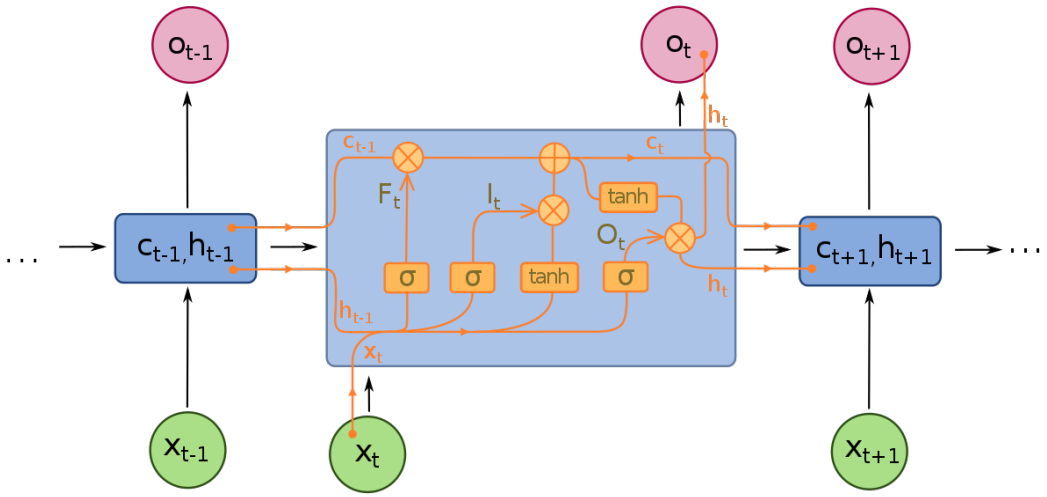
CHALMERS
UNIVERSITY OF TECHNOLOGY



Groundwater and gravity modelling using recurrent neural networks

Master's thesis in Complex Adaptive Systems

Magnus Våge

Department of Space, Earth and Environment
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2021

Master's thesis

Groundwater and gravity modelling using recurrent neural networks

MAGNUS VÅGE

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Space, Earth and Environment
Onsala Space Observatory Division
Chalmers University of Technology
Gothenburg, Sweden 2021

Groundwater and gravity modelling using recurrent neural networks
MAGNUS VÅGE

Supervisor: Maxime Mouyen, Department of Space, Earth and Environment
Examiner: Rüdiger Haas, Department of Space, Earth and Environment

Master's Thesis
Department of Space, Earth and Environment
Onsala Space Observatory Division
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: An LSTM unit [1]

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2021

Groundwater and gravity modelling using recurrent neural networks
MAGNUS VÅGE
Department of Space, Earth and Environment
Chalmers University of Technology

Abstract

Artificial intelligence has gained a lot of interest in recent years due to its impressive performance on a variety of prediction and classification tasks. These results are in large part a result of the increase in computational capacity and the availability of large data sets.
Here we apply artificial intelligence on the problem of predicting local variations of gravity due to local hydrological effects at the Onsala Space Observatory. Hydrological effects imply redistribution of mass in close proximity to the measuring equipment, and therefore contributes to the variations seen in the local strength of gravity. The objective of this thesis is to model groundwater and residual gravity levels based on data from a normal weather station, giving us the ability to construct useful simulated groundwater and residual gravity data measurements without the actual equipment in place. The modelling is performed using recurrent neural networks containing LSTM units, and with some convolutional preprocessing of the input data. The resulting models perform very well for predicting groundwater levels, reaching an RMSE between observed and predicted values of 0.046 m over the course of 2019 which is comparable to other results using neural networks to predict hourly groundwater levels ([2], [3]).
The neural networks initially performed quite poorly when predicting residual gravity. However, by utilizing our good results from the groundwater predictions, we extended our available data set from about 3 to almost 8 years. The best model trained on the 8 year data set achieved an RMSE of 5.961 nms^{-2} on the test set, a more than 6-fold improvement over the networks trained on the 3 year data set.
The conclusion is that recurrent neural networks are suitable for modelling groundwater levels and residual gravity, but their performance is highly dependent on the amount of data available and the preproceessing applied to the data.

Keywords: rnn, lstm, gravity, groundwater, seq2seq, cnn.

Acknowledgements

I would like to extend a very big thank you to Maxime Mouyen, for lending me his expertise and problem solving abilities for the duration of this project. I would also like to thank Mats Granath and Martin Eriksson for rewarding discussions.
The computations were enabled by resources provided by the Swedish National Infrastructure for Computing (SNIC) at Chalmers Centre for Computational Science and Engineering (C3SE) partially funded by the Swedish Research Council through grant agreement no. 2018-05973. A total of approximately 100 000 core hours were used for the computations that this report is based upon.

Lastly, I want to thank Cornelia and Sigge for all your love and support.

Magnus Våge, Gothenburg, February 2021

Contents

List of Figures

List of Tables

1
Introduction

Onsala Space Observatory has been operating a weather station and a superconducting gravimeter since 2009, and a well for groundwater level measurements since 2015. The groundwater level represents a variations of a significant mass in close proximity to the gravimeter, and is therefore an important contributing factor the the variations seen in the local strength of gravity [4], so it would be very valuable to be able to reconstruct the groundwater level from available meteorological data for the time period before the actual well measurements were available. Reconstructed groundwater levels for 2009-2015 would ideally provide 6 more years of data on which to base models linking hydrological processes to gravity variations.
For that purpose, this thesis will explore the feasibility in modelling the groundwater level based on meteorological data, and also model the residual gravity based on meteorological data and groundwater level measurements. The task of identifying the effect from hydrological processes on gravity is challenging [5]. In this thesis we will evaluate how recent advancements within artificial intelligence and machine learning can aid in this task. The area of groundwater modelling using artificial intelligence is getting significant attention from the scientific community [6], however only 4 out of 67 papers reviewed in [6] attempt to model the groundwater on an hourly basis, which is what we will do in this project.
The model is not a simple regression problem, since the current groundwater level not only depends on current rainfall, but also historical rainfall, discharge rates etc. For this purpose, we will use recurrent neural networks with LSTM (Long short-term memory) units for the modelling, since these types of networks are especially well suited for this type of problem, and the networks are capable of maintaining a memory of previous events when predicting current events. The residual gravity signal will need extensive preprocessing in order to remove effects we are unable to model with the available input data.
The thesis start with an overview of the relevant theory surrounding recurrent neural networks, LSTM units, the superconducting gravimeter, and the phenomena we want to remove from the superconducting gravimeter signal. The method section covers how this preprocessing is performed, and what recurrent neural network architectures are used. The results and conclusion sections covers the actual performance of the different architectures, and makes suggestions for further improvements of the models.

2

Theory

2.1 Artifical intelligence and machine learning

Machine learning, a field within the broader field of artificial intelligence, focuses on the task of constructing "computer systems that automatically improve through experience. It has progressed dramatically over the past two decades, from laboratory curiosity to a practical technology in widespread commercial use" [7]. Some recent success stories powered by machine learning can be seen in Figure 2.1.

Geosciences, a field that for a long time has been gathering high-quality data from both satellites and earth based systems, are also benefiting from the progress of machine learning [8].

Within the field of machine learning, methods are broadly divided into supervised, unsupervised and reinforcement learning techniques. Unsupervised learning is used in situations where you do not have a priori knowledge of the correct output for a given input. Reinforcement learning methods are used when training agents to perform optimally in an environment, and that agent can affect the environment it operates in. In this project, where we are trying to predict a known time series using other time series as input, we will use supervised learning techniques. For more information and details, we recommend the excellent book [9], which is the source of much of the information in this section.

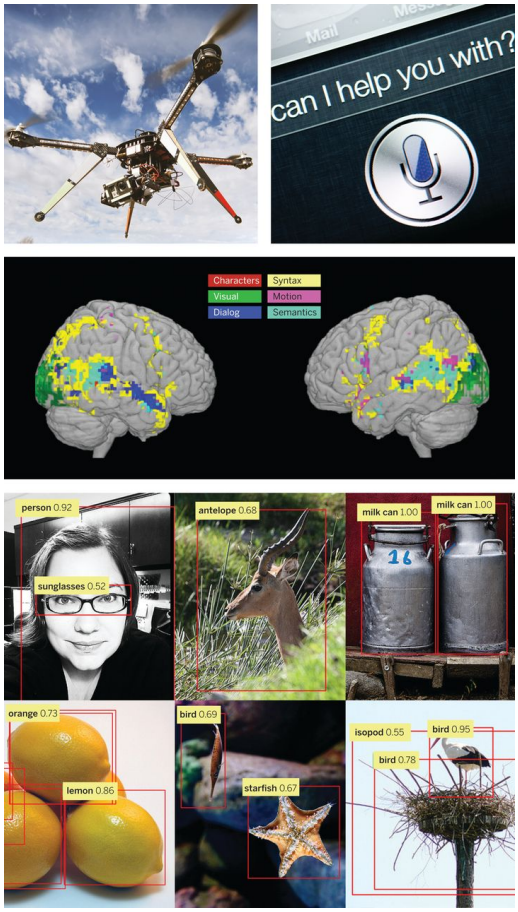


Figure 2.1: Recent success stories within the field of machine learning. Clockwise from top left: Autonomous vehicle control, speech recognition, neuroscience, computer vision. Image from [7].

2.1.1 Supervised learning

Supervised learning methods work by being supplied training data in the form of the input, which could be more or less anything; a vector, an image, a piece of video etc. Also supplied is the correct output for this input. The supervised learning method chosen then tries to infer a function mapping the input to the output, which in the ideal scenario also generalizes to produce the correct output for new pairs of input/output, not seen during training. The form of the output depends on the type of learning problem. In classification problems, the output is the correct class. A classical example is object recognition, where the input is a series of images, and the output is the correct classification of object(s) present in the images. In regression problems, the output is the correct output value for the given input value. An example would be predicting the value of a piece of real estate (the output) based on size, number of rooms, previous sales in the area etc. (the inputs).

2.1.1.1 Data set division

Within supervised learning, the data that is to be used is usually divided into three sets: Training, validation and test. The training set is used by the machine learning algorithm when inferring the function mapping from input to output.
Periodically during training, the model in its current state is applied to the data in the validation set to provide an unbiased estimate of the performance of the model. The information gained from the validation set can for instance be used to determine when training should stop, or if any parameters within the model should be adjusted.
Once training is completed, the final model is applied to the test set, which played no part whatsoever in the training process, in order to gauge the final performance of the model. A good rule of thumb when dividing data into sets is 70/15/15 % for training/validation/test sets.

2.1.2 Neural networks

One supervised learning method is the neural network. They are mathematical constructs originally inspired by how the neurons in the human brain works, even though modern artificial neural networks bear little resemblance to its biological counterpart. Input data is fed to a number of processing units called (input) neurons, the output of which are then fed on to further neurons, until it reaches a set of neuron(s) designated as the output neurons. The neurons in a neural network are organized in layers, called the input layer, hidden layer(s) and output layer as seen in Figure 2.2. Each vertical row of neurons in Figure 2.2 of neurons constitute a layer, and the arrows connecting layers with each other represents connections between layers. Neurons within a layer are normally not connected to each other (with one important exception, the recurrent neural network, which will be introduced later). In Figure 2.2 the layers are sequential, but that is not a requirement. The internal structure of a neural network could take on basically any shape, with for example the output from one layer splitting into two distinct paths containing different number of layers, and then merging together again before reaching the output layer.

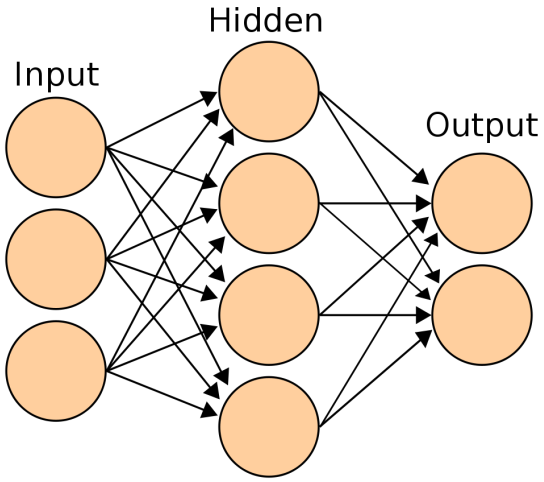


Figure 2.2: Conceptual structure of a neural network [10].

2.1.2.1 The artificial neuron

Each neuron in a layer has one-way connections to neurons in the layer before and after it. Data flows from the left to the right in the example network in Figure 2.2. Each neuron k in a layer has m connections to the m neurons in the preceding layer, and each of these connections has a weight w_{kj} associated with it, with j being the index of the connections ($1 \leq j \leq m$). Each neuron also has an internal bias, b_k. Each neuron performs the operation in Equation 2.1

$$y_k = \varphi\left(\sum_{j=1}^{m} w_{kj}x_j + b_k\right), \tag{2.1}$$

which is visualized in Figure 2.3. In Equation 2.1, y_k is the output value of neuron k and x_j is the output value of neuron j in the preceding layer. w_{kj} and b_k are the trainable parameters of the neuron. They are usually initialized to some random small value, and then optimized through training. φ is called the activation function. One main task of the activation function is to regulate the output of the neuron. Without an activation function, the neuron would just output a linear combination of the inputs, $y_k = \sum_{j=1}^{m} w_{kj}x_j + b_k$. Without regulation of the output, this could lead to exploding outputs as we move through the layers of the network. Also, a non-linear activation function must be used in at least one layer of the network for the network to be able to learn anything more complicated than trivial relationships between input and output (with a linear activation function, the output is bound to be a linear combination of the inputs).
Common choices of activation function are the hyperbolic tangent or logistic function. Both have the property of being nonlinear around 0, giving the network the ability to model more complex relationships between inputs and outputs, and they are bounded so that the magnitude of the outputs remain within acceptable limits throughout the network.
Another popular activation function is the rectified linear unit (ReLU), which is just the identity function clamped at 0 for negative inputs: $f(x) = \max(0, x)$. It does

not have the same attractive property of boundedness, so care has to be taken when using it, but it has nonetheless been very successful within the fields of computer vision [11] and speech recognition [12]. The choice of activation function is critical for the performance of a network [13]. In the simplest case where a network is just made up of one neuron acting as both the input and output neuron, the network can easily be expressed as Equation 2.2

$$y = \varphi\left(x + b\right), \tag{2.2}$$

with y being the output, x the input and b the bias of the neuron. The neuron described in this section is what can be considered a "vanilla" type neuron, and when used in a neural network, a collection of this type of neuron makes up a type of layer often called a dense or fully connected layer.

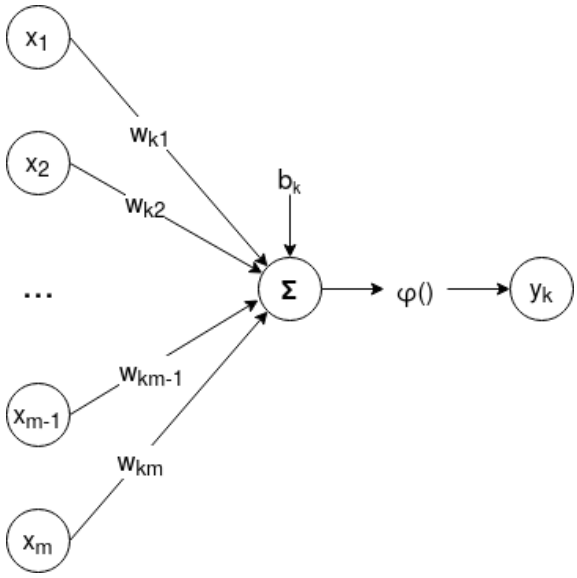


Figure 2.3: The operation performed by an artificial neuron k. x_j are the inputs from neurons in preceding layers, w_{kj} are the weights for the connections between input neurons and the neuron k, b_k is the neuron bias, $\varphi()$ is the activation function and y_k is the final output of neuron k.

2.1.2.2 Recurrent neural networks

When dealing with inputs and/or outputs that span several dimensions (spatial or temporal), specialized networks have been developed to deal with these situations. Recurrent neural networks (RNNs) are a subset of neural networks tailored for working with temporal sequences. These classes of networks contain recurrent layers, which can maintain internal states that are modified by the input, in contrast to non-recurrent neural networks where the internal state (the weights and biases of the neurons) is fixed. In a recurrent neural network, input data is provided to the network one time step at a time, and for each time step, the network produces an output and also updates its internal state based on the supplied input. The internal state is stored in the network across time steps. This gives the recurrent neural

network a form of memory, where the current output is not only dependent on the current input, but also on the current state of the network. In order to accomplish this, the artificial neurons must be replaced by units capable of storing an internal state and modify it based on the information passing through it. The most popular and widely used units for this purpose are LSTM and GRU (Gated recurrent unit) units.

2.1.2.2.1 LSTM units LSTM units (Figure 2.4) were conceived in 1997 [14], and has been shown to perform very well [15]. The units maintain a hidden internal state that can be modified by data passing through it, giving the units the ability to remember what has happened previously, and use that memory to influence its behaviour. The unit consists of a cell with several gates, namely the input ($\mathrm{I_t}$), forget ($\mathrm{F_t}$), and output ($\mathrm{O_t}$) gates. The flow in Figure 2.4 can be read from left to right. The merging of two lines represents a concatenation. σ represents a sigmoid activation function, tanh represents a hyperbolic tangent activation function, X represents a pointwise multiplication, and + represents pointwise addition.

First, the forget gate decides, based on the current value of the input (x_t), and the previous value of the output (h_{t-1}), what parts of the internal cell state (c_{t-1}) to "forget". Forgetting part of a state is not a binary operation. The forget gate will, based on the values of (x_t) and (h_{t-1}) passed through the sigmoid activation function, output a number between 0 and 1 for each member of (c_{t-1}). Each member of (c_{t-1}) is then multiplied with the value produced by the forget gate, so for example a forget gate value of 0.5 would mean that this particular member of the cell state would be halved. Next, the input gate in a very similar fashion determines what parts of the cell state gets updated with information from the current time step. Finally, the output gate determines what the actual output (h_t) of the cell should be for this time step. (c_t) and (h_t) are retained within the cell for use at time step t+1.

The trainable parameters of the LSTM unit are hidden away into the four activation functions used by the different gates. In the case of the forget gate $\mathrm{F_t}$, the actual mathematical operation performed by the sigmoid activation function is

$$\mathrm{F_t} = \sigma\left(w_F \cdot [h_{t-1}, x_t] + b_f\right). \tag{2.3}$$

Here, w_F is the vector of weights for the forget gate, $[h_{t-1}, x_t]$ is the concatenation of the current input and the previous output, and b_f is a bias for the gate. If we were to have a scalar input and output, this gate would have three weights, two members of w_F and the bias b_f. In total across the unit and its three gates, the unit would have 12 trainable parameters (the input gate has 6 parameters, since it uses both a sigmoid and a hyperbolic tangent activation function).

There also exists a very similar type of recurrent unit, called the gated recurrent unit (GRU) [16]. This unit works in a very similar fashion, but it uses gates called the update gate $\mathrm{Z_t}$ and reset gate $\mathrm{R_t}$ instead, and there is no hidden cell state, only the previous output gets passed on to the next time step. In this report, we will be using the LSTM unit for recurrent layers of the networks.

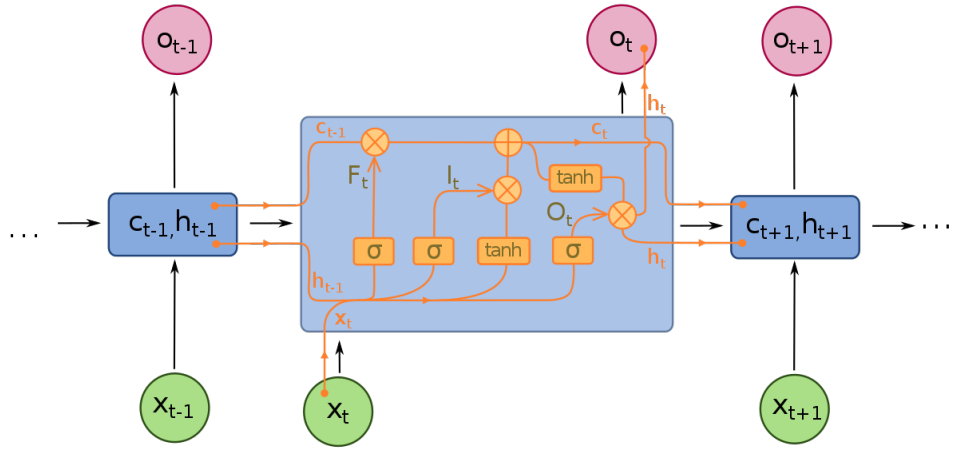


Figure 2.4: LSTM unit architecture [1]. The merging of two lines represents a concatenation. σ and tanh represents a sigmoid and hyperbolic tangent activation function respectively, X and + represents pointwise multiplication and addition.

2.1.2.2.2 Stateful recurrent neural networks Normally, a recurrent neural network resets its internal state (for LSTM units h_t and c_t) to its initial state (usually 0) every time the weights are updated. Since we will be training over a very long sequence of data, and wish to update the weights many times as we move through the training data, we don't want to reset the state after every weight update. We can do this by employing stateful recurrent layers in our network (applicable to any type of recurrent unit chosen to make up the recurrent layers). This however, means that we now have to manually control when the states are reset. Care has to be taken to reset the states manually, before the start of every training epoch.

2.1.2.3 Convolutional neural networks

When dealing with spatially distributed data such as images, there is a suitable class of neural networks called CNNs, Convolutional Neural Networks. A core component of these networks are the convolutional layers. These layers convolve a trainable kernel with the input to the layer, and outputs the result of that convolution, see figure 9.1 of [9] for an excellent visual explanation. The CNN can, by training, modify the weights in the kernel so that the output of the convolution is responsive to a feature in the input that is of interest to the network.

2.1.2.4 Hyperparameters

In the previous sections we have spoken about trainable parameters, which are optimized as part of the training of the network. Neural networks are also highly dependent on other parameters which need to be set before training begins. These parameters are called hyperparameters. Such parameters would for instance be how many LSTM units to use in a specific recurrent layer, how many fully connected layers to use, what activation function to use etc. Choosing these hyperparameters is an exploratory task, where a search is conducted across a space of hyperparameter settings largely guided by experience. The search can be performed either as a

grid search where all possible combinations of hyperparameters are exhaustively searched, a random search where the hyperparameter space is randomly sampled, or by a number of other techniques [17].

