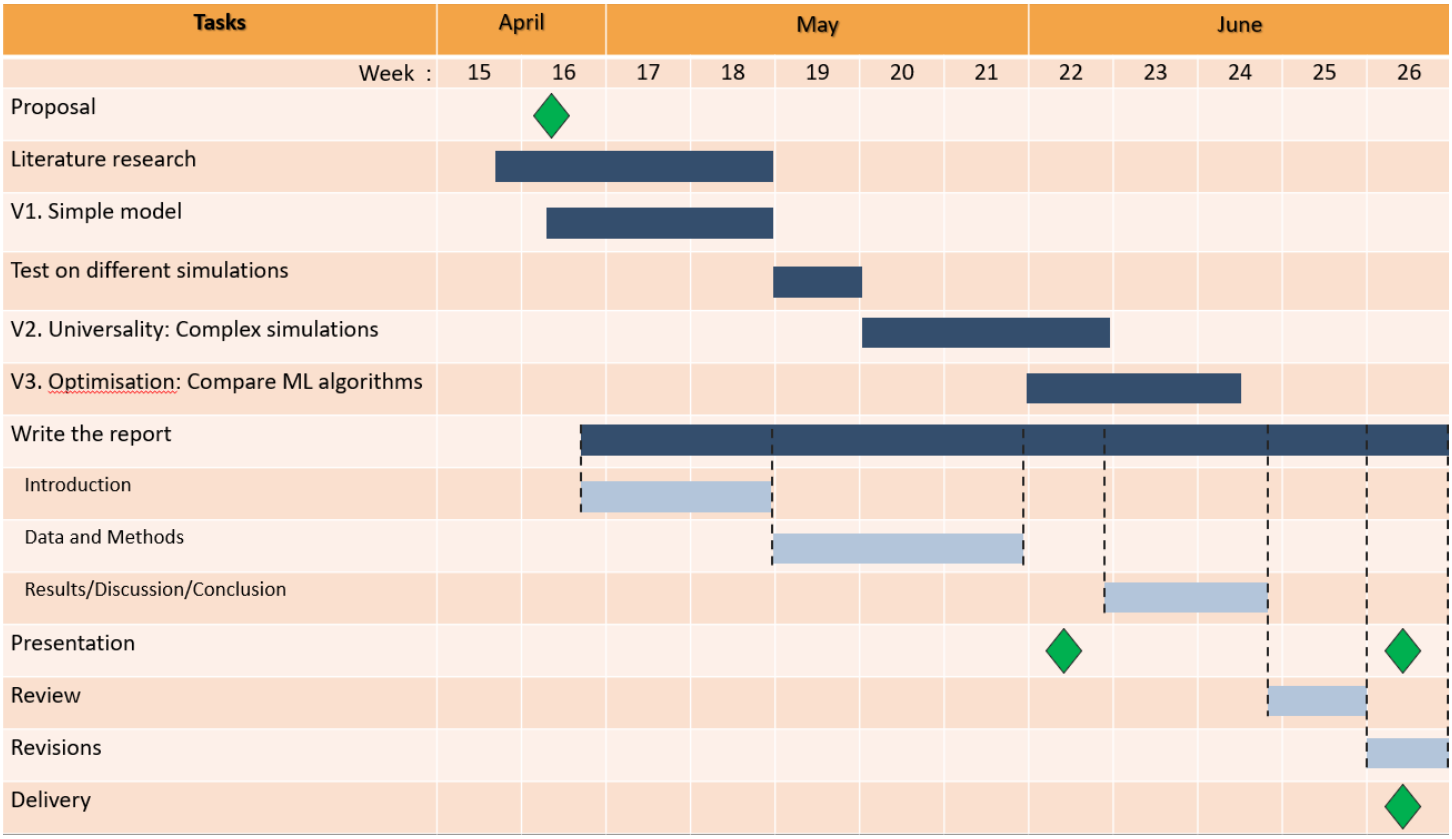

Figure 2: Planning in the form of a Gannt chart of the MSc project.

5 Organisation

University of Utrecht - Faculty of Science, Graduate School of Natural Sciences

Princetonlaan 8a, 3584 CB Utrecht, The Netherlands

Supervisors

First supervisor: Prof. dr. Derek Karssenberg - d.karssenberg@uu.nl
Second supervisor: MSc. Oriol Pomarol Moya – o.pomarolmoya@uu.nl

Examiners

First Examiner: Prof. dr. Derek Karssenberg - d.karssenberg@uu.nl
Second Examiner: person – <email>

Meetings

Meetings with the first supervisor will occur once every two weeks. For these meetings a small presentation will be prepared with the current state of the project and the desired next steps. These meetings will also be used to evaluate if any changes need to be made to the planning.

Meetings with the second supervisor will be conducted as needed. The second supervisor is available for questions and guidance.

Deliverables

- Software containing the simulation models used during the research
- Software of the ML adaptation of the simulation model
- Report
- Presentation (poster/ pptx)

6 Data management plan

Software
The software will be locally developed and daily pushed to a Github repository to ensure access to the most recent scripts.
Storage
The simulation data can quickly rise to many gigabytes, in that case Google drive is used for saving the data.
File system and naming convention
These will be decided on during the project. For the end product there will be a clear structured file system with a readme.txt that includes the overview.

Ethical evaluation of data usage

The Data Ethics Decision Aid (DEDA) will be used during the project. DEDA is developed by the Utrecht Dataschool to help guide responsible data usage for all data related projects. Since in this project simulated data is used this will only be a very small part of the report.

References

[1] Matthew Chantry et al. “Opportunities and challenges for machine learning in weather and climate modelling: hard, medium and soft AI”. In: *Philosophical Transactions of the Royal Society A* 379.2194 (2021), p. 20200083.

[2] Catherine Odelia de Burgh-Day and Tennessee Leeuwenburg. “Machine Learning for numerical weather and climate modelling: a review”. In: *EGUsphere* 2023 (2023), pp. 1–48.