2.1.2.5 Training a neural network (backpropagation and gradient descent)

The performance of a neural network is only as good as its training. The performance of a neural network is evaluated using a loss function, usually cross entropy-based for classification problems, and a simple Mean Squared Error (MSE) loss function for regression problems. The most commonly used method for training a neural network is called backpropagation [18], or BP for short. First, all our training data is passed to our network (passing all our training data to the network once is called an epoch). BP then works by calculating the partial derivatives of the chosen loss function with respect to each weight in the network. This gradient information is then used by the selected optimization algorithm (usually a form of stochastic gradient descent) to update the weights in order to lower the loss function. The weights can be updated more often than once per epoch. If we feed a subsample of n instances of training data to our network at a time, then apply the BP and optimization algorithm and update our weights before feeding the next n samples to our network, we are said to be using a batch size of n. The batch size is another hyperparameter, and in general a network exhibits better performance when using a smaller batch size [19].
With this method comes an important hyperparameter, the learning rate. The learning rate hyperparameter dictates how much the gradient descent method modifies the weights in the direction of a lower loss function value. With a too high learning rate, the gradient descent methods would take large steps in the network weight space each time it updates, which would lead to problems converging on a good set of weights. A learning rate set too low would lead to the network not reaching its optimum performance in a reasonable time. These three scenarios for a very simple example with just a single weight are visualized in Figure 2.5. The learning rate does not need to be static throughout the course of the training. A common approach is to use an adaptive approach with respect to learning rate, starting out with a high learning rate, and lowering the learning rate whenever the network runs into a plateau in its training. This gives us the benefit of quickly moving to a part of the weight space where performance is "quite good", then lowering the learning rate in order to find the optimum weights within this part of the weight space.
BP also requires that the activation function used in the neurons of the network is differentiable.

2.1.2.5.1 Backpropagation through time When training an RNN using BP, we have to take into account the recurrent nature of the network. In order to apply backpropagation to an RNN, we first need to unroll the recurrent parts of the network. The concept is visualized in Figure 2.6. It is done by treating the recurrent connection from a unit as a connection to a copy of the downstream network (including itself) instead of back to itself. Weights of the downstream

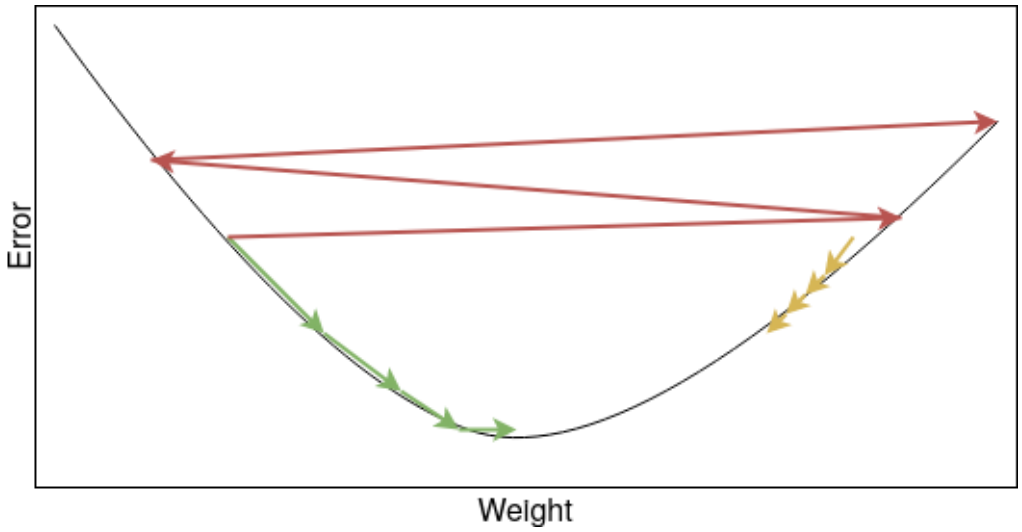


Figure 2.5: The effect of learning rate. Green: Just right, quickly approaches the minimum error. Red: Too high, the error tends to diverge. Yellow: too low, we do not make good progress towards the minimum.

network are shared across all time steps, since the downstream network is in fact one and the same network for all time steps. The number of copies that our original network is unrolled to during the backwards pass depends on how many time steps are being fed into the network in a single forward pass. With the network unrolled, the recurrent links no longer exist and normal BP can be applied.

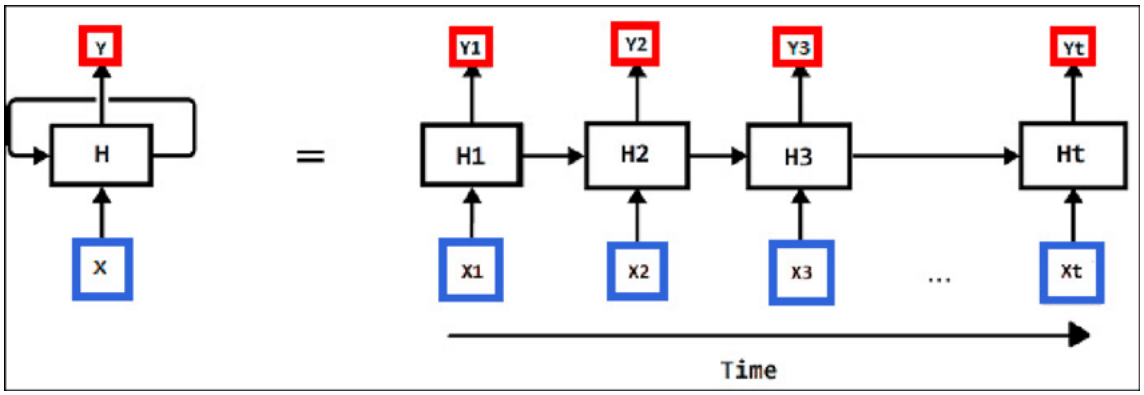


Figure 2.6: Unrolled recurrent neural network (figure from [20]). H on the left hand of the figure represents a single recurrent unit making up a part of a recurrent layer of a neural network, it could for instance be 1 of the 4 units in the hidden layer in Figure 2.2.

2.1.2.6 Overfitting

While a neural network can be trained to very accurately model the data it is provided during training, that performance is useless unless the network also performs well on new inputs not seen during training by the network, since we want to use the network in a predictive capacity. The phenomenon where a neural network performs

well on the data it trained on but not on new data is called overfitting. To try and mitigate this when dealing with these types of learning methods, we make use of the division of our data into training, validation and test sets. The training data set is the one used by the BP algorithm to iteratively update the weights of the network. After each epoch (one epoch being a complete pass through all the training data with the associated weight updates), the network is applied to the validation data set and the loss function is evaluated. The result of this application of the network to the validation data has indirect influence on the training process. It can for example determine when to lower the learning rate, or when to consider the training finished. Finally, once the network is fully trained, it is applied to the test data set, not used as either training or validation data. The result of the network on the test data set can be considered the true performance of the network.

2.1.2.7 Regularization techniques

Methods to combat overfitting are called regularization techniques. A basic and important regularization technique is to regularize our input and output data. If we have two scalar inputs, x_1 and x_2, with x_1 in the range of ± 1 and the other in the range of $0 < x_2 < 1000$, and we want to predict y which ranges from $10^4 < y < 10^7$, the weights of our network would have to do a lot of heavy lifting just to scale the data correctly. By just standardizing or normalizing our data, the network can focus on finding the relationship between inputs and outputs. These scalings can easily be reversed outside of the neural network if the absolute numerical value of the output is of importance (such as in regression tasks).
Common methods are L1/L2 regularization, dropout and early stopping. They all modify different aspects of the training of the network to make it more resistant to overfitting.
L1/L2 regularization works by modifying the loss function by adding a regularization term, incurring an extra penalty for large weights in the connections between neurons. For L1 regularization, the term added is

$$\lambda \sum_k |w_k| \tag{2.4}$$

where λ is the regularization factor, and w_k is the weight of connection k. For L2 regularization, the term is

$$\lambda \sum_k w_k^2. \tag{2.5}$$

Dropout works by, during training of the network, randomly setting a fraction x of the outputs from a layer to 0. The outputs from nodes in that layer that are not set to 0, are instead scaled up by a factor $\frac{1}{1-x}$ in order to keep the sum of outputs unaffected by the applied dropout. This technique discourages the formation of complex co-adaptations within the network, where some nodes might start to compensate for errors introduced by nodes in previous layers, which would not work on previously unseen data [21].
Early stopping is more a time saver than a regularization technique, and works by monitoring the validation loss during training. Once the validation loss no longer decreases (or starts to increase again), we are starting to see overfitting, and early

stopping then steps in and ends the training. With this method, yet another hyperparameter comes into play: patience. The validation loss is inherently noisy, it does not monotonically decrease with each training epoch. If we were to stop training at the first sign of increased validation loss, we would get nowhere. The patience hyperparameter determines how many epochs we are willing to wait and see if the long-term trend of the validation loss is actually decreasing or increasing. A low patience means we risk stopping too early, a high patience means that we migh not stop until some overfitting has already developed. Early stopping can be a great time saver when exploring hyperparameters, since you don't have to run the training for the full number of training epochs you specify in your search, but only run the training as long as you are seeing an improvement in the validation loss. Usually, network with poorly chosen hyperparameters will see the validation loss stop decreasing very quickly, and you can then end training.

2.1.2.8 Summary

To summarize, a neural network is a construct made up of interconnected layers. Each layer is either made up by a number of units (which could be normal artifical neurons, LSTM units etc.), or the layer performs some other operation on the data (such as a convolutional layer). If a layer is made up of n units, that layer is said to be of width or size n.
The network is governed by a variety of hyperparameters, determining not only the overall architecture and size of individual components, but also governing many aspects of the learning process. A search across hyperparameters is a necessary part of developing a well functioning neural network.
The network is repeatedly fed all the available data in the training set (each repetition is called an epoch), and the weights of the network components are updated using the chosen optimization algorithm periodically. The number of samples fed to the network between weight updates is called the batch size.
After each epoch, the network in its current state of training is applied to the data in the validation set. The performance of the network on the validation set can be used to guide decisions on the training process. Should the learning rate be modified? Are we starting to see overfit and should end training?
Once we determine that training is completed, we apply the final network to the data in the test set to get a final unbiased judgement on the performance of the network.

2.2 Superconducting gravimeter

The superconducting gravimeter (SG) is a sensitive, low-noise and stable instrument, compared to previous mechanical systems. It is capable of measuring variations in gravity on a very small scale, reaching an accuracy of about 1 nms^{-2}. Compare this to the average value of 9.81 ms^{-2}. The performance of the SG has enabled studies of tidal, atmospheric and hydrological effects on gravity with previously unachievable precision [22]. Changes in gravity can also be used as a method of detecting magma movements, allowing early detection of possible future volcano

eruptions [23]. One important note to make is that the SG is a relative instrument, so it can not measure the absolute strength of gravity, only changes to it. The availability of these measurements enables a wealth of studies to be performed, since the SG detects any movement of mass, from the inner core of the Earth to the atmosphere, and the motion of celestial bodies.

2.2.1 Principles of operation

The superconducting gravimeter at the Onsala Space Observatory is an "Observatory SG" produced by GWR Instruments, Inc. For an overview of the core components of the SG, see Figure 2.7. For a deeper introduction to the design of the instruments, see [24]. The interior of the machine uses liquid Helium to operate at cryogenic temperatures (9.2 K). A magnetic field is produced by induced currents in a pair of superconducting coils (upper and lower coil in Figure 2.7). The magnetic field induces currents on the surface of the superconducting Niobium sphere in the center of Figure 2.7, providing a levitation force for the sphere. The fact that both the coils and sphere are superconducting means that the currents will remain stable, and therefore also provide a stable levitation of the sphere. When the force of gravity changes, so does the position of the sphere. The magnetic fields in the SG are tuned so that a small change in gravity gives a large displacement of the sphere, providing a high degree of sensitivity. The displacement of the sphere is sensed by a capacitive sensor (the plates surrounding the sphere), and a current is applied to the feedback coil in such a way as to maintain the original position of the sphere. The amount of current flowing in the feedback coil is proportional to the change in gravity, and requires an absolute gravimeter in order to determine the constant of proportionality. Once calibrated, the amount of current in the feedback coil can be used to determine the change in gravity.

2.2.2 Precision

The SG achieves an precision approaching 1 nGal (10^{-11} ms^{-2}) over periods ranging from seconds to years [25].

2.3 Sources of gravity variations

The SG registers the cumulative effect that all physical phenomena has on gravity. In order to study a specific phenomena, all other phenomena has to be removed from the cumulative signal first. These phenomena vary greatly in frequency, amplitude and periodicity as seen in Figure 2.8.

2.3.1 Tides

Any heavenly body changing its relative position with respect to the Earth will change the strength and direction of the local gravity field on Earth. The most significant astronomical sources are the Moon and the Sun. They are, in combination with the rotation of the earth, responsible for tides (both oceanic and solid). Tides

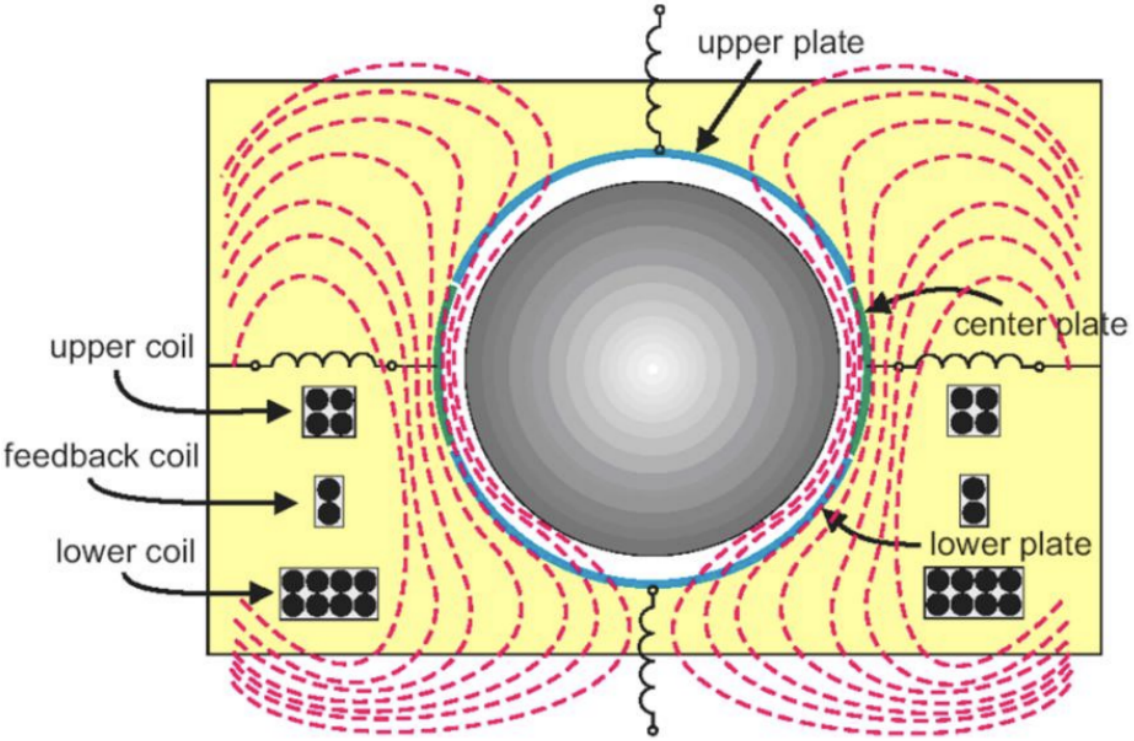


Figure 2.7: Conceptual drawing of flux lines, plates and Nb sphere (figure and caption from [25]).

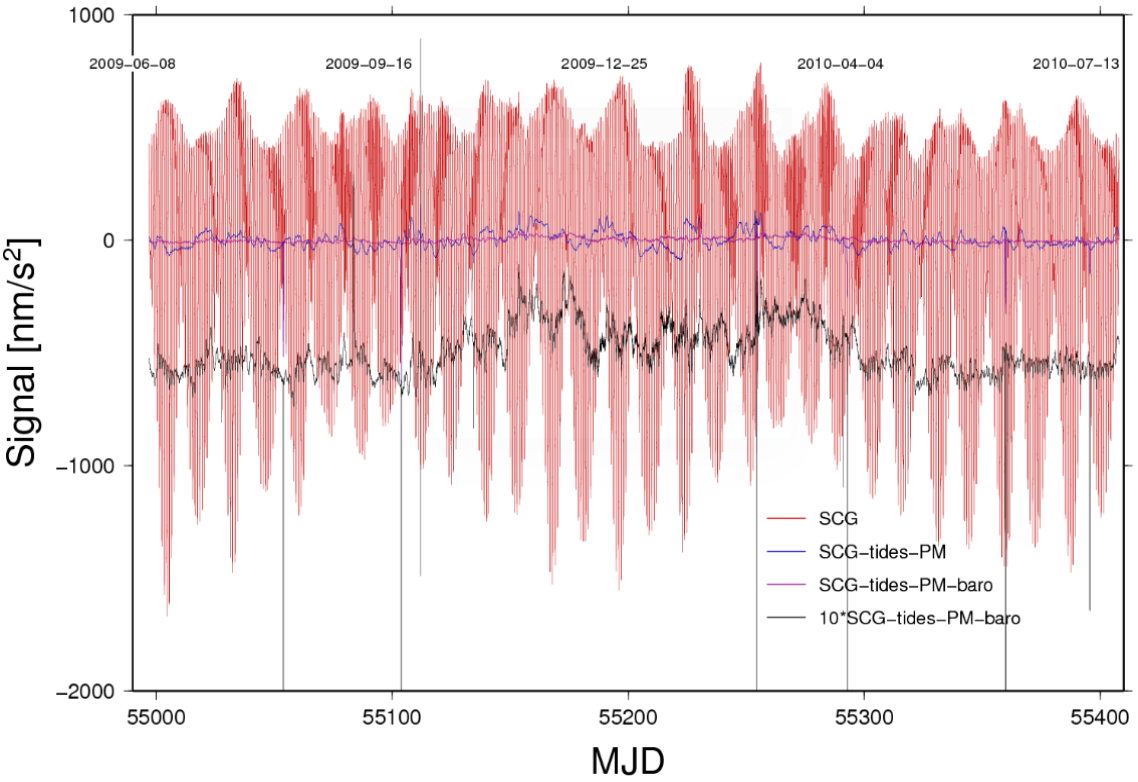


Figure 2.8: Raw and processed measurements from the first year of operation of the Onsala Space Observatory SG [26].

in general cause deformation and mass redistribution, which in turn changes the local gravity field strength.

2.3.2 Polar motion

Polar motion, the next significant contributor to the signal, is caused by the Earth's axis of rotation moving relative to the crust of the Earth. The strength of these variations are on the order of 15 μGal, or about 150 nms^{-2} [27]. Since we are seeing variations on the order of $\pm$1000 nms^{-2} in the raw unprocessed signal, we can assume that the majority of the signal can be attributed to tidal effects, with the next component in order of amplitude is the polar motion.

2.3.3 Seismic events

The SG is sensitive enough to pick up seismic events from across the Earth. An example of this is shown in Figure 3.1. The example in Figure 3.1 is a result of the vibration of the ground on which the SG is standing, but the SG can also detect the change in gravity from the redistribution of mass, a change which propagates at the speed of light. Such detections, among other things, give further insights into the makeup of the interior of the Earth [28], and could aid in faster determination of earthquake magnitudes [29].

2.3.4 Meteorological effects

Looking again at Figure 2.8, the signal corrected for tides and polar motion is most strongly affected by barometric pressure (denoted "baro" in the plot). The effect is linear, and usually between 2 and 4 nms^{-2}hPa^{-1} [30].

2.3.5 Hydrological effects

The presence of water in proximity of the SG can have a noticeable influence on the measured strength of gravity. A theoretical infinite layer of water 10 millimeters thick would produce an increase in gravity by 0.4 μGal, or 4 nms^{-2}. Increase in gravity has been observed in close correlation with rainfalls at this ratio [22].
Rainfall either evaporates back into the atmosphere through a process called evapotranspiration, or it infiltrates the ground, or if the ground is already saturated or the water is moving to quick to infiltrate the ground, it will flow across the ground until it is able to infiltrate the ground or join a watercourse (this process is called runoff).
The level in the ground where the soil or fissures become completely saturated with water is called the water table. The level of the water table can quite easily be measured through a drilled well, but building a well is complex and expensive. Also, in locations with complex underground conditions, it is not a given that a single water table measurement is representative of the total mass of water present in the vicinity of the well [22]. Therefore, to more accurately model the effect, knowledge of the spatial distribution of groundwater would be required [4]. Gaining that knowledge is a very complicated task, requiring detailed knowledge of underground conditions

[31]. Nevertheless, it is important to quantify the effect of groundwater on gravity, and not only for hydrological purposes. If we want to investigate any other effect on gravity, the hydrological effects must first be identified and removed ([5], [32]). Is this also something that machine learning has the potential to help us with?
As detailed in the next section, for the period of 2009-2015, we have a gravitational and meteorological record but no groundwater record. If we are able to reconstruct the groundwater level for 2009-2015 from meteorological records, we are more than doubling the amount of data on which to model the local hydrological effects on gravity at the Onsala Space Observatory.

3
Methods

3.1 Available data sets

The Onsala Space Observatory operates a variety of sensors. The data sets available in this project are listed in Table 3.1.

Table 3.1: Available data sets

Data set	First date	Last date	Samples	Sampling rate
Gravimeter	2009-06-15	2020-01-13	333936000	1 second
Groundwater level	2015-08-31	2019-12-31	2280151	1 minute
Meteorological data	2009-08-20	2019-12-31	5332407	Approx. 1 minute

The data sets consist of one or more measurements each, listed in Table 3.2.

Table 3.2: Measurements in each data set

Data set	Measurement	Unit
Gravimeter	Relative gravity	nms^{-2}
Groundwater level	Groundwater level	m
Meteorological data	Temperature	°C
	Relative humidity	RH%
	Wind speed	ms^{-1}
	Atmospheric pressure	hPa
	Rain	mm

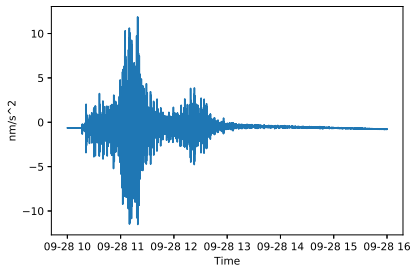
3.2 Data preprocessing

3.2.1 Cleaning up the gravity signal

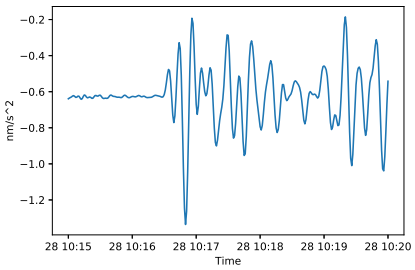
Since the raw signal from the SG is the cumulative sum of all sources of gravity, we need, to the best of our ability, remove the components of the signal that is not the target of this study.

3.2.1.1 Seismic events

Since the SG is such a sensitive device, and samples at such a high rate, it readily detects seismic events which completely overwhelms the signal we are interested in. An example is shown in Figure 3.1, where the SG detected the 2018 Sulawesi earthquake [33]. The effect is the acceleration of the ground transmitted through the earth, not a change in gravity (although direct detection of gravity changes due to earthquakes has been accomplished elsewhere [34]). Looking at Figure 3.1b, we can see that the SG detects it as a strong oscillation. The seismic events were removed from our signal by passing it through a low-pass filter with a cut-off period of 240 seconds.



(a) Raw SG data between 10:00 and 16:00, 2018-09-28



(b) Raw SG data between 10:15 and 10:20, 2018-09-28

Figure 3.1: Raw SG data detecting the 2018 Sulawesi earthquake [33].

3.2.1.2 Tides

Tidal effects (both solid earth and oceanic tides) in the gravity measurements were modeled by tidal analysis using Baytap08 [35] and Tsoft [36]. The resulting tidal component of our gravity record was then removed.

3.2.1.3 Non-tidal ocean loading and atmospheric pressure loading effects

Non-tidal ocean loading effects were removed using the T-UGOm model [37]. This correction was only available for the dates 2010-01-01 through 2018-10-31. The earth surface displacement due to atmospheric loading was removed using the ERA5 model [38].

3.2.1.4 Polar motion

Polar motion effects was modeled using Tsoft and EOP parameters available for download from [39] and removed from the gravity signal.

3.2.1.5 Trend and other outliers

At this point, the data still contain some long-term (decadal) trends, and some outliers. The trend comes from instrument drift and very slow geological processes,

such as land rise due to glacial isostatic adjustment (GIA) [40], which is active in Sweden. The trend was removed by removing the slope of a linear fit from the data. The outliers were mostly removed algorithmically, and a few by visual inspection.

3.2.2 Filling gaps in data

No data set is perfect. The data has gaps due to equipment maintenance, failures etc. These gaps need to be managed in some manner. Since we are working with time series data, data points are not independent of each other, so we cannot simply remove data points that are not complete (all input features and output feature present at that time step), we need a continuous evenly sampled data set. In the data sets available, the gaps present are listed in Table 3.3. A big gap is defined as a gap longer than 10 minutes. All gaps were filled using a linear interpolation method, with respect taken to the time stamp of the missing data points, instead of assuming a linear spacing in time between subsequent rows in the data.The software used for this step (and in many other places throughout the project) was Pandas, a data analysis and manipulation tool built on top of the Python programming language [41].

Table 3.3: Gaps in the data sets.

Data set	Number of gaps	Big gaps	Longest gap
Gravimeter	551376	491	9 hours, 5 minutes
Groundwater level	21	6	25 days, 10 hours, 43 minutes
Meteorological data	1493	33	2 days, 6 hours, 18 minutes
Mareograph	2576	13	16 hours, 1 minute

3.2.3 Resampling/downsampling

The recurrent layers that will be used in our neural networks require data points to be evenly spaced in time. Our different data sets are sampled at different rates, and at different times. In order to get coherent timestamps, and a manageable sized data set for computations, all data was resampled and downsampled with homogeneous timestamps at a 1 hour sampling rate by linearly interpolating each measurement to the nearest even minute, then taking the mean of the measurements within each hour (except for rainfall when the rain during the last hour is already readily available in the data set). This downsampling also introduces an averaging effect which will impact the performance of our neural networks. Some measurements, such as the wind speed for instance, will vary greatly over the course of an hour. These variations, which could have an effect on evapotranspiration and therefore the infiltration of surface water into the water table, are unfortunately something that we will lose by downsampling. The averaging will also reduce any inherent noise in the data.

3.2.4 Final preprocessed data

After applying all the preprocessing steps, the final data set is visualized in Figure 3.2. Groundwater levels exhibit sharp rises after a rainfall, and then exponential decay. There is also a seasonal trend. Residual gravity, humidity and temperature exhibit clear seasonal trends. Wind speed is fairly consistent across the dataset. Atmoshperic pressure has a lower variance during the summers.

3.3 Neural network

3.3.1 Architecture

Machine learning models do not yet base their performance in physical equations, and there are no ways of analytically determine what type of network will perform the best for a given problem. We will explore reasonable candidate architectures within two major classes; sequential and non-sequential architectures. They are visualized in Figure 3.3. They both consist of a number of LSTM layers followed by a number of densely connected layers, but differ in how the layers are connected. In the sequential case (Figure 3.3a), each layer feeds the next in a sequential fashion. In the non-sequential case (Figure 3.3b), each LSTM layer feeds into both the next LSTM layer, and the first densely connected layer through a concatenation layer. This means that the first dense layer in the non-sequential model has access to $N \times k$ different LSTM interpretations of the input data, with N being the number of LSTM layers and k being the number of LSTM units in a layer. For comparison, in the sequential model, the first densely connected layer only has access to the outputs of the k units of the last LSTM layer. The assumption we seek to test is that the increased number of inputs to the densely connected part of the network in the non-sequential architecture will enable the network to make better predictions, without overfitting to the training data.
All architectures will be explored as stateful LSTM networks, and fed the input data 1 time step at a time. This means that the unrolling of the recurrent parts of the network during the backwards pass of the BP algorithm is technically not needed, since it is the number of time steps that determines how many copies of the recurrent parts of the network emerge from the unrolling. The resulting unrolled network will look just like the network did before the unrolling. In Figure 3.3, dropout layers are not shown for brevity. In reality, dropout layers are always in place between all dense layers, and in the sequential case after each LSTM layer. In the non-sequential case, a dropout layer is in place between each LSTM layer and the concatenation layer, but not in-between the LSTM layer.
We will also explore the effects of giving the network the ability to preprocess the inputs, specifically using 1D convolutional layers in front of the LSTM layers.

3.3.2 Hyperparameter tuning

Hyperparameter selection is of paramount importance to a well functioning neural network. There are no fixed rules on what constitutes a good network architecture

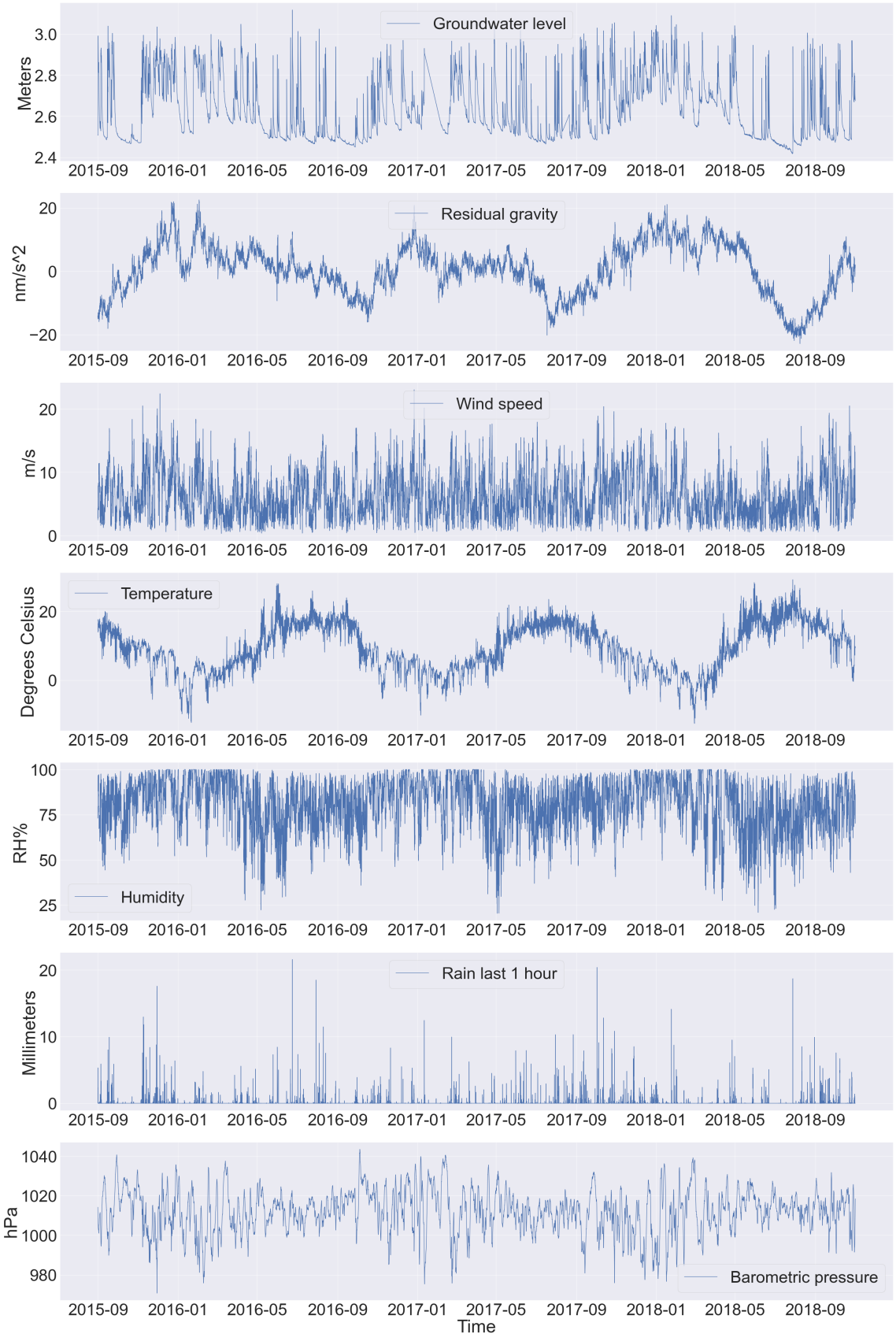
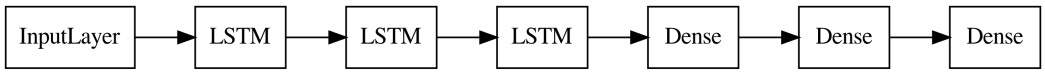
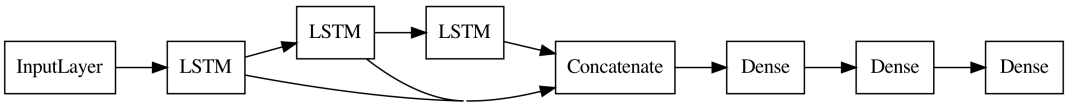


Figure 3.2: Preprocessed data used (after scaling) as training and validation sets for groundwater predictions, and as training, validation and test sets for gravity predictions.



(a) Sequential architecture



(b) Non-sequential architecture

Figure 3.3: Examples of the two classes of architectures explored, both examples using 3 LSTM layers (dropout layers not shown for brevity).

for a given problem, so usually empirical trial-and-error experiments are combined with a search across parameter space in order to find architectures (and associated parameters) that work well with the given problem. In this report we will be using a grid search in order to search a reasonably sized parameter space for good candidates. A trade-off has to be made between the scope of the grid search and computational time required. Even though we have access to an HPC cluster, it still takes a considerable amount of time to conduct a search.

3.3.3 Data set organisation

3.3.3.1 Groundwater predictions

For groundwater predictions, data from 2015-08-31 until 2018-03-14 was used as the training set, data from 2018-03-14 until 2018-10-31 was used as the validation set and data from 2019-01-01 until 2019-12-31 was used as the test set. The gap of 2 months between the end of the validation set and the start of the test set is deliberate, in order to not give the model the benefit of a proper starting state when using the test set. The division between training/validation/test data is 61/15/24 %. The training and validation sets are shown in Figure 3.2.

3.3.3.2 Gravity predictions

For gravity predictions, data from 2015-08-31 until 2017-11-17 was used as the training set, data from 2017-11-18 until 2018-05-10 was used as the validation set and data from 2018-05-11 until 2018-10-31 was used as the test set. The division between training/validation/test data is 70/15/15 %. The start date 2015-08-31 is dictated by the fact that this is the start of the groundwater well measurements. The end date 2018-10-31 is dictated by the availability of the T-UGOm corrections [37]. The training, validation and test sets are shown in Figure 3.2.

3.3.4 Computational considerations

The neural networks were realized using Tensorflow/Keras [42], and the grid search and training of candidate models were performed on an HPC cluster. Due to the fact that we are using stateful recurrent layers in our networks, it is difficult to

parallelize the training of a single model, since the model needs to be fed the input data in a sequential fashion.
In order to parallelize as much as possible, when performing a grid search, the grid search was spread out over a number of nodes splitting the grid search on a single hyperparameter (most often the number of LSTM layers in the model). By doing this, we were able to have a handful of computing nodes share the computational load of a single grid search.
Furthermore, to use the number of available cores on each node as efficiently as possible, we used a process pool sized to the number of available cores on each node when training the models. This means that on a node with 64 cores, we were training 64 different models simultaneously.

4
Results

Machine learning is still an area guided by trial-and-error and experimentation, for that reason this results section not only presents the best results achieved, but also spends a significant amount of text and figures going through the variation of performance across choices of architecture, hyperparameters, and input data.
Consistently throughout the results section of this report, the performance metric is the mean squared error (MSE) between the observed and predicted values of the target physical quantity, and lower MSE equals better performance. In the running text and final results tables (tables 4.8 and 4.15), the MSE is the unscaled MSE with units m^2 for groundwater, nm^2s^{-4} for gravity. For performance distributions as a function of hyperparameter settings , the MSE is that of the scaled data that was fed to the networks. When scaled MSE is used, it is clearly denoted on the figure axes. Performance distributions are visualized using box plots, with the box representing the interquartile range (IQR), and the whiskers extending 1.5 times the IQR past the low and high quartiles.
The best performing models are selected based on their performance on the validation set, and when a "best performance on a test set" is mentioned, the performance is that of the model that was selected based on its validation set performance, meaning that there might be models that performed even better on the test set.

4.1 Groundwater predictions

Groundwater levels are predicted using several different architectures and inputs from Figure 3.2. Architectures used are both the sequential and non-sequential LSTM layouts discussed in Section 3.3.1. We will gauge the impact of using convolutional layer preprocessing of the inputs, and also using not only the current values as input, but provide the models with a 24 hour history of all inputs. We will also remove one input at a time to try and gauge their relative importance to the performance of the networks. The Pearson correlation coefficient between input quantities and target quantity are listed in Table 4.1. Rainfall looks quite poorly correlated with the groundwater level, but this is misleading. The rainfall measurement is quite discontinuous, we can have a massive downpour one hour, and zero rainfall the next. Also, the groundwater level rises as long as rain falls. Once the rain stops, the rainfall input variable goes down to zero just as it was before the rainfall, while the groundwater level remains at a higher level than it was before the rainfall. Correlation does not take into account this type of relationship. Hyperparameters explored for the different types of networks can be found in plots for each

Table 4.1: Input quantity Pearson correlation coefficients with groundwater levels across the training and validation sets.

Input quantity	Correlation with groundwater level
Wind speed	0.236
Temperature	−0.429
Relative humidity	0.401
Rain last 1 hour	0.164
Air pressure	−0.434

experiment in the Appendix. The case with 1 LSTM layer leads to the same architecture in the sequential and non-sequential architectures, so that specific choice of hyperparameter was only explored as part of the sequential architectures. As the experiments progressed, hyperparameter space was narrowed somewhat in order to reduce the computational time needed.

4.1.1 Sequential model using only rainfall data

The input and target variable are visualized in Figure 3.2, "Rain last 1 hour" as the input and "Groundwater level" as the output. The best MSE achieved on the validation data is 0.0129 m^2, visualized in Figure 4.1 using the hyperparameters in Table 4.2. The performance of the network across the hyperparameter space is visualized in Figure A.1. The hyperparameter most influential on the performance on the model seems to be dropout rate where a clear trend can be seen in the performance. Also impactful is the LSTM layer width, where layers wider than 5 units show a trend of worsening results. The size of the dense layers seems to have a minor impact, although a slightly worse performance on the validation set can be seen with increasing layer sizes. This could be an indication of overfitting and that the dropout layers are not as effective in suppressing it as the layers grow in size. The model correctly identifies the sharp rise and exponential decay after a rainfall, but since no other data is available, the long-term seasonal pattern seen in the groundwater levels is not captured by this model.

Table 4.2: Hyperparameters for the best performing model using only rainfall as input.

Hyperparameter	Value
Number of LSTM layers	3
LSTM layer width	5
Dropout rate	0.2
First dense layer width	10
Second dense layer width	5
Batch size	1
Learning rate	10^{-5}

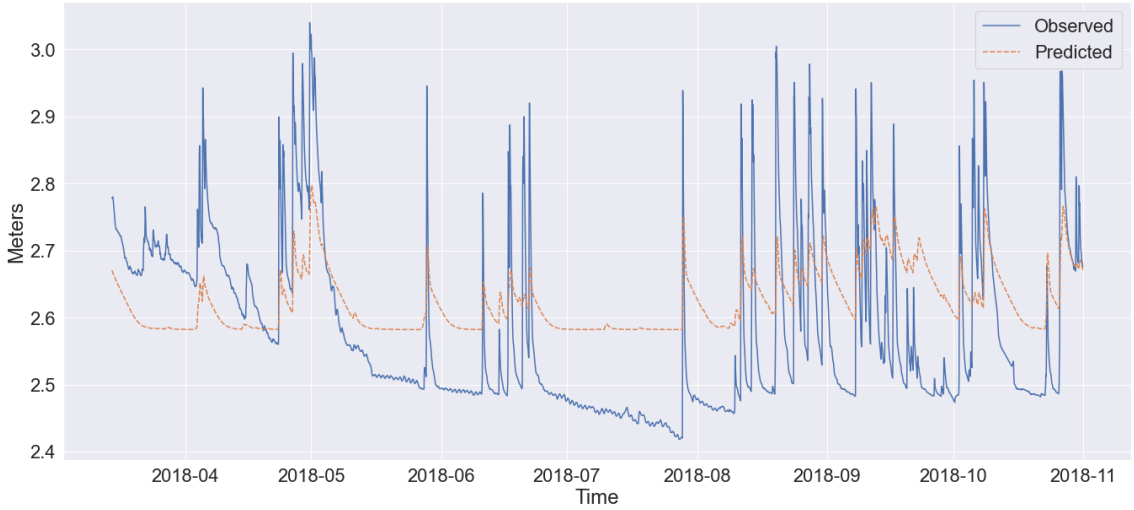


Figure 4.1: Best fit on validation set using only rainfall as input. MSE 0.0129 m^2.

4.1.2 Sequential model using all available meteorological data

The input and target variable are visualized in Figure 3.2, "Groundwater level" as the output and all remaining variables except "Residual gravity" as input, a total of 5 input features. The performance of the network across the hyperparameter space is visualized in Figure A.2. The risk of overfitting increases drastically with more than 3 LSTM layers, while those networks also show the best results. This could indicate that we have a quite complex situation for the optimization algorithm, where it is easy to converge to some local minima of the loss function that causes the model to not generalize well outside the training set. The best MSE achieved on the validation set is 0.0029 m^2 (visualized in Figure 4.2), with the same model achieving an MSE of 0.0039 m^2 on the test set, using the hyperparameters in Table 4.3. This model far outperforms the rainfall only model in Figure 4.1. There is quite a bit of noise in the predicted values during periods of little or no rain.

Table 4.3: Hyperparameters for the best performing sequential model using all available meteorological data.

Hyperparameter	Value
Number of LSTM layers	2
LSTM layer width	10
Dropout rate	0.0
First dense layer width	20
Second dense layer width	5
Batch size	1
Learning rate	10^{-5}

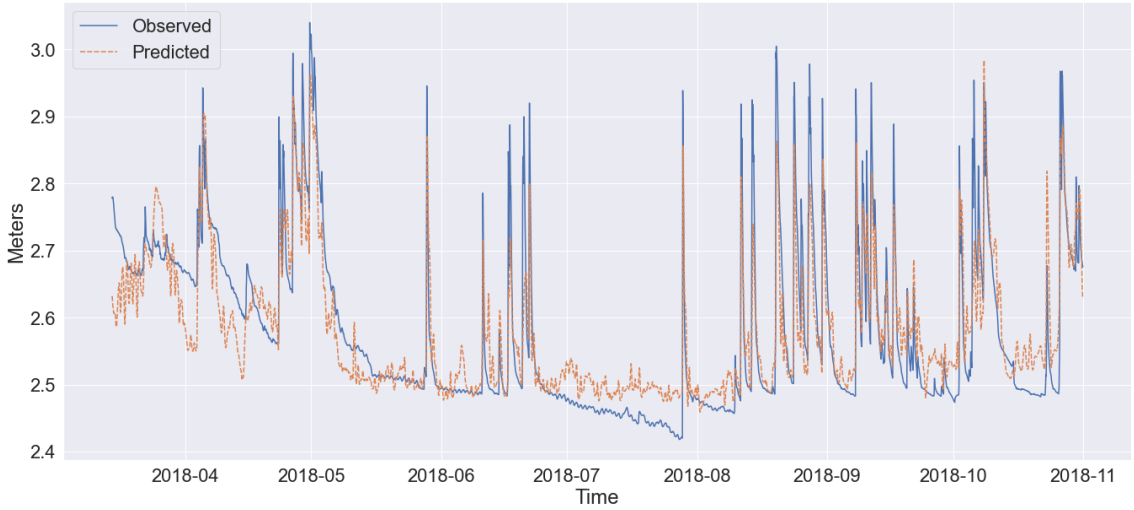


Figure 4.2: Best fit on validation set, sequential model using all available meteorological data. MSE 0.0029 m^2.

4.1.3 Non-sequential model using all available meteorological data

The same inputs are used as in the sequential case, all 5 meteorological features in Figure 3.2 are used to predict the groundwater level, but now using the non-sequential architecture in Figure 3.3b. Also, the explored hyperparameter space was altered a bit in order to accommodate the increased number of inputs to the first dense layer. The two explored parameters for the width of the first dense layers are now 20 and 40 units (compared to 10 and 20 for the sequential cases). The second dense layer hyperparameter space was also modified, exploring 5, 10 and 20 units instead of only 5 and 10 in the sequential case. The performance of the network across the hyperparameter space is visualized in Figure A.3. The most influential hyperparameter appears to be the LSTM layer width, and it is similar to the sequential case. An increasing number of units per layer increases the computational capacity of the model, allowing it to theoretically produce a better fit, with the downside that overfitting becomes more and more likely as the computational capacity grows. The best MSE achieved on the validation set is 0.0027 m^2 , visualized in Figure 4.3 using the hyperparameters in Table 4.4. This model achieved the same MSE, 0.0027 m^2 on the test set, indicating we are not suffering from any major overfitting. This model performs slightly better than the sequential model using the same input features.

4.1.4 Non-sequential model with CNN preprocessing using all available meteorological data

The exponential decay of groundwater levels is a slow process compared to the changes in the meteorological quantities we use to predict the groundwater level. The models using all available data has so far exhibited a noisy predicted value during periods when the observed value is just slowly decaying (see figures 4.2 and

Table 4.4: Hyperparameters for the best performing non-sequential model using all available meteorological data.

Hyperparameter	Value
Number of LSTM layers	4
LSTM layer width	100
Dropout rate	0.2
First dense layer width	20
Second dense layer width	5
Batch size	1
Learning rate	10^{-5}

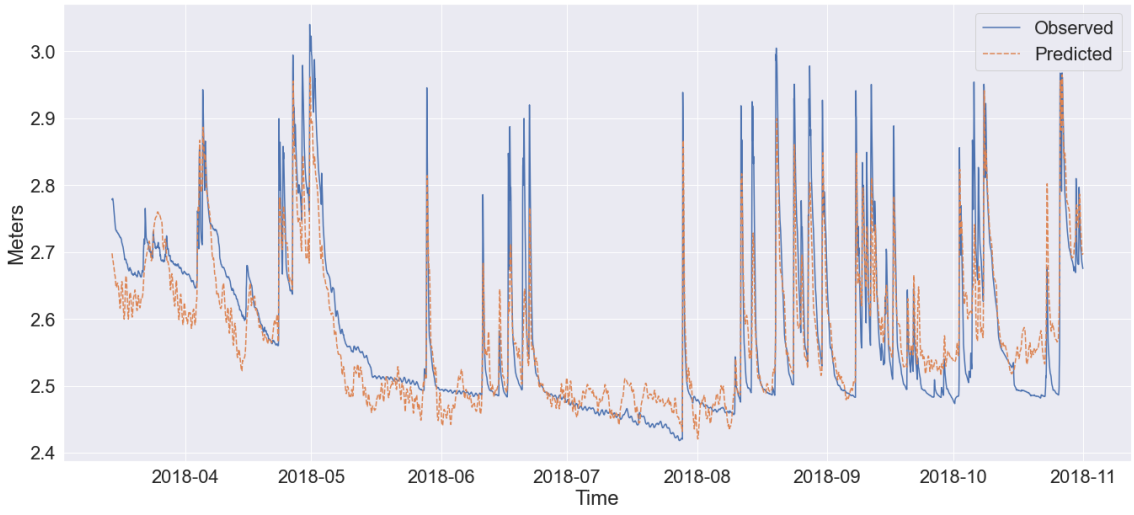


Figure 4.3: Best fit on validation set, non-sequential model using all available meteorological data. MSE 0.0027 m^2.

4.3, during the period 2018-07 - 2018-08). One tentative way to help the models deal with this situation is to add a convolutional preprocessing layer to the network before feeding the input data to the LSTM stage of the network. This gives the network the ability to use a convolution over a set period of time as its input instead of just using the current values of the input. Two experiments were performed. In both cases, 24 data points (representing 24 hours of data) were fed into independent 1D convolutional layers, 1 layer per input variable. The convolutions were configured to use a kernel width of 24 as well, meaning that the outputs of the convolutional layers will be a single scalar per input variable. These convolutional layers gives the network the ability to learn what combination of quantities from the past 24 hours is best suited to be used as in input to the LSTM stage of the network. In the first experiment, no constraints were placed on the weights in the convolution. This model did not lead to any performance increase, the best result was an MSE of 0.0096 m^2 on the validation set and 0.0137 m^2 on the test set. The model lost the ability to model the sharp rises in groundwater level following rainfalls, but maintained the ability to model the long-term changes in groundwater levels.

In the second experiment, a constraint of non-negativity was placed on the weights of the convolutional part of the model. This means that none of the values of the past 24 hours can be negatively weighted into the single value passed along the the LSTM stage of the model. This change did not however increase performance, the best model yielded an MSE of 0.0095 m^2 on the validation set and 0.0148 m^2 on the test set. The model did not regain any capability to model the sharp rises in groundwater (Figure 4.4).

The averaging effect imposed by these convolutional layers most likely negatively impacts the performance of the LSTM stage during periods of fast changes in groundwater levels. The convolution used is static, so the same averaging over 24 hours will be used regardless of the current dynamics, which probably leads to a very hard optimization problem. Hyperparameters for the best performing model using the non-negative constraint can be found in Table 4.5, a full overview of performance across hyperparameters can be found in Figure A.4.

Table 4.5: Hyperparameters for the best performing non-sequential model with CNN preprocessing (constrained to non-negative values) using all available meteorological data.

Hyperparameter	Value
Number of LSTM layers	3
LSTM layer width	10
Dropout rate	0.1
First dense layer width	40
Second dense layer width	10
Batch size	1
Learning rate	10^{-5}

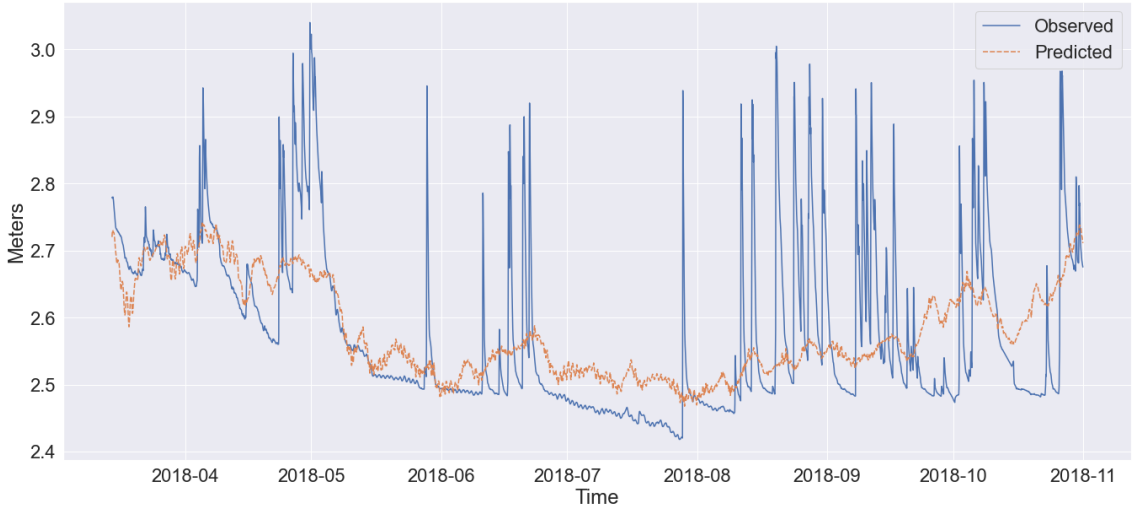


Figure 4.4: Best fit on validation set, non-sequential model with non-negative CNN preprocessing of the inputs using all available meteorological data. MSE 0.0095 m^2.

4.1.5 Non-sequential model with CNN preprocessing and current values

The models using CNN preprocessing did not perform particularly well, but by looking at figure 4.4 during the period 2018-07 - 2018-08, it looks like those models were somewhat better equipped to handle the periods of little to no rainfall (compare figures 4.3 and 4.4 during the period 2018-07 - 2018-08). Here, we combine the previous models by providing a non-sequential LSTM model with both the current values of the inputs and a CNN convolution over the past 24 hours for each input. The CNN was once again constrained to use non-negative weights. This approach seems to work better than either method did on its own (figures 4.3, 4.4), with the best model achieving an MSE of 0.0024 m^2 on the validation set (Figure 4.5) and 0.0021 m^2 on the test set (Figure 4.6). This is the best performance on the test set in this report. Hyperparameters for the best performing model are listed in Table 4.6 and the model is visualized in Figure 4.7. Performance across the explored hyperparameter space is visualized in Figure A.5.

Table 4.6: Hyperparameters for the best performing non-sequential model with CNN preprocessing and current values using all available meteorological data.

Hyperparameter	Value
Number of LSTM layers	4
LSTM layer width	50
Dropout rate	0.2
First dense layer width	20
Second dense layer width	5
Batch size	1
Learning rate	10^{-5}

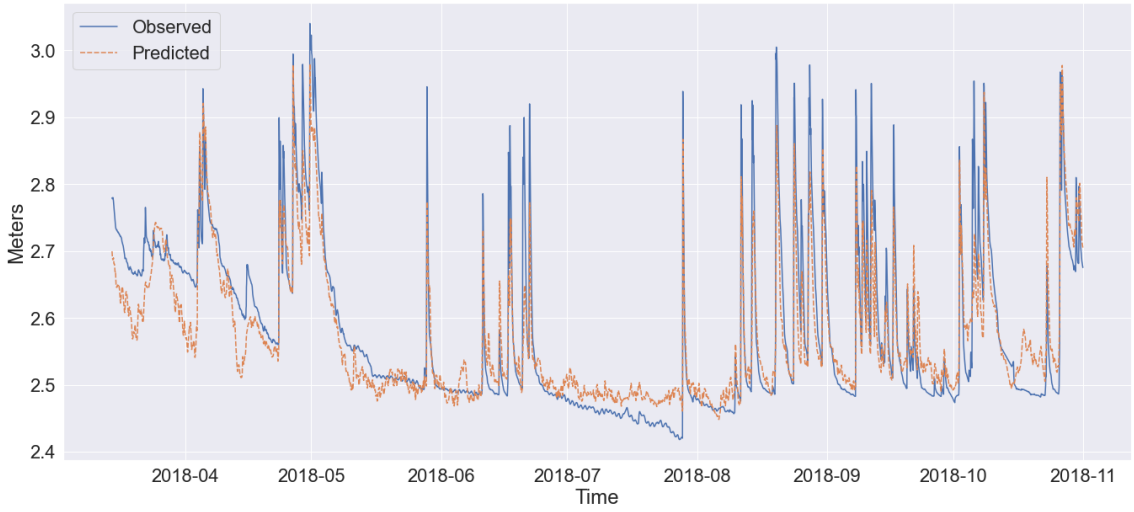


Figure 4.5: Best fit on validation set, non-sequential model with non-negative CNN preprocessed and current values using all available meteorological data. MSE 0.0024 m^2.

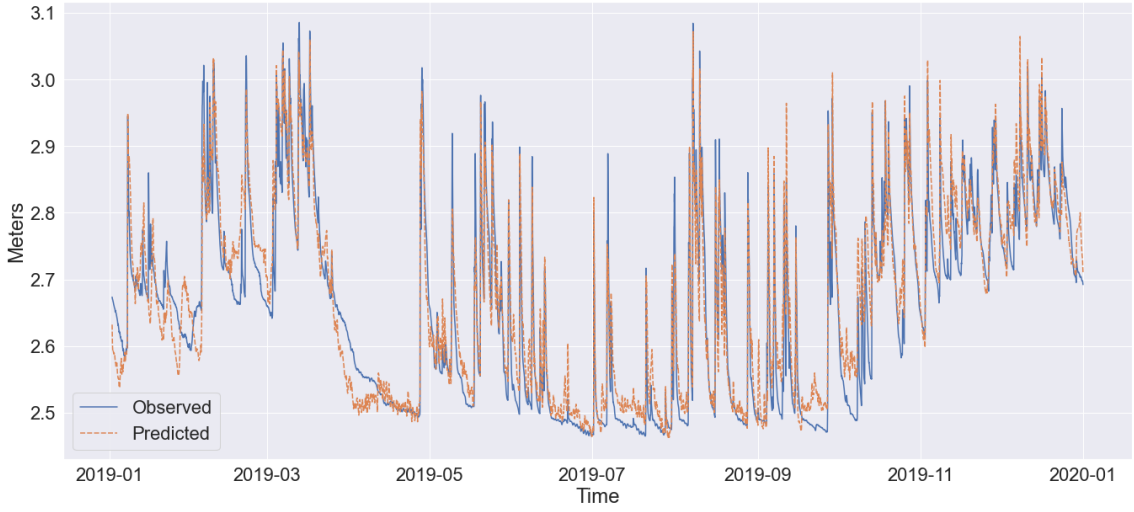


Figure 4.6: Best fit on test set, non-sequential model with non-negative CNN preprocessed and current values using all available meteorological data. MSE 0.0021 m^2.

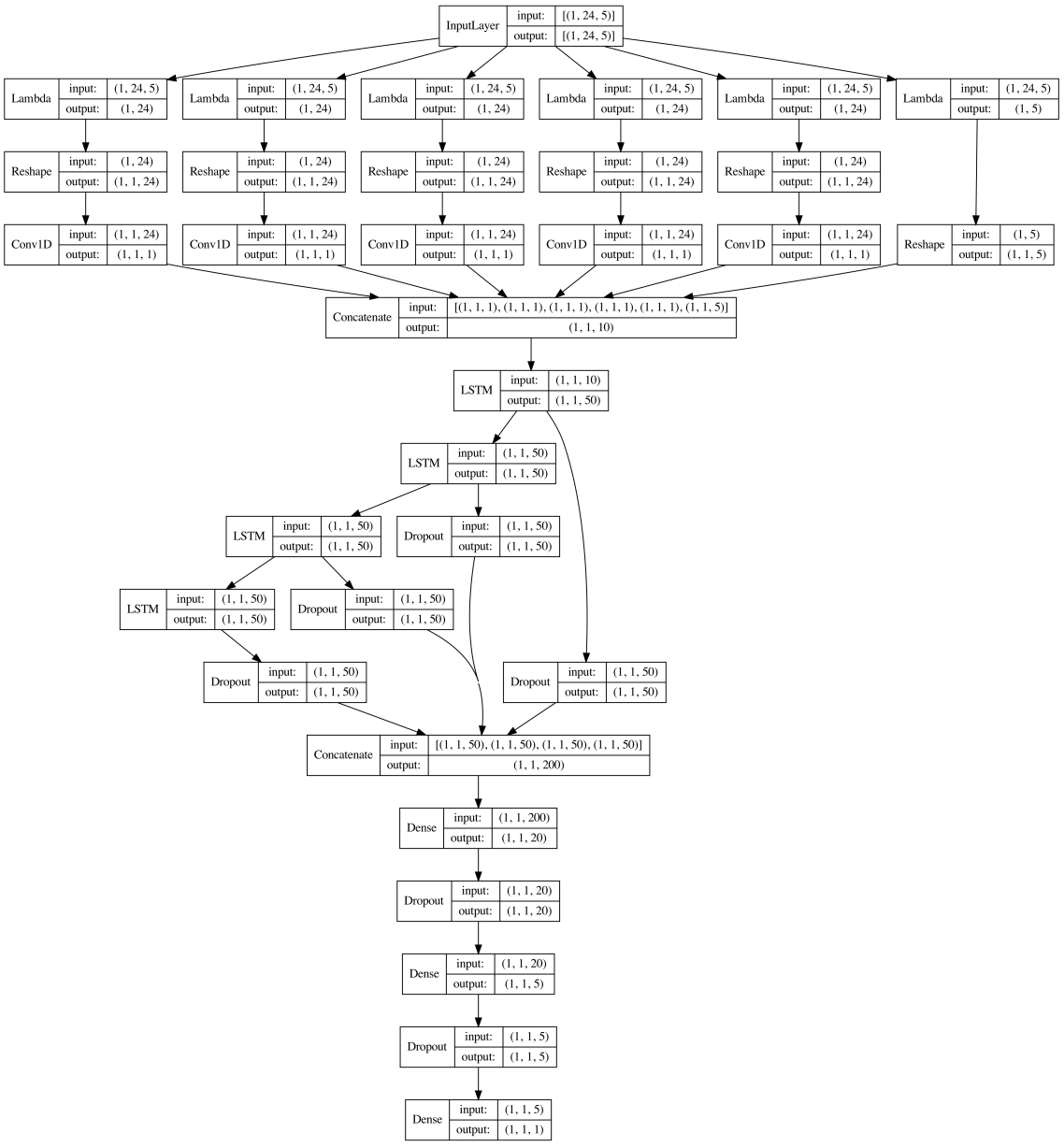


Figure 4.7: Best performing architecture, non-sequential model with non-negative CNN preprocessed and current values using all available meteorological data. Each block represents a layer, arrows represent flow of data between layers. Input and output data shapes are represented as a set of three digits on the form (batch size, number of time steps, number of features).

4.1.6 Non-sequential model with access to historical values

Are convolutional layers a good approach to summarizing the last 24 hours of data for consumption by the LSTM stage of the networks? Could not the LSTM stage handle summarizing by itself? To test this hypothesis, we eliminated the CNN preprocessing stage as seen in Figure 4.7, and fed the last 24 hours of data for all our features straight into the first LSTM layer. These networks performed on a level comparable to the networks in the Section 4.1.5, with the 2 LSTM layer architecture reaching an MSE of 0.0022 m^2 on the validation set (Figure 4.8, the best result on the validation set in this report), and the 4 and 5 LSTM layer architectures reaching an MSE of 0.0024 m^2 on the test set. The 2 LSTM layer architecture did not perform as well on the test set, reaching an MSE of 0.0033 m^2 (Figure 4.9). Hyperparameters for the best validation fit are listed in Table 4.7, performance across the explored hyperparameter space is visualized in Figure A.6. Visually comparing Figure 4.5 against 4.8, and Figure 4.6 against 4.9, it appears that this model, without the CNN preprocessing, is somewhat less noisy and qualitatively looks more like the observed signal, even though that is not clearly reflected in the MSE performance of the model (0.0022 m^2 for this model versus 0.0024 m^2 for the model in Section 4.1.5 on the validation set, 0.0024 m^2 for this model versus 0.0021 m^2 on the test set for the model in Section 4.1.5).

Table 4.7: Hyperparameters for the best performing non-sequential model with access to historical values using all available meteorological data.

Hyperparameter	Value
Number of LSTM layers	2
LSTM layer width	50
Dropout rate	0.1
First dense layer width	40
Second dense layer width	10
Batch size	1
Learning rate	10^{-5}

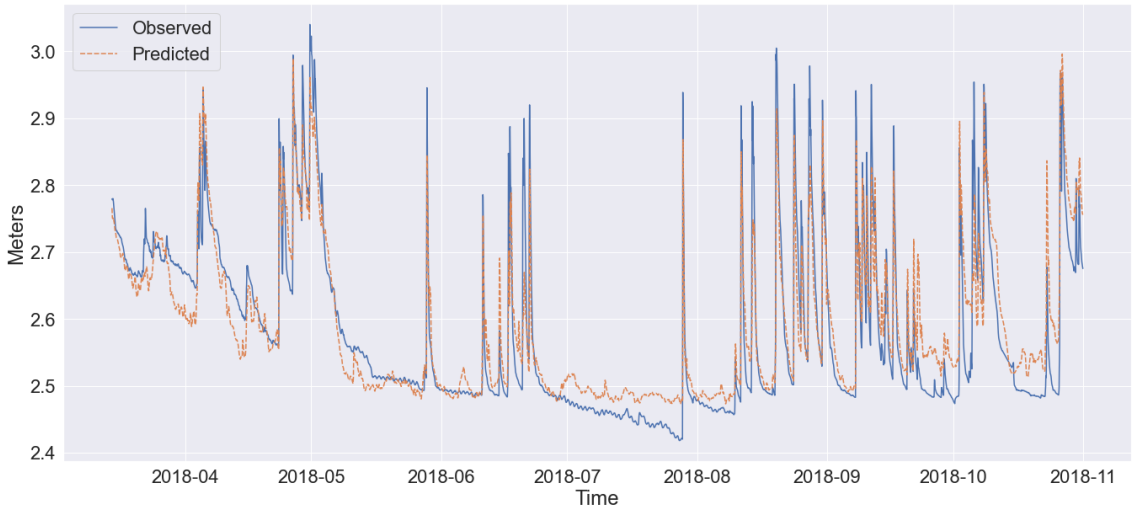


Figure 4.8: Best fit on validation set, non-sequential model with access to historical values using all available meteorological data. MSE 0.0022 m^2.

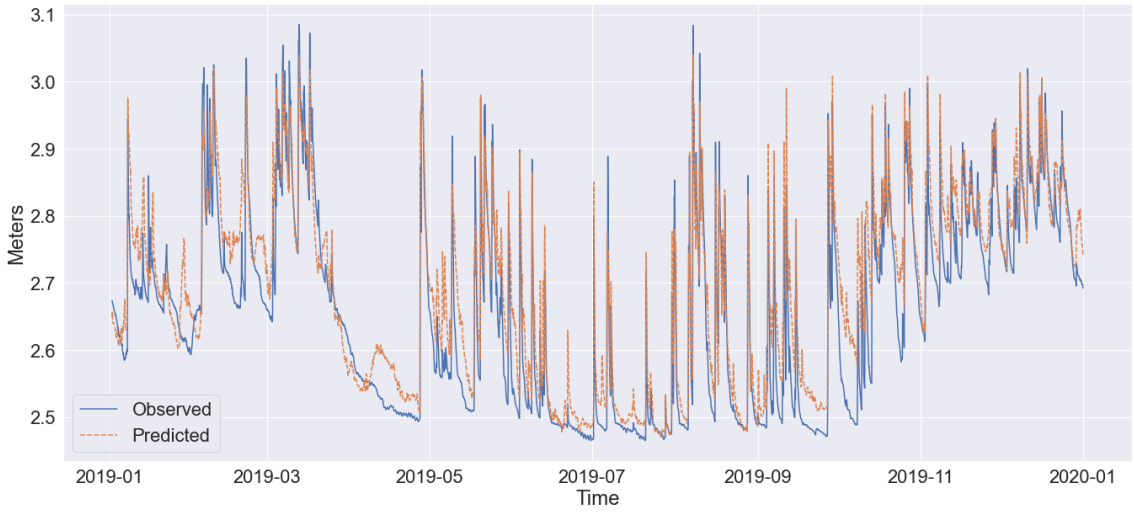


Figure 4.9: Best fit on test set, non-sequential model with access to historical values using all available meteorological data. MSE 0.0033 m^2.

4.1.7 Non-sequential model with a subset of the meteorological data as input

To further explore the data, we reran the grid search on a subset of the hyperparameters while only supplying 4 of the 5 inputs to the model. The subset of hyperparameters was chosen to encompass the best performing hyperparameters for the case where all inputs were supplied to the model. This should give us an indication of how important the removed variable is for the performance of the model. The results of this exploration is visualized in Figure 4.10, and it shows that the model performs significantly worse when removing temperature and rainfall, indicating that these two input variables are the most important to the model. For performance on the test set, please see Table 4.8.

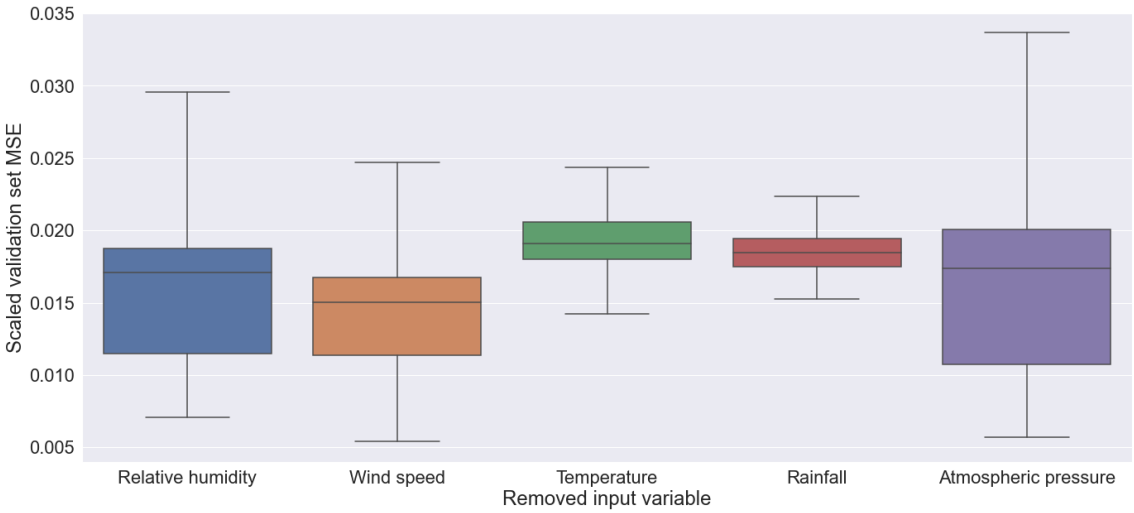


Figure 4.10: Performance on validation set, non-sequential model with one of the input variables removed.

4.1.8 Comparison to physical and mathematical models

There exists a variety of groundwater level models not depending on the use of machine learning, such as theoretically based models [43], numerical methods [44] and several others. To gauge the performance of our models, we will compare our results to one of them, and also a pure mathematical linear regression.

4.1.8.1 Simple groundwater accumulation and discharge model

Crossley et al. [43] introduce a simple model for groundwater levels,

$$h(i) = \sum_{j=0}^{j=i} r(j) \left(1 - e^{-(i-j)/\tau_1}\right) e^{-(i-j)/\tau_2}, \tag{4.1}$$

where $h(i)$ is the groundwater level at time i, $r(j)$ is the amount of rainfall at time j and τ_1 and τ_2 are time constants. It was applied to the measurements from Onsala, and the Nelder-Mead simplex direct search method was applied to

find the best fit, defined as the minimum MSE between observed and modeled groundwater levels. Before the hydrological model was applied, the minimum value of the observed groundwater level was deducted from those observations, since the model asymptotically approaches 0 with no rain. The model was fitted on the same data used for training the neural networks, and evaluated on the data used as validation data for the neural networks. The best fit on the training data was achieved with $\tau_1 = 1.0155$ days and $\tau_2 = 0.2757$ days, yielding an MSE of 0.0395 m^2. This model applied to the validation data yielded an MSE of 0.0210 m^2. The modeled groundwater level is presented in Figure 4.11.

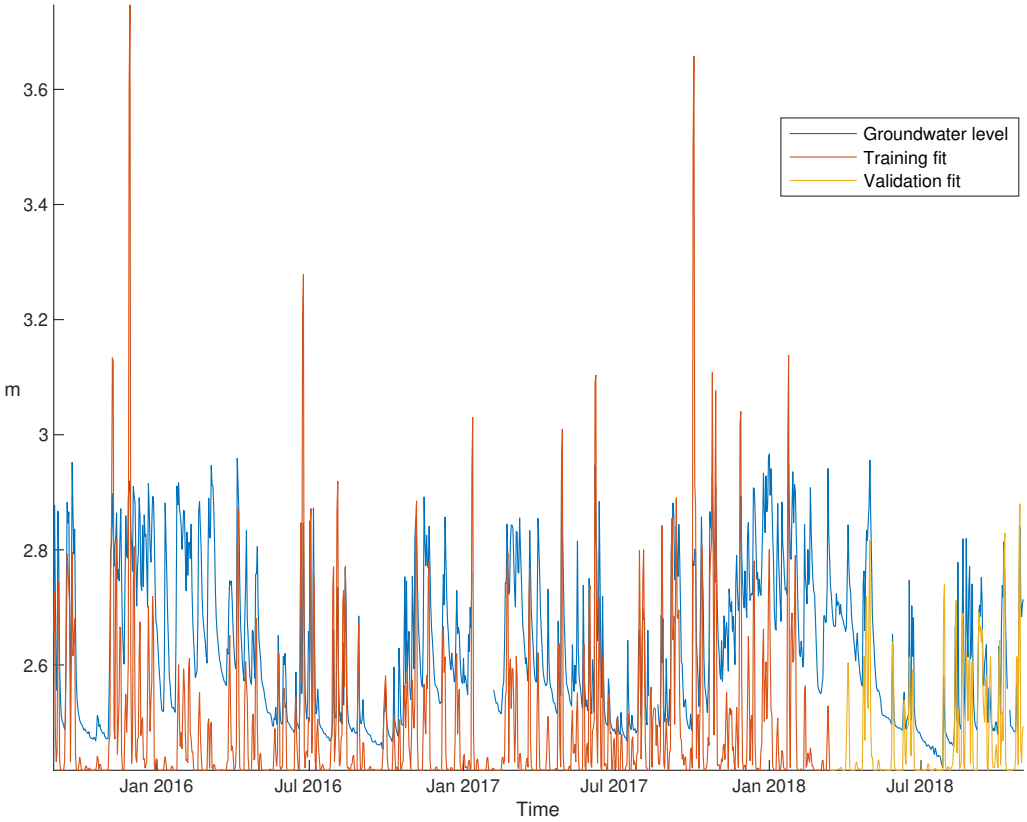


Figure 4.11: Best fit to observed groundwater levels with the simple groundwater accumulation and discharge model (Equation 4.1). MSE 0.0395 m^2 for the training fit, 0.0210 m^2 for the validation fit.

4.1.8.2 Modified groundwater accumulation and discharge model

The original model produces a qualitatively quite poor fit. An argument can be made that water entering or leaving the aquifer can do so through paths that differ in capacity and speed, you could construct a modified model which is a linear combination of original simple models,

$$h(i) = \sum_n \left(k_n \sum_{j=0}^{j=i} r(j) \left(1 - e^{-(i-j)/\tau_{1n}}\right) e^{-(i-j)/\tau_{2n}} \right), \tag{4.2}$$

with the new variable k_n as a scaling factor. Fitting this modified model, with $n = 2$, to the observed groundwater data yields a better fit (seen in Figure 4.12), reaching an MSE of 0.0140 m^2 on the training data and 0.0087 m^2 on the validation data. Further increase of n yields little to no improvement in MSE. This modified model

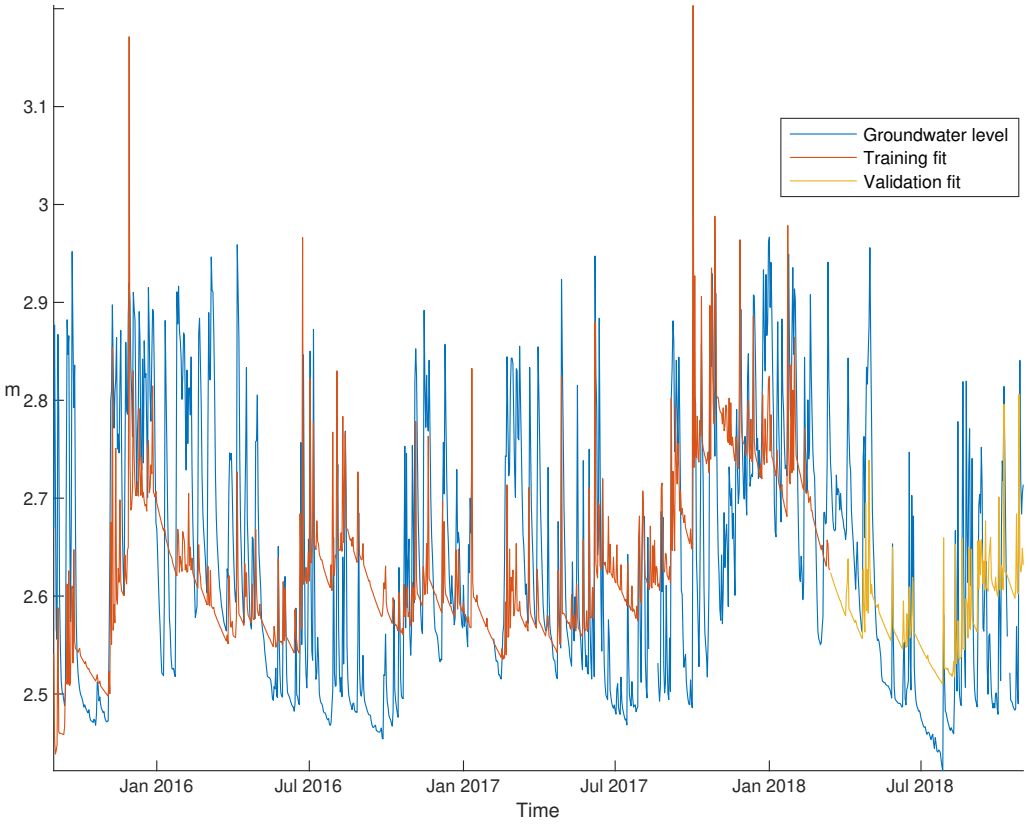


Figure 4.12: Best fit to observed groundwater levels with the modified model, $n = 2$, (Equation 4.2). MSE 0.0140 m^2 for the training fit, 0.0087 m^2 for the validation fit.

with $n = 2$ will serve as the baseline of comparison to judge the performance of the deep learning methods used in this report.

4.1.8.3 Elastic net regularized linear regression

The elastic net is a regularization method for linear regression [45]. In this project, we have 5 predictors (our input variables) to estimate the groundwater level with. An ordinary least squares linear regression could leave us in a situation where we have a lot of variance in our estimated groundwater level. Elastic net tries to regularize this by penalizing large weights when including a predictor in the linear regression model. It combines a method called Ridge regression, which augments the ordinary least squares loss by also penalizing large predictor weights proportionally to the square of the weight, and Lasso regression which does the same thing but penalizing proportionally to the absolute value of the weight. This regularization means that the loss function takes the form of

$$L = \frac{1}{2n}|\mathbf{y} - \mathbf{X}\mathbf{w}|_2^2 + \alpha r|\mathbf{w}|_1 + \frac{1}{2}\alpha(1-r)|\mathbf{w}|_2^2. \tag{4.3}$$

With the first term in Equation 4.3 being the normal ordinary least squares error, the second term is the Lasso term which penalizes the absolute value of the weigths, and the third term being the Ridge term which penalizes the square of the weights. n is the number of samples, $\mathbf{y}$ are the true observed values, $\mathbf{X}$ is our input data, $\mathbf{w}$ is the weights applied to our input, α represents the amount of regularization applied and r determines the ratio of Lasso vs Ridge regularization applied. In order to compare our results with a linear regression, a 10-fold cross-validated grid search of an elastic net regularized linear regression was performed in order to determine the best values of the regularization parameters α and r when fitted to the data from the training set. The grid search yielded a lowest MSE of 0.0117 m^2, with an α of 0.01 and an r of 0 (meaning that the model purely focuses on the Ridge penalty). On the validation set, the model achieved an MSE of 0.0102 m^2 . This places the linear regression performance quantitatively somewhere in between the two physical models discussed above.

4.1.9 Summary of results

The networks using rainfall only as its input are vastly outperformed by networks using all available meteorological data, but even so the performance is somewhat better than a simple deterministic model (Eq. 4.1), but worse than a modified model (Eq. 4.2). The fact that the models using all meteorological data outperform the rainfall only model is not surprising, since the inputs that are added (wind, temperature, atmospheric pressure, humidity) all significantly impact how much of the water introduced as rain actually makes it into the water table.

Non-sequential networks perform better than sequential networks when using all available meteorological data as input.

From the performance of non-sequential networks using all but one of the available meteorological inputs, it appears that air pressure is the least important input, and rainfall the most important. This contrasts with the correlations in Table 4.1, where rain was the least correlating variable and air pressure the most.

Adding a CNN preprocessing stage before the LSTM stage of the non-sequential models did not increase performance. However, by supplying the LSTM stages with both the CNN preprocessed 24-hour history and the current values of the inputs, we could see a further performance increase over the non-sequential models.

When supplying the raw 24 hour history directly to the LSTM stage, we saw performance comparable with the best performing networks using CNN preprocessing.

The best performing neural network models outperform existing hydrological models. The best performing model, the non-sequential LSTM using 4 LSTM layers, with non-negative CNN and current values as input, had an average prediction error of 0.013 meters (with a 95 % confidence interval of [-0.078, 0.105] for a single prediction assuming a generalized normal distribution [46]), visualized in Figure 4.14. By applying the best performing model on the meteorological data available from the period 2010-01-01 until 2015-08-30, which is before the groundwater measuring well was constructed at the Onsala Space Observatory, we are able to reconstruct almost 6 years of groundwater levels (Figure 4.13), which will be used when moving on to predict the residual gravity. A full summary of results across models is available in

Table 4.8.

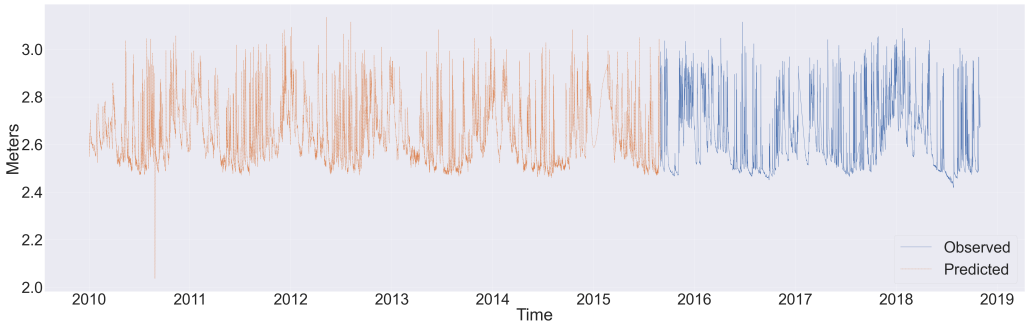


Figure 4.13: Predicted groundwater levels 2010-01-01 - 2015-08-30, observed groundwater levels 2015-08-31 - 2018-10-31. Some artefacts can be seen in the predicted levels, such as an outlier in late 2010 and a period in early 2015 caused by lack of meteorological data. Blue values are actual measurements, orange values are predicted values.

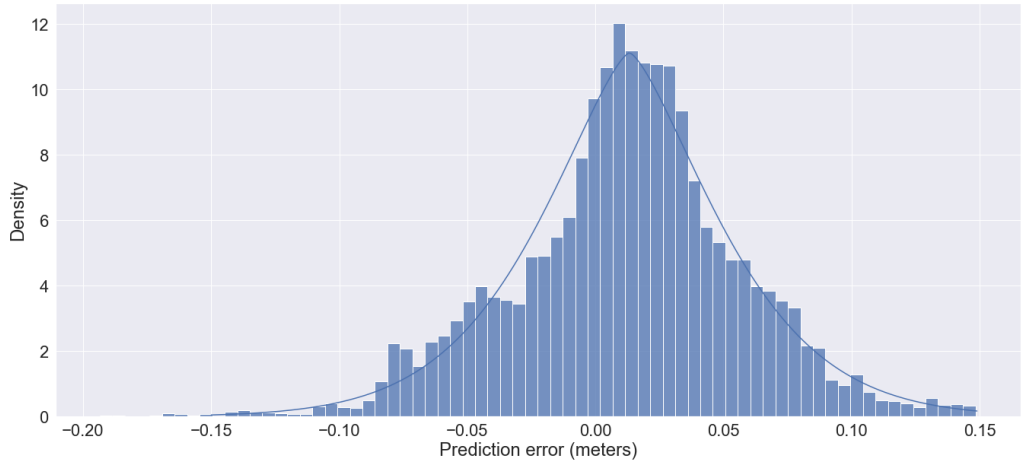


Figure 4.14: Prediction error probability density plot by the best performing neural network model. A fitted generalized normal distribution with parameters $\mu = 0.0133$, $\alpha = 0.0494$, $\beta = 1.4166$ is also plotted for comparison.

4.1.9.1 Impact of LSTM unit state

Using the best performing model and its predictions on the validation set, we can get an estimate of how much of past data is used by the LSTM cells in order to reach its optimal state. Performing the predictions on the validation set with the cell state as it was at the end of the training set (which immediately precedes the validation set), and also performing the predictions with the cell state reset to zero, shows us how long it takes for the model without the proper initial state to converge with the model using the proper state (Figure 4.15). The model needs almost 10 days before the predictions converge, giving us an estimate of the window of time needed for the model to reach this level of accuracy in its predictions.

Table 4.8: Summary of groundwater results. Underlined values are the best performing models within the group, bold is best performance overall.

Model	Input data	# of LSTM layers	Val. set MSE (m^2)	Test set MSE (m^2)
Sequential LSTM	Rainfall only	1	0.0134	0.0145
		2	0.0135	0.0150
		3	<u>0.0129</u>	<u>0.0133</u>
		4	0.0136	0.0146
		5	0.0136	0.0179
Sequential LSTM	All met. data	1	0.0037	<u>0.0033</u>
		2	<u>0.0029</u>	0.0039
		3	0.0040	0.0046
		4	0.0044	0.0051
		5	0.0043	0.0054
Non-sequential LSTM	All met. data	2	0.0030	0.0032
		3	0.0031	0.0038
		4	<u>0.0027</u>	<u>0.0027</u>
		5	0.0029	0.0040
Non-sequential LSTM	No wind speed		<u>0.0024</u>	0.0033
	No temperature		0.0051	0.0045
	No humidity		0.0032	0.0041
	No air pressure		0.0026	<u>0.0028</u>
	No rainfall		0.0067	0.0090
Non-sequential LSTM with unconstrained CNN	All met. data		0.0096	<u>0.0137</u>
Non-sequential LSTM with non-negative CNN	All met. data		<u>0.0095</u>	0.0148
Non-sequential LSTM with non-negative CNN and current values	All met. data	2	0.0026	0.0027
		3	0.0025	0.0032
		4	<u>0.0024</u>	<u>**0.0021**</u>
		5	0.0025	0.0048
Non-sequential LSTM	Last 24 hours of all met. data	2	<u>**0.0022**</u>	0.0033
		3	0.0024	0.0033
		4	0.0025	<u>0.0024</u>
		5	0.0024	<u>0.0024</u>
Simple model (Eq. 4.1)	Rainfall only		0.0210	
Modified model (Eq. 4.2)	Rainfall only		<u>0.0087</u>	
Elastic net linear regression	All met. data		0.0117	0.0102

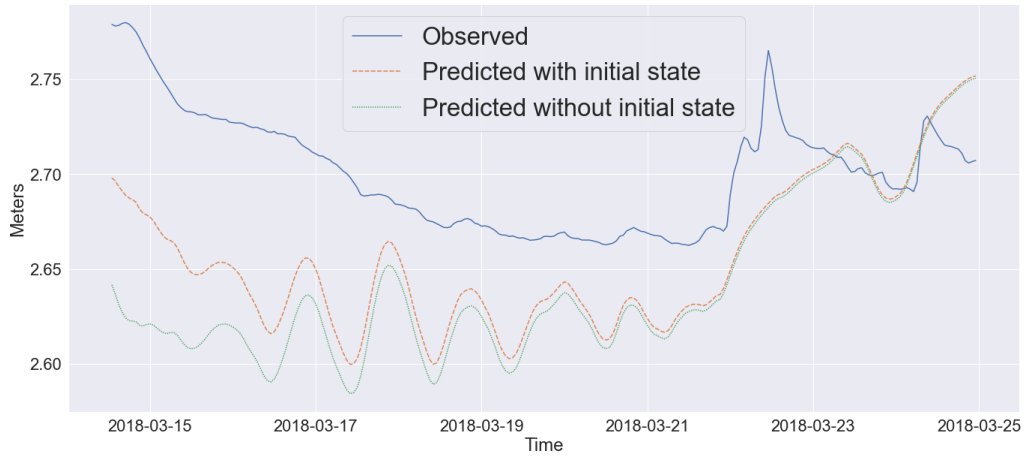


Figure 4.15: Effect of initial state on predictions (excerpt from the start of the validation set).

4.2 Gravity predictions

Residual gravity was predicted using the same sequential and non-sequential models used in the groundwater case, using "Residual gravity" as the target quantity, and all the remaining 6 quantities in Figure 3.2 as inputs. The Pearson correlation coefficient between input quantities and target quantity are listed in Table 4.9. In order to gauge the impact of temperature, an input quantity that has no direct causal relationship with the residual gravity we are trying to predict, predictions were made without temperature as an input as well. Also, one of the best performing models from the groundwater modelling was applied to this problem as well. Rainfall is almost completely uncorrelated with the target quantity and would in a normal regression scenario be removed, but since we are using an LSTM based model, uncorrelated does not mean unimportant (see the importance of rainfall in the groundwater case), so we will keep the rainfall as an input quantity.

Table 4.9: Input quantity correlations with residual gravity across the training and validation sets.

Input quantity	Correlation with residual gravity
Groundwater level	0.487
Wind speed	0.099
Temperature	−0.580
Relative humidity	0.227
Rain last 1 hour	−0.001
Air pressure	−0.121

4.2.1 Sequential model

The sequential model with the best performance on the validation set, using 4 LSTM layers, achieved an MSE of 14.298 $\mathrm{nm^2 s^{-4}}$ on the validation set (Figure 4.16). The

same model however performed significantly worse on the test set, achieving an MSE of 214.613 nm^2s^{-4} (Figure 4.17). Comparing figures 4.16 and 4.17, we can see that the test set contains a strong long-term seasonal trend not present in the validation set. This might have caused the training to stop once performance no longer improved on the validation set, and the network had at that time not yet learned how to model the long-term signal. If we instead look at the model that performed the best on the test set (MSE 29.835 nm^2s^{-4}), using 2 LSTM layers (Figure 4.18), we can see that it has done a better job at learning the long-term trend. The hyperparameters for the model with the best performance on the validation set are listed in Table 4.10, the hyperparameters for the model with the best performance on the test set are listed in table . Performance across the explored hyperparameter space is visualized in Figure A.7.

Table 4.10: Hyperparameters for the best performing sequential model on the gravity validation set.

Hyperparameter	Value
Number of LSTM layers	4
LSTM layer width	100
Dropout rate	0.0
First dense layer width	20
Second dense layer width	10
Batch size	1
Learning rate	10^{-5}

Table 4.11: Hyperparameters for the best performing sequential model on the gravity test set.

Hyperparameter	Value
Number of LSTM layers	2
LSTM layer width	5
Dropout rate	0.1
First dense layer width	10
Second dense layer width	50
Batch size	100
Learning rate	10^{-3}

4.2.2 Non-sequential model

The best performing non-sequential model achieved an MSE of 10.961 nm^2s^{-4} on the validation set (Figure 4.19). It achieved an MSE of 264.456 nm^2s^{-4} on the test set (Figure 4.20). The model has not been able to capture the long term seasonal changes present in the test set. The hyperparameters for the best performing model are listed in Table 4.12. Performance across the explored hyperparameter space is visualized in Figure A.8.

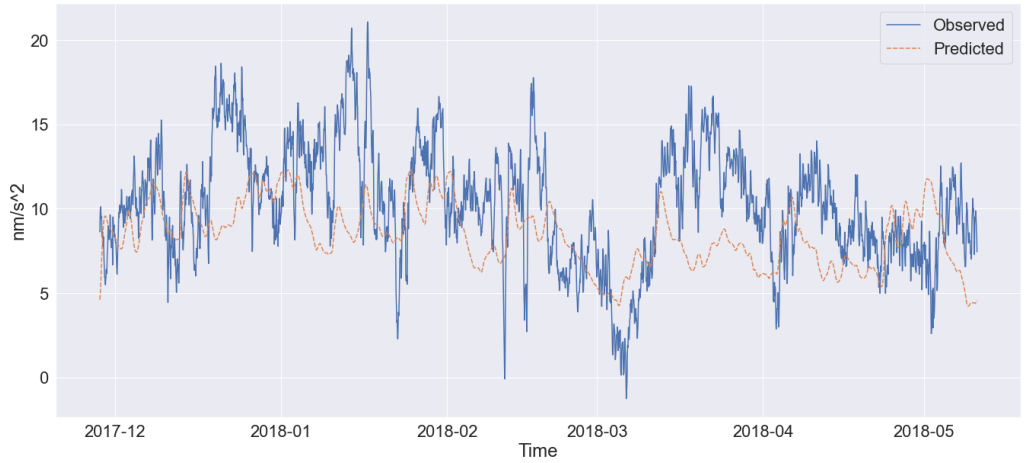


Figure 4.16: Best fit on gravity validation set, sequential model with 4 LSTM layers. MSE 14.298 $\mathrm{nm}^2\mathrm{s}^{-4}$.

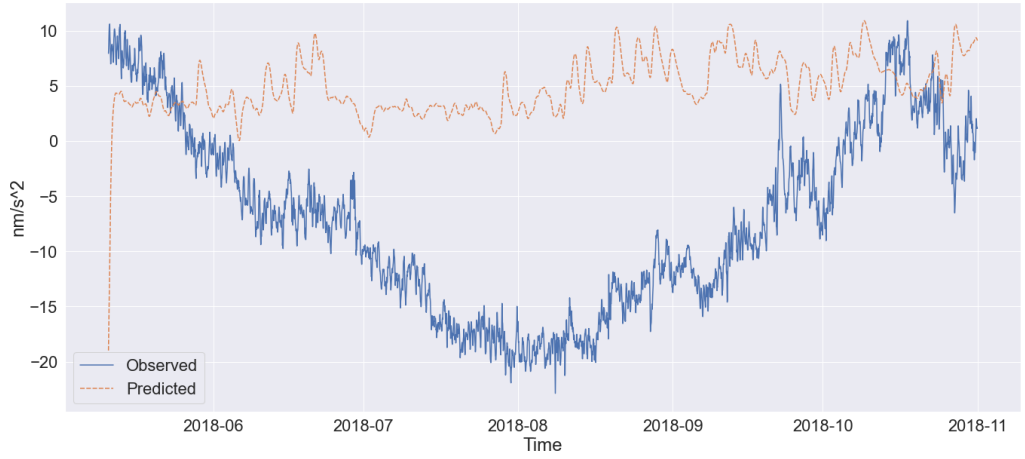


Figure 4.17: Best fit on gravity test set, sequential model with 4 LSTM layers. MSE 214.613 $\mathrm{nm}^2\mathrm{s}^{-4}$.

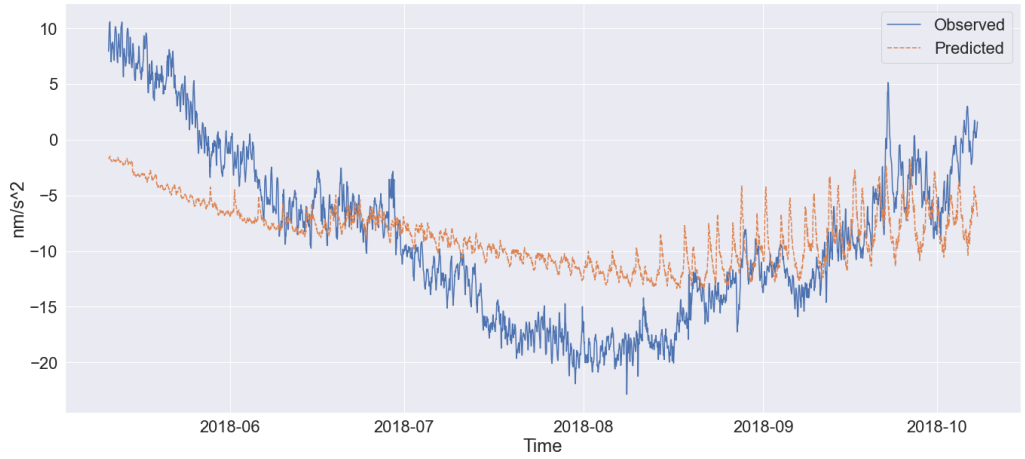


Figure 4.18: Best fit on gravity test set, sequential model with 2 LSTM layers. MSE 29.835 $\mathrm{nm}^2\mathrm{s}^{-4}$.

Table 4.12: Hyperparameters for the best performing non-sequential gravity model.

Hyperparameter	Value
Number of LSTM layers	2
LSTM layer width	10
Dropout rate	0.3
First dense layer width	20
Second dense layer width	5
Batch size	1
Learning rate	10^{-5}

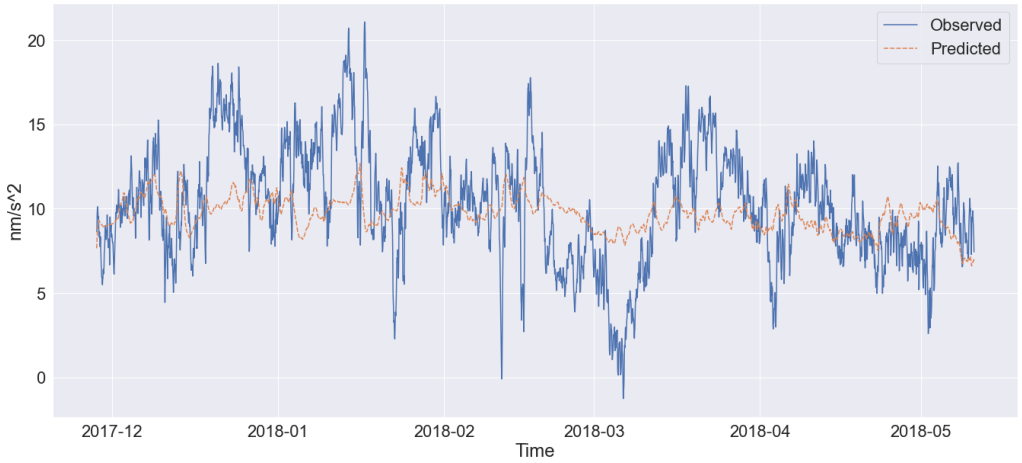


Figure 4.19: Best fit on gravity validation set, non-sequential model. MSE 10.961 $\mathrm{nm}^2\mathrm{s}^{-4}$.

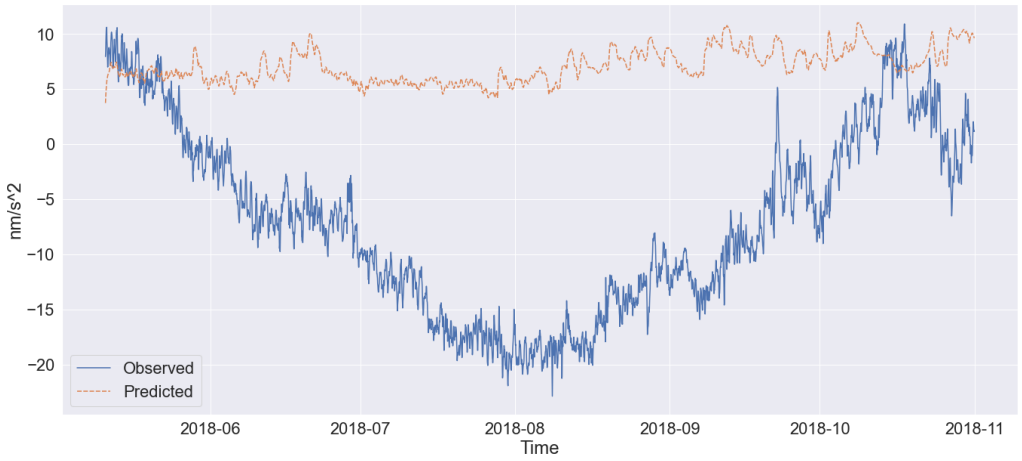


Figure 4.20: Best fit on gravity test set, non-sequential model. MSE 264.456 $\mathrm{nm}^2\mathrm{s}^{-4}$.

4.2.3 Non-sequential model with simulated groundwater input

The non-sequential model performed badly on data not encountered during training, with an MSE an order of magnitude higher on the test set than on the validation set. In order to test the validity of our results when predicting groundwater levels, we augmented the data set used when predicting residual gravity with the predicted groundwater levels in Figure 4.13.
This gave us almost 5 more years of groundwater data, enabling us to train the non-sequential model on a much bigger data set. Using the 70/15/15 % split, our training set is now 2010-01-01 - 2016-02-17, our validation set is 2016-02-18 - 2017-05-11 and our test set is 2017-05-12 - 2018-10-31. All other parameters were left unchanged and the models were trained. The best performing model trained on data ranging back to 2010 was an MSE of 11.935 nm^2s^{-4} on the period 2017-11-18 - 2018-05-10 (the validation set for the models in Section 4.2.2, part of the set for this model), and 35.536 nm^2s^{-4} on the period 2018-05-11 - 2018-10-31, the test set for the models in Section 4.2.2. This is a major improvement, considering the best result for the non-sequential model using data from 2015 and forward only achieved an MSE of 235.895 nm^2s^{-4} on the test set, an improvement by a factor of about 6.5. The hyperparameters for the best performing model are listed in Table 4.13. The fit on the test set for this model, 2017-05-12 - 2018-10-31, is visualized in Figure 4.21. In order to provide comparisons with the models in Section 4.2.2 and specifically figures 4.19 and 4.20, the fit on the period 2017-11-18 - 2018-05-10 is visualized in Figure 4.22 and the fit on the period 2018-05-11 - 2018-10-31 is visualized in Figure 4.23. The performance across hyperpameter space is visualized in Figure A.9.

Table 4.13: Hyperparameters for the best performing non-sequential gravity model trained on data starting in 2010, including simulated groundwater measurements.

Hyperparameter	Value
Number of LSTM layers	3
LSTM layer width	5
Dropout rate	0.0
First dense layer width	20
Second dense layer width	20
Batch size	10
Learning rate	10^{-4}

4.2.4 Sequential model without temperature

When removing temperature as an input, the performance of our model dropped. The best performing model achieved an MSE of just 18.385 nm^2s^{-4} on the validation set (Figure 4.24), and 60.344 nm^2s^{-4} on the test set (Figure 4.25). Looking at the hyperparameters for the best performing model (Table 4.14), we can see that the best performing model is a very small one, with just a single LSTM unit in 1 single layer. A small model has a small computational capacity, indicating that the

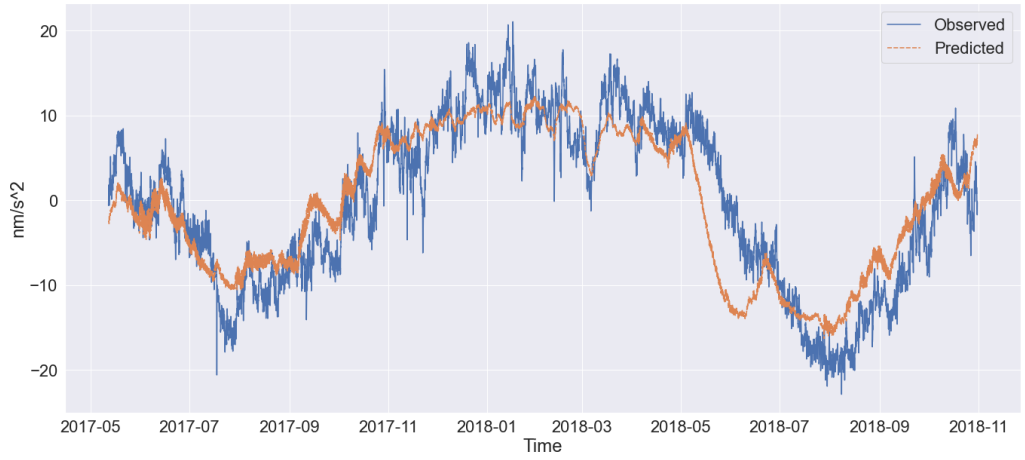


Figure 4.21: Best fit on the model native gravity test set, non-sequential model trained on data starting in 2010, including simulated groundwater measurements.

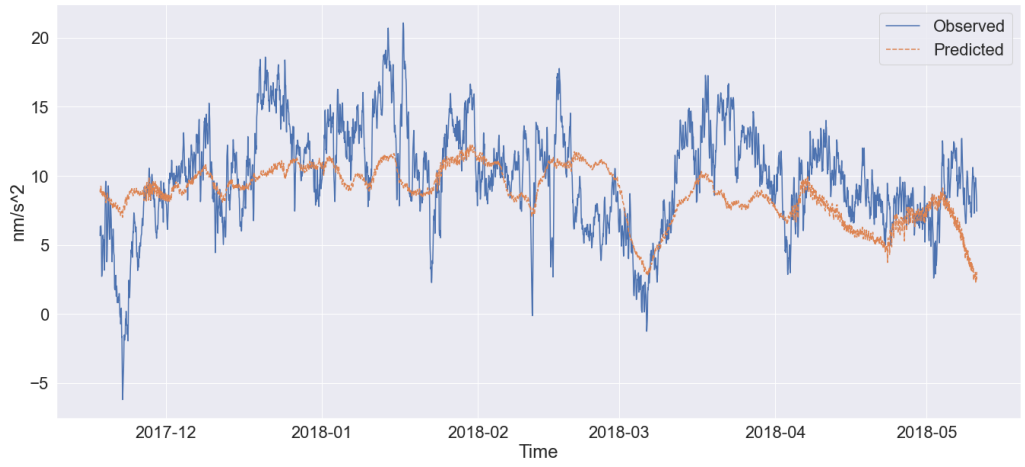


Figure 4.22: Best fit on original gravity validation set, non-sequential model trained on data starting in 2010, including simulated groundwater measurements. MSE 11.935 $\mathrm{nm}^2\mathrm{s}^{-4}$.

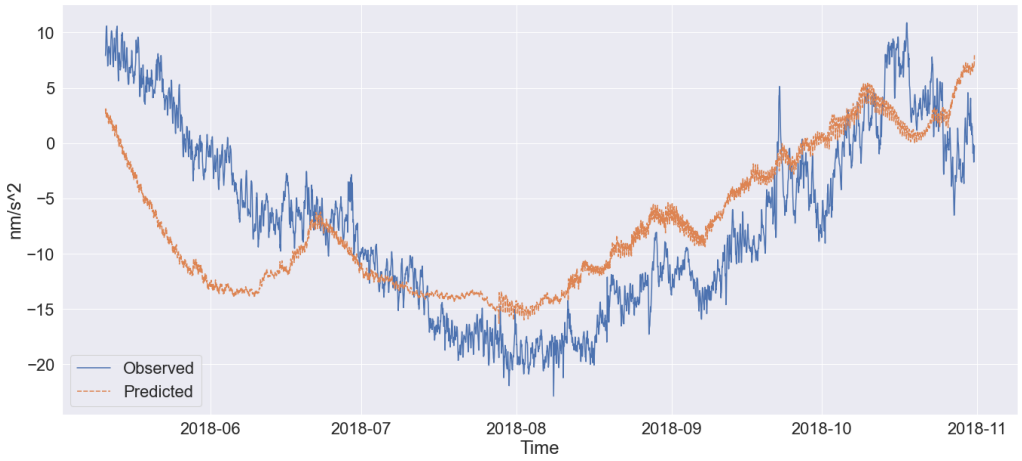


Figure 4.23: Best fit on original gravity test set, non-sequential model trained on data starting in 2010, including simulated groundwater measurements. MSE 35.536 nm^2s^{-4}.

model was only able to rudimentary fit the inputs to the outputs. If there were more complex relationships between the inputs and outputs, a more complex model would have performed better. Performance across the explored hyperparameter space is visualized in Figure A.10.

Table 4.14: Hyperparameters for the best performing sequential gravity model without temperature.

Hyperparameter	Value
Number of LSTM layers	1
LSTM layer width	1
Dropout rate	0.0
First dense layer width	10
Second dense layer width	10
Batch size	100
Learning rate	10^{-3}

4.2.5 Comparison to a linear regression

Just as in the groundwater case in Section 4.1.8.3, a 10-fold cross-validated grid search of an elastic net regularized linear regression was performed in order to determine the best values of the regularization parameters α and r when fitted to the data from the training set. The grid search yielded a lowest MSE of 31.430 nm^2s^{-4}, with an α of 10^{-4} and an r of 0. On the validation set, the model achieved an MSE of 50.363 nm^2s^{-4}, and on the test set an MSE of 71.315 nm^2s^{-4}.

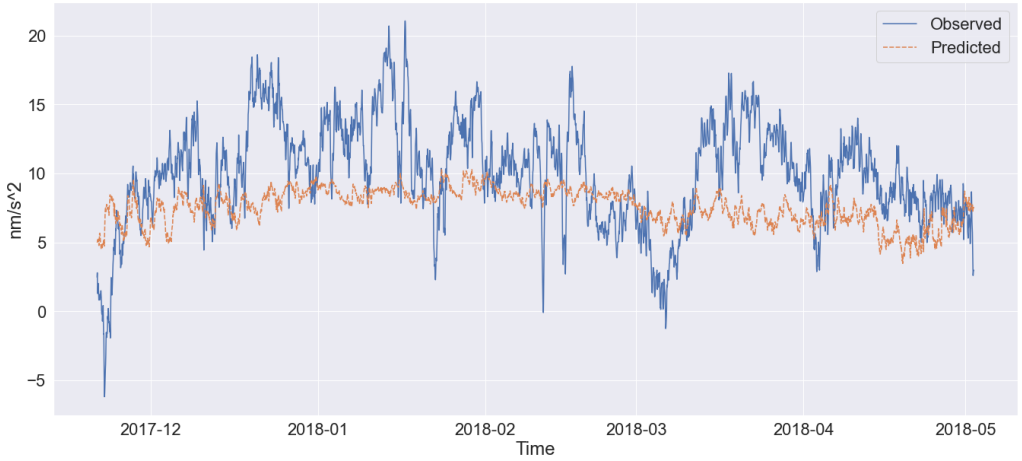


Figure 4.24: Best fit on gravity validation set, sequential model without temperature input. MSE 18.385 $\mathrm{nm}^2\mathrm{s}^{-4}$.

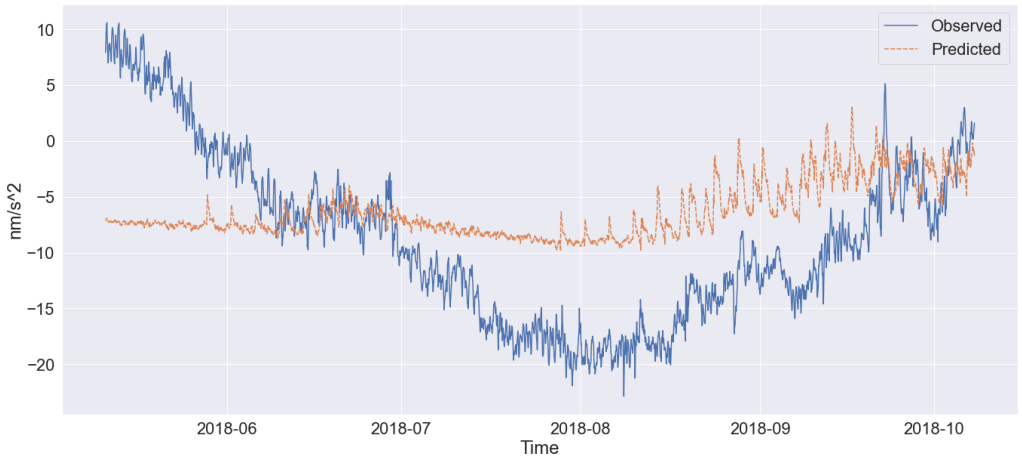


Figure 4.25: Best fit on gravity test set, sequential model without temperature input. MSE 60.344 $\mathrm{nm}^2\mathrm{s}^{-4}$.

4.2.6 Summary of gravity results

The best performing model on the validation set is the non-sequential model (Figure 4.19), but the non-sequential model trained on simulated groundwater measurements (Figure 4.22) has fitted the nuances of the validation set better, despite the slightly higher MSE. On the test set, the best performing model is the sequential model with 2 LSTM layers (Figure 4.18). However, despite its slightly higher MSE, the non-sequential model trained with simulated groundwater measurements (Figure 4.23), appears to be picking up much more of the nuances in the observed data. No model using data from 2015 and onwards fits the seasonal trend in the test set in a good way. The presence of a seasonal trend in the test set, which is not present in the validation set, likely causes training to end before the network is able to fit long-term changes, because the validation set loss function will not reward a model that fits the long-term changes. The average error of the non-sequential model trained with simulated groundwater measurements across the test set was -0.09 nms^{-2}, with a 95 % confidence interval of [-17.99, 7.72] assuming a generalized logistic distribution (Figure 4.26). A full summary of results across models is available in Table 4.15.

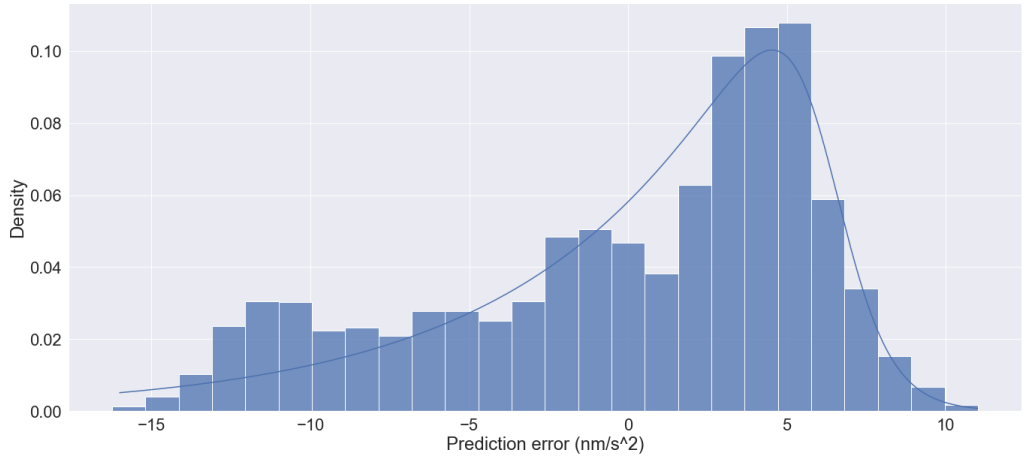


Figure 4.26: Gravity prediction error probability density plot by the best performing neural network model. A fitted generalized logistic distribution is also plotted for comparison

Table 4.15: Summary of gravity results. Underlined values are the best performing models within the group, bold is best performance overall. Values denoted with a * were in the model test set, listed in the validation set column for comparison reasons (see discussion in Section 4.2.3).

Model	Input data	# of layers	Val. set MSE (nm^2s^{-4})	Test set MSE (nm^2s^{-4})
Sequential LSTM	All met. data	1	30.928	49.137
		2	22.156	**29.835**
		3	27.748	63.781
		4	14.298	214.613
		5	15.937	247.830
Non-sequential LSTM	All met. data	2	**10.961**	264.456
		3	19.664	235.895
		4	18.516	260.615
		5	11.887	246.190
Non-sequential LSTM	All met. data since 2010 incl. simulated groundwater	2	17.162*	36.987
		3	11.905*	35.536
		4	20.285*	44.198
		5	18.155*	36.934
Sequential LSTM	All met. data except temperature	1	18.385	60.344
		2	22.244	56.044
		3	18.418	55.274
		4	23.504	61.566
		5	20.003	192.408
Elastic net linear regression	All met. data		50.363	71.315

5
Conclusion

5.1 Gravity predictions

Modelling the gravity was initially a difficult task for the neural networks. By augmenting the available data with predicted groundwater levels, we were able to more than double the amount of data on which to base our models, with a dramatic increase in performance as a result (Figure 4.21). This model achieved an MSE of 35.536 nm^2s^{-4} (RMSE 5.961 nms^{-2}) on the test set, a more than 6-fold improvement over the networks trained with data from only 2015 and onwards (Figure 4.20). The augmentation of data sets with simulated measurements appears to be a viable way to improve neural network performance in this setting.
For residual gravity predictions, one focus for improvement should be in more precise preprocessing of the data. The residual gravity signal in this report still has some components at tidal frequencies, and the effects of global hydrology has not been accounted for.

5.2 Groundwater predictions

Overall, recurrent neural networks were able to predict groundwater levels quite accurately. The best model fitted the test set with an MSE of 0.0021 m^2 (RMSE of 0.046 m). The addition of historical data (be it in convoluted or direct form) as inputs to the LSTM stage helped the models to perform better. One could argue that the history of an input variable should be something that the LSTM unit should be able to memorize on its own (since that is the expressed purpose of the LSTM unit), but by lending them a hand by giving them direct access to historical values, it could free up storage capacity in the LSTM units to focus on the finer details in the signal. What could be done to further improve the performance of these predictions?
One potential area of improvement would be the models themselves. As seen, the best performing models have their pros and cons. What if we increase the amount of historical data supplied to the network? What if we tried other regularization techniques, such as L1/L2 regularization instead of or in addition to dropout? Could the combination of supplying 24 hours of historical data in both direct and convoluted form further improve the performance?
We could also focus on improving and expanding the inputs to the model. Evapotranspiration is an important factor in determining how much of rainfall makes it into the water table. Evapotranspiration is very hard to measure directly, but

if eddy covariance measurements were available at the Onsala Space Observatory site, it would be interesting to rerun these experiments to see if the accuracy would improve. Several of the input variables we do have access to influence evapotranspiration, but direct measurements would be an interesting addition to the line-up. It is also widely used in other similar efforts [6]. Another interesting input would be soil temperature. The soil acts as a big buffer, meaning the soil temperature fluctuates much more slowly then the air temperature we have been using for these models. Soil humidity could also be a measurement of interest. One thing we have been unable to ascertain is how the rain gauge in Onsala handles snowfall. If it registers snow just as it does rain, that would lead to issues for our models since they can not distinguish between rain and snow in the input data, and that difference would have an impact on the observed groundwater level.
The model has performed well enough to be a viable way of extracting historical groundwater levels from meteorological data at the Onsala Space Observatory (Figure 4.13). One important thing to note is that the performance of the model was predicated on the existence of accurate actual groundwater measurements from the location in question. This means that we can apply the model on historical meteorological data in order to produce approximate historical groundwater levels (assuming that we can assume that the geological conditions have remained somewhat constant), but we can not apply the model to another physical location and expect to achieve an accurate prediction, since geological conditions most likely are different at that location. The model produced in this report could serve as a starting point for validating its performance at other locations with different geological and meteorological condition. The same data sets would have to be available. This concept, where a model trained on one problem is later applied to another (related) problem is called transfer learning, an area of active research.
In order to produce a model capable of predicting groundwater levels at a location where no groundwater measurements are available, we would have to train a model on data from several locations at once, while also supplying the model with enough input information to deduce important geological factors (such as τ_1 and τ_2 in the discharge models used in this report). Dedicated research on this topic could lead to groundwater level models capable of not only generalizing in time, but also in space.

Bibliography

[1] Wikipedia. *Recurrent neural network — Wikipedia, The Free Encyclopedia*. http://en.wikipedia.org/w/index.php?title=Recurrent%20neural%20network&oldid=987141659. [Online; accessed 13-November-2020]. 2020.

[2] Sanghoon Lee, Kang-Kun Lee, and Heesung Yoon. "Using artificial neural network models for groundwater level forecasting and assessment of the relative impacts of influencing factors". In: *Hydrogeology Journal* 27.2 (Sept. 2018), pp. 567–579. DOI: 10.1007/s10040-018-1866-3. URL: https://doi.org/10.1007/s10040-018-1866-3.

[3] Riccardo Taormina, Kwok-wing Chau, and Rajandrea Sethi. "Artificial neural network simulation of hourly groundwater levels in a coastal aquifer system of the Venice lagoon". In: *Engineering Applications of Artificial Intelligence* 25.8 (Dec. 2012), pp. 1670–1676. DOI: 10.1016/j.engappai.2012.02.009. URL: https://doi.org/10.1016/j.engappai.2012.02.009.

[4] Atsushi Mukai. "Effect of groundwater on gravity observation at Kyoto". In: *Gravity, Geoid and Marine Geodesy*. Springer, 1997, pp. 123–130.

[5] Michel Van Camp, Olivier de Viron, Arnaud Watlet, Bruno Meurers, Olivier Francis, and Corentin Caudron. "Geophysics From Terrestrial Time-Variable Gravity Measurements". In: *Reviews of Geophysics* 55.4 (Nov. 2017), pp. 938–992. DOI: 10.1002/2017rg000566. URL: https://doi.org/10.1002/2017rg000566.

[6] Taher Rajaee, Hadi Ebrahimi, and Vahid Nourani. "A review of the artificial intelligence methods in groundwater level modeling". In: *Journal of Hydrology* 572 (May 2019), pp. 336–351. DOI: 10.1016/j.jhydrol.2018.12.037. URL: https://doi.org/10.1016/j.jhydrol.2018.12.037.

[7] M. I. Jordan and T. M. Mitchell. "Machine learning: Trends, perspectives, and prospects". In: *Science* 349.6245 (July 2015), pp. 255–260. DOI: 10.1126/science.aaa8415. URL: https://doi.org/10.1126/science.aaa8415.

[8] Markus Reichstein, Gustau Camps-Valls, Bjorn Stevens, Martin Jung, Joachim Denzler, Nuno Carvalhais, and Prabhat. "Deep learning and process understanding for data-driven Earth system science". In: *Nature* 566.7743 (Feb. 2019), pp. 195–204. DOI: 10.1038/s41586-019-0912-1. URL: https://doi.org/10.1038/s41586-019-0912-1.

[9] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[10] Wikipedia. *Artificial neural network — Wikipedia, The Free Encyclopedia*. http://en.wikipedia.org/w/index.php?title=Artificial%20neural%20network&oldid=984752210. [Online; accessed 28-October-2020]. 2020.

[11] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. "Deep sparse rectifier neural networks". In: *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings. 2011, pp. 315–323.

[12] Lazlo Tóth. "Phone recognition with deep sparse rectifier neural networks". In: *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. 2013, pp. 6985–6989. DOI: 10.1109/ICASSP.2013.6639016.

[13] Bekir Karlik and A Vehbi Olgac. "Performance analysis of various activation functions in generalized MLP architectures of neural networks". In: *International Journal of Artificial Intelligence and Expert Systems* 1.4 (2011), pp. 111–122.

[14] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Computation* 9.8 (Nov. 1997), pp. 1735–1780. DOI: 10.1162/neco.1997.9.8.1735. URL: https://doi.org/10.1162/neco.1997.9.8.1735.

[15] Alex Graves. "Generating sequences with recurrent neural networks". In: *arXiv preprint arXiv:1308.0850* (2013).

[16] Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. "Empirical evaluation of gated recurrent neural networks on sequence modeling". English (US). In: *NIPS 2014 Workshop on Deep Learning, December 2014*. 2014.

[17] Wikipedia. *Hyperparameter optimization — Wikipedia, The Free Encyclopedia*. http://en.wikipedia.org/w/index.php?title=Hyperparameter%20optimization&oldid=1002995616. [Online; accessed 31-January-2021]. 2021.

[18] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. "Learning representations by back-propagating errors". In: *Nature* 323.6088 (Oct. 1986), pp. 533–536. DOI: 10.1038/323533a0. URL: https://doi.org/10.1038/323533a0.

[19] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. "On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima". In: *CoRR* abs/1609.04836 (2016). arXiv: 1609.04836. URL: http://arxiv.org/abs/1609.04836.

[20] Ekaansh Khosla, Dharavath Ramesh, Rashmi Priya Sharma, and Samuel Nyakotey. "RNNs-RT: Flood based Prediction of Human and Animal deaths in Bihar using Recurrent Neural Networks and Regression Techniques". In: *Procedia Computer Science* 132 (2018), pp. 486–497. DOI: 10.1016/j.procs.2018.05.001. URL: https://doi.org/10.1016/j.procs.2018.05.001.

[21] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. "Dropout: a simple way to prevent neural networks from overfitting". In: *The journal of machine learning research* 15.1 (2014), pp. 1929–1958.

[22] John M. Goodkind. "The superconducting gravimeter". In: *Review of Scientific Instruments* 70.11 (Nov. 1999), pp. 4131–4152. DOI: 10.1063/1.1150092. URL: https://doi.org/10.1063/1.1150092.

[23] F. Greco, D. Carbone, F. Cannavò, A. A. Messina, and G. Siligato. "Absolute and Relative Gravity Measurements at Volcanoes: Current State and New

Developments Under the NEWTON-g Project". In: *International Association of Geodesy Symposia*. Springer Berlin Heidelberg, 2020. DOI: 10.1007/1345_2020_126. URL: https://doi.org/10.1007/1345_2020_126.

[24] Jacques Hinderer, David J. Crossley, and Richard J. Warburton. "Superconducting Gravimetry". In: *Treatise on Geophysics*. Elsevier, 2015, pp. 59–115. DOI: 10.1016/b978-0-444-53802-4.00062-2. URL: https://doi.org/10.1016/b978-0-444-53802-4.00062-2.

[25] GWR Instruments Inc. *Operating Principles of the Superconducting Gravity Meter*. URL: http://www.gwrinstruments.com/pdf/principles-of-operation.pdf. (accessed: 2020-09-21).

[26] Hans-Georg Scherneck. *A PowerPoint presentation, Introduction and Status August 2010*. http://holt.oso.chalmers.se/hgs/SCG/StatusAug2010.pdf. [Online; accessed 28-October-2020]. 2010.

[27] Tadahiro Sato, Kazunari Nawa, Kazuo Shibuya, Yoshiaki Tamura, Masatsugu Ooe, Katsutada Kaminuma, and Yuichi Aoyama. "Polar Motion Effect on Gravity Observed with a Superconducting Gravimeter at Syowa Station, Antarctica". In: *International Association of Geodesy Symposia*. Springer Berlin Heidelberg, 1997, pp. 99–106. DOI: 10.1007/978-3-662-03482-8_16. URL: https://doi.org/10.1007/978-3-662-03482-8_16.

[28] Severine Rosat, Yves Rogister, David Crossley, and Jacques Hinderer. "A search for the Slichter triplet with superconducting gravimeters: Impact of the density jump at the inner core boundary". In: *Journal of Geodynamics* 41.1-3 (Jan. 2006), pp. 296–306. DOI: 10.1016/j.jog.2005.08.033. URL: https://doi.org/10.1016/j.jog.2005.08.033.

[29] Martin Vallée, Jean Paul Ampuero, Kévin Juhel, Pascal Bernard, Jean-Paul Montagner, and Matteo Barsuglia. "Observations and modeling of the elastogravity signals preceding direct seismic waves". In: *Science* 358.6367 (Nov. 2017), pp. 1164–1168. DOI: 10.1126/science.aao0746. URL: https://doi.org/10.1126/science.aao0746.

[30] T. Klügel and H. Wziontek. "Correcting gravimeters and tiltmeters for atmospheric mass attraction using operational weather models". In: *Journal of Geodynamics* 48.3-5 (Dec. 2009), pp. 204–210. DOI: 10.1016/j.jog.2009.09.010. URL: https://doi.org/10.1016/j.jog.2009.09.010.

[31] Philip Brunner, Craig T. Simmons, Peter G. Cook, and René Therrien. "Modeling Surface Water-Groundwater Interaction with MODFLOW: Some Considerations". In: *Ground Water* 48.2 (Mar. 2010), pp. 174–180. DOI: 10.1111/j.1745-6584.2009.00644.x. URL: https://doi.org/10.1111/j.1745-6584.2009.00644.x.

[32] Takahito Kazama, Shuhei Okubo, Takayuki Sugano, Shigeo Matsumoto, Wenke Sun, Yoshiyuki Tanaka, and Etsuro Koyama. "Absolute gravity change associated with magma mass movement in the conduit of Asama Volcano (Central Japan), revealed by physical modeling of hydrological gravity disturbances". In: *Journal of Geophysical Research: Solid Earth* 120.2 (Feb. 2015), pp. 1263–1287. DOI: 10.1002/2014jb011563. URL: https://doi.org/10.1002/2014jb011563.

[33] Incorporated Research Institutions for Seismology. *Mww7.5 Minahassa Peninsula, Sulawesi.* http://ds.iris.edu/ds/nodes/dmc/tools/event/10953070. [Online; accessed 5-November-2020]. 2018.

[34] Jean-Paul Montagner, Kévin Juhel, Matteo Barsuglia, Jean Paul Ampuero, Eric Chassande-Mottin, Jan Harms, Bernard Whiting, Pascal Bernard, Eric Clévédé, and Philippe Lognonné. "Prompt gravity signal induced by the 2011 Tohoku-Oki earthquake". In: *Nature Communications* 7.1 (Nov. 2016). DOI: 10.1038/ncomms13349. URL: https://doi.org/10.1038/ncomms13349.

[35] Duncan Agnew. *Baytap08: A Program for Analyzing Tidal Data.* https://igppweb.ucsd.edu/~agnew/Baytap/baytap.html. [Online; accessed 5-November-2020]. 2008.

[36] Michel Van Camp and Paul Vauterin. "Tsoft: graphical and interactive software for the analysis of time series and Earth tides". In: *Computers & Geosciences* 31.5 (June 2005), pp. 631–640. DOI: 10.1016/j.cageo.2004.11.015. URL: https://doi.org/10.1016/j.cageo.2004.11.015.

[37] Florent Lyard, Fabien Lefevre, Thierry Letellier, and Olivier Francis. "Modelling the global ocean tides: modern insights from FES2004". In: *Ocean Dynamics* 56.5-6 (Sept. 2006), pp. 394–415. DOI: 10.1007/s10236-006-0086-x. URL: https://doi.org/10.1007/s10236-006-0086-x.

[38] E. A. Spiridonov and O. Yu. Vinogradova. "Atmospheric Loading Displacements". In: *Izvestiya, Atmospheric and Oceanic Physics* 55.11 (Dec. 2019), pp. 1814–1819. DOI: 10.1134/s0001433819110227. URL: https://doi.org/10.1134/s0001433819110227.

[39] *IERS - IERS - Earth orientation data.* URL: https://www.iers.org/IERS/EN/DataProducts/EarthOrientationData/eop.html (visited on 01/31/2021).

[40] Per-Anders Olsson, Kristian Breili, Vegard Ophaug, Holger Steffen, Mirjam Bilker-Koivula, Emil Nielsen, Tõnis Oja, and Ludger Timmen. "Postglacial gravity change in Fennoscandia—three decades of repeated absolute gravity observations". In: *Geophysical Journal International* 217.2 (Jan. 2019), pp. 1141–1156. DOI: 10.1093/gji/ggz054. URL: https://doi.org/10.1093/gji/ggz054.

[41] *pandas - Python Data Analysis Library.* URL: https://pandas.pydata.org/ (visited on 01/31/2021).

[42] *Keras: the Python deep learning API.* URL: https://keras.io/ (visited on 02/01/2021).

[43] David J. Crossley. "Comprehensive Analysis of 2 years of SG Data from Table Mountain Colorado." In: *Proc. 13th Int. Symp. on Earth Tides* (1997). URL: https://ci.nii.ac.jp/naid/10007393992/en/.

[44] Michael G. McDonald and Arlen W. Harbaugh. "The History of MODFLOW". In: *Ground Water* 41.2 (Mar. 2003), pp. 280–283. DOI: 10.1111/j.1745-6584.2003.tb02591.x. URL: https://doi.org/10.1111/j.1745-6584.2003.tb02591.x.

[45] Hui Zou and Trevor Hastie. "Regularization and variable selection via the elastic net". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 67.2 (Apr. 2005), pp. 301–320. DOI: 10.1111/j.1467-9868.

2005.00503.x. URL: https://doi.org/10.1111/j.1467-9868.2005.00503.x.

[46] Wikipedia. *Generalized normal distribution — Wikipedia, The Free Encyclopedia*. http://en.wikipedia.org/w/index.php?title=Generalized%20normal%20distribution&oldid=1000235907. [Online; accessed 25-January-2021]. 2021.

A

Appendix

A.1 Performance of models across hyperparameter space

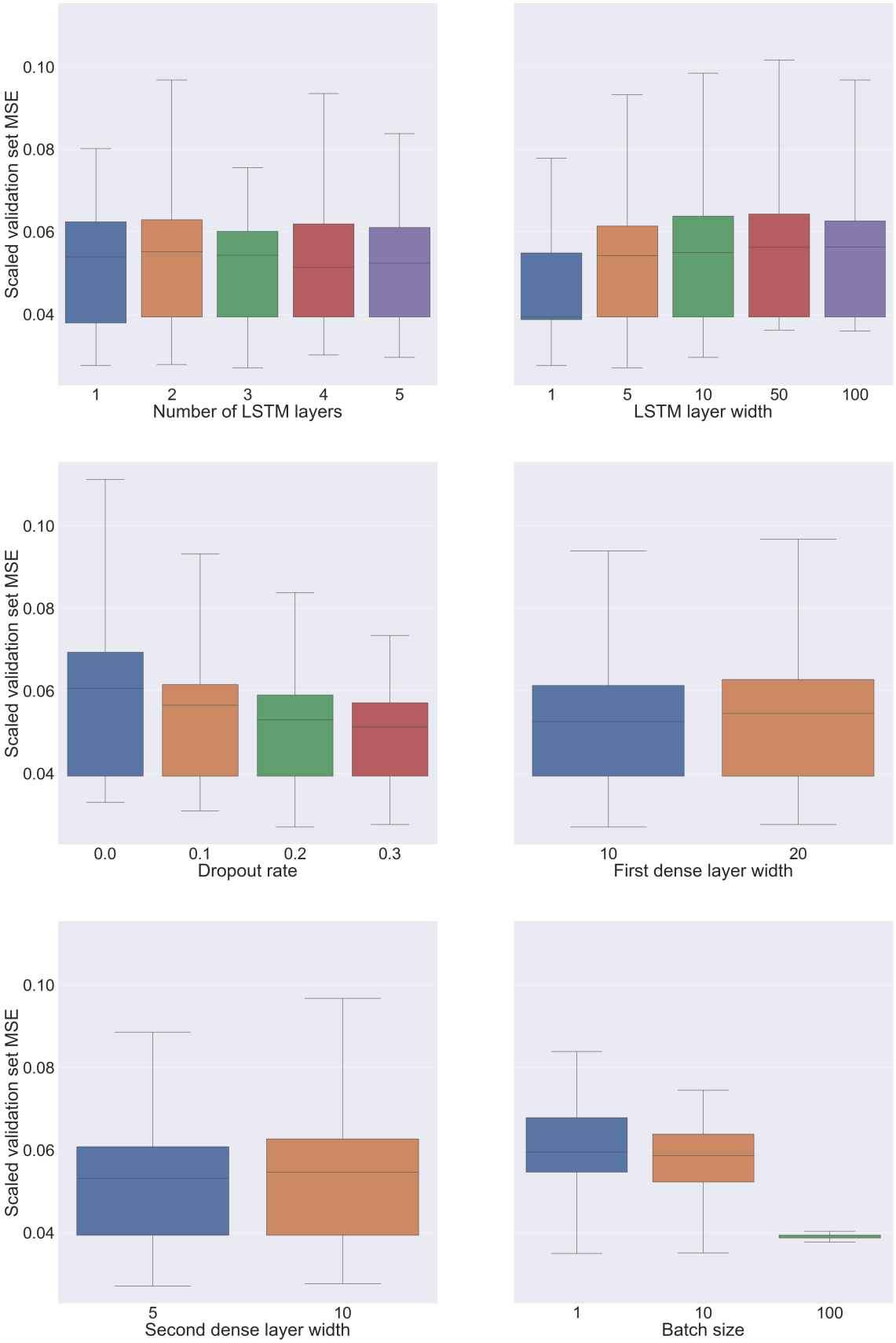


Figure A.1: Performance across explored hyperparameter space (rainfall only model)

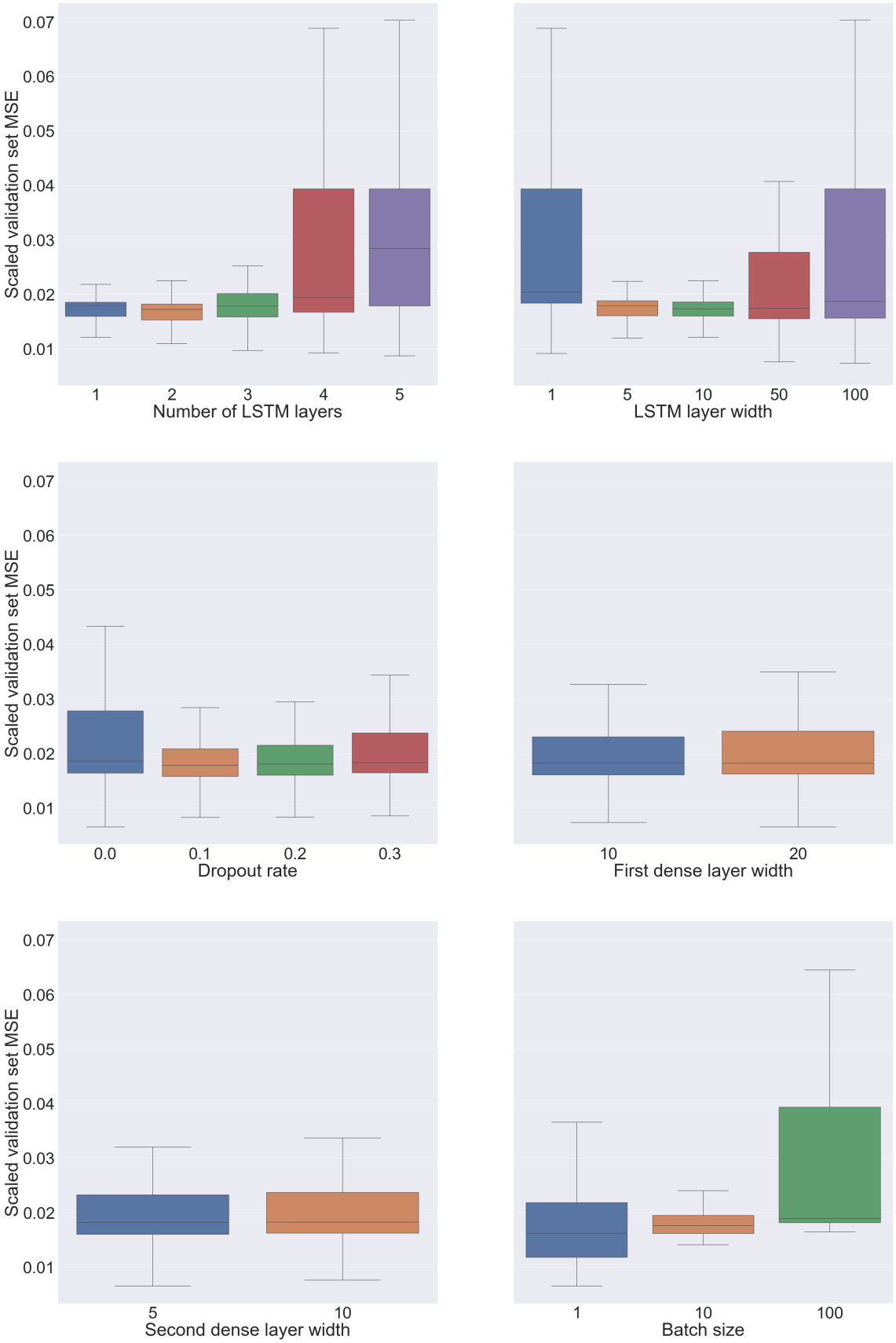


Figure A.2: Performance across explored hyperparameter space (sequential model using all available meteorological data)

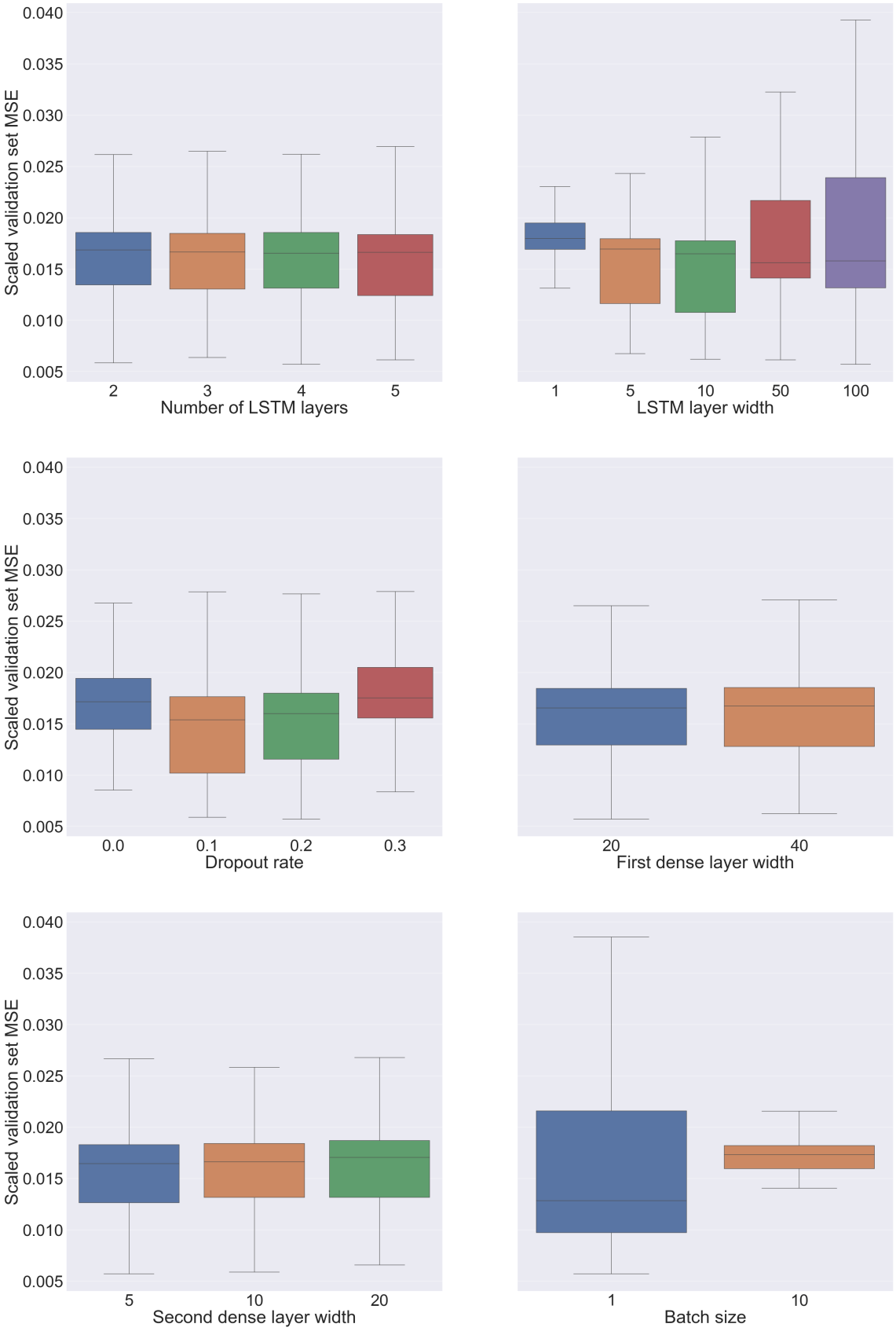


Figure A.3: Performance across explored hyperparameter space (non-sequential model using all available meteorological data)

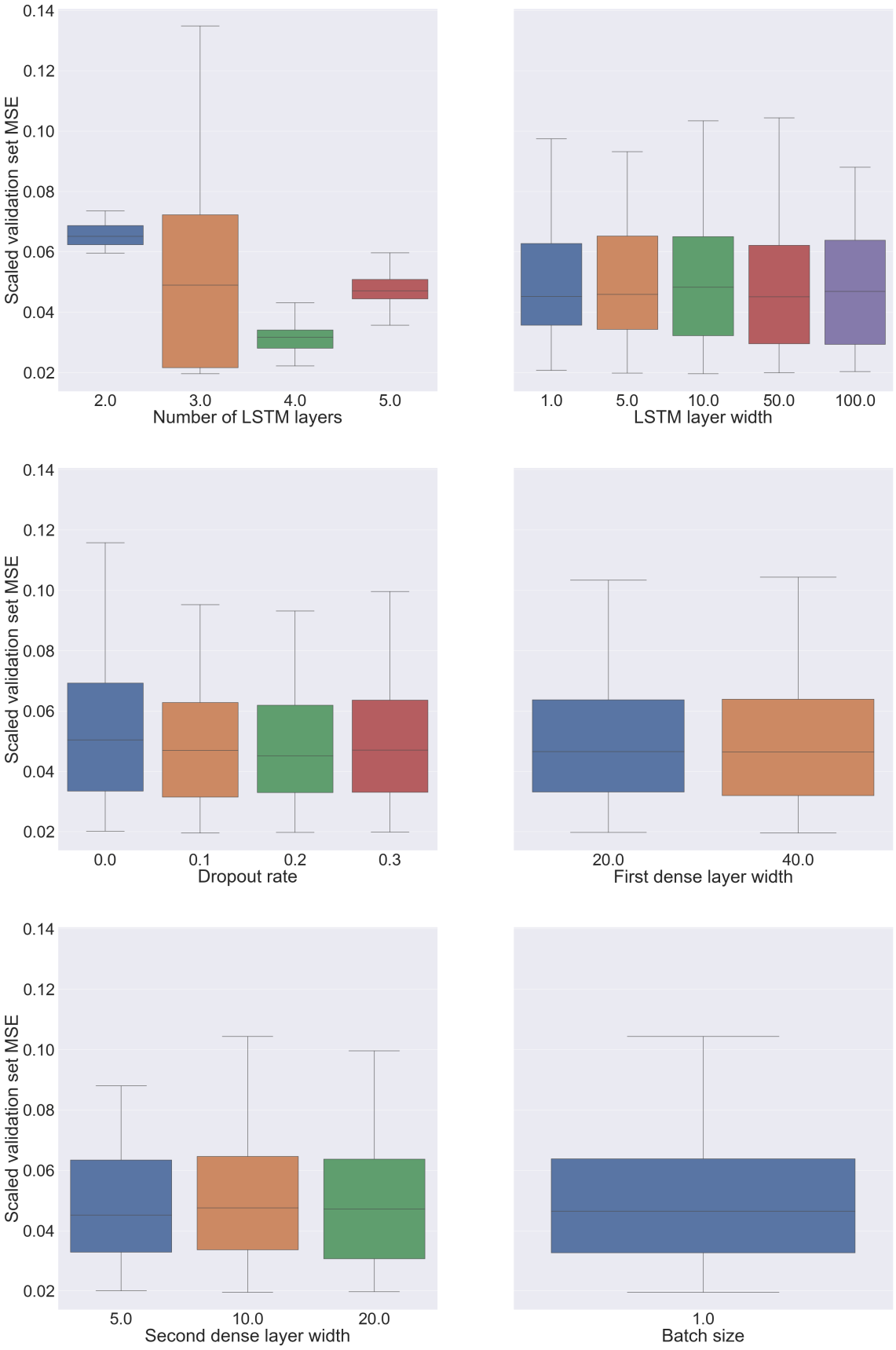


Figure A.4: Performance across explored hyperparameter space (non-sequential model with CNN preprocessing (constrained to non-negative values) using all available meteorological data.)

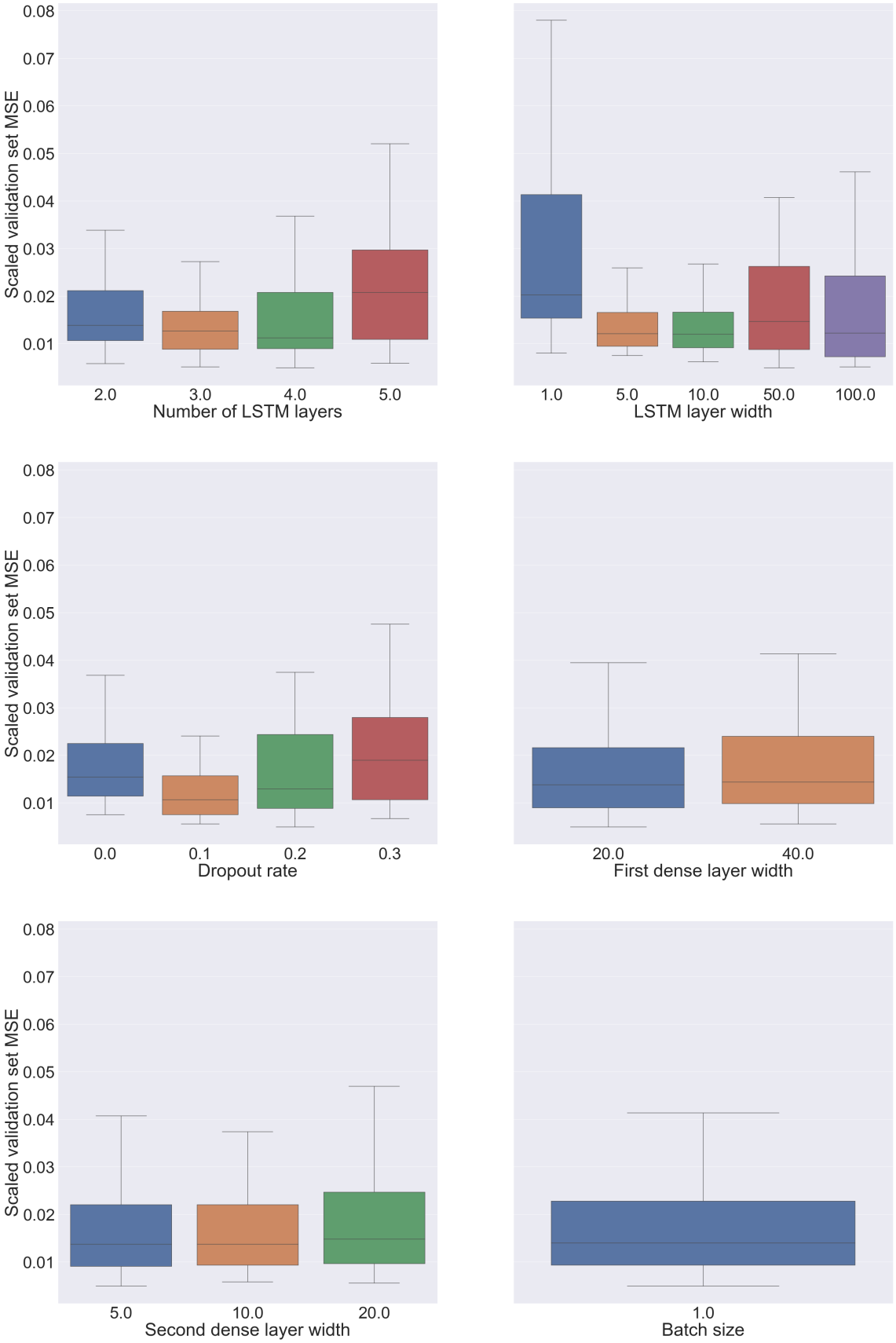


Figure A.5: Performance across explored hyperparameter space (non-sequential model with CNN preprocessing and current values, using all available meteorological data)

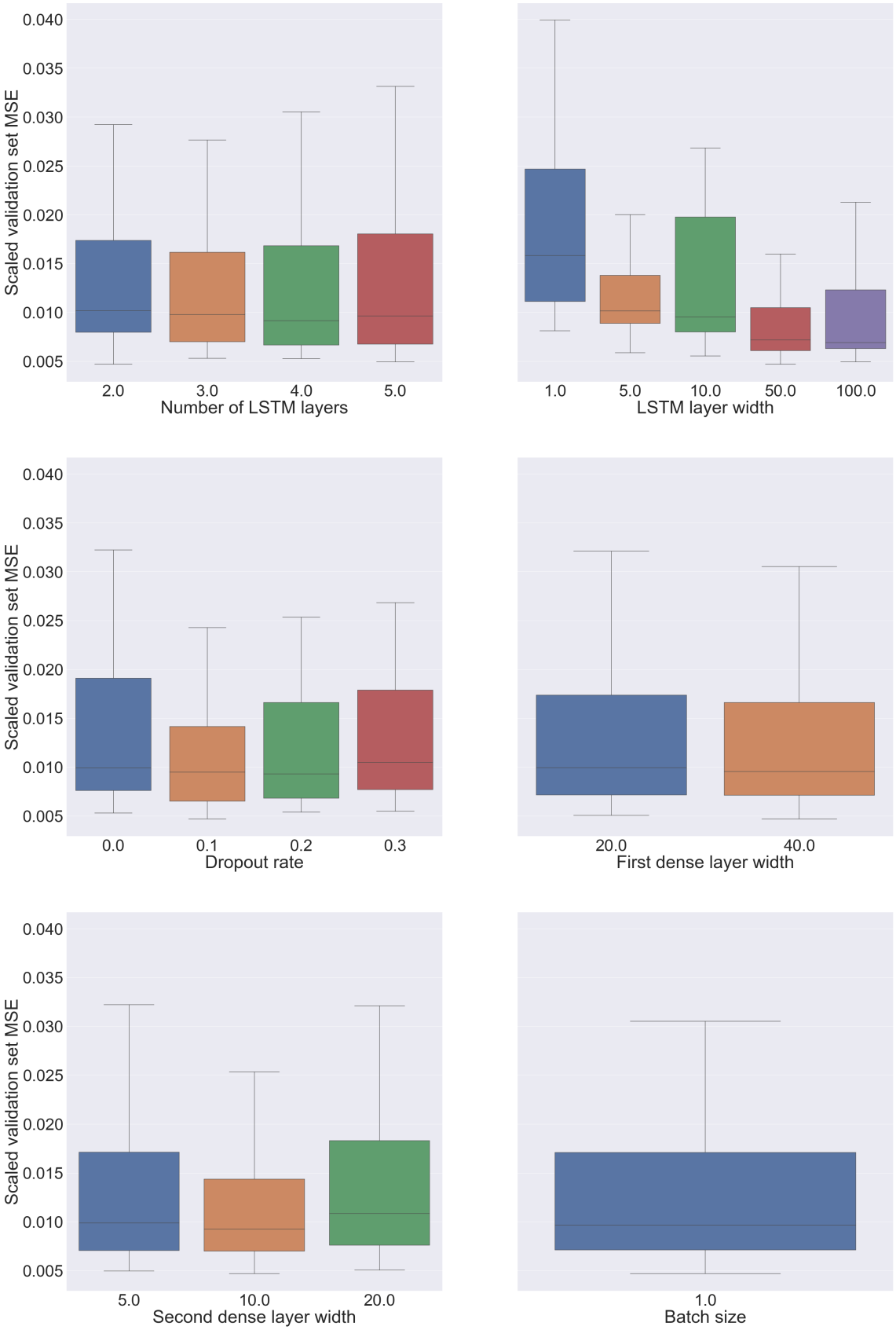


Figure A.6: Performance across explored hyperparameter space (non-sequential model with access to historical values using all available meteorological data)

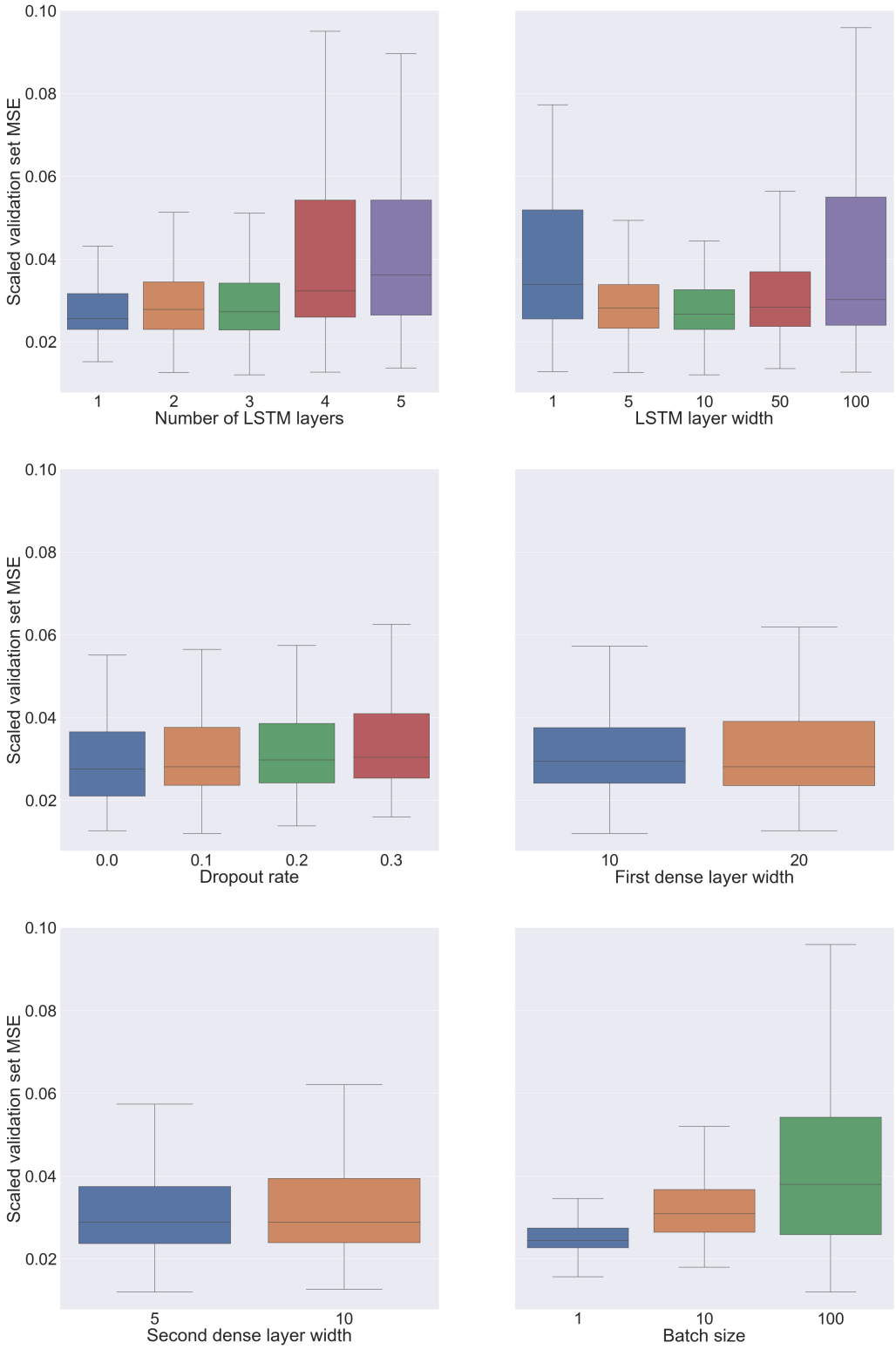


Figure A.7: Performance across explored hyperparameter space (gravity sequential model)

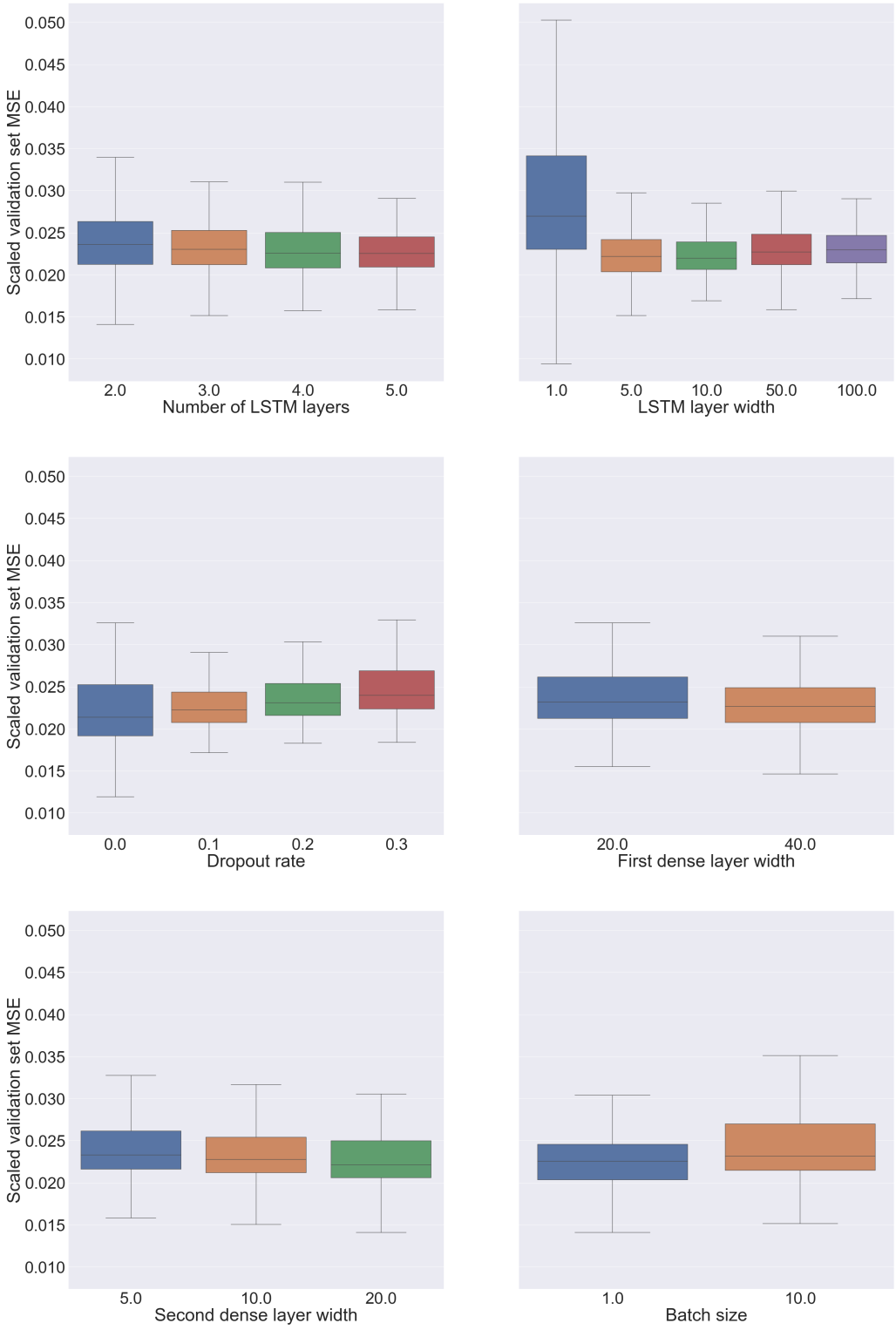


Figure A.8: Performance across explored hyperparameter space (gravity non-sequential model)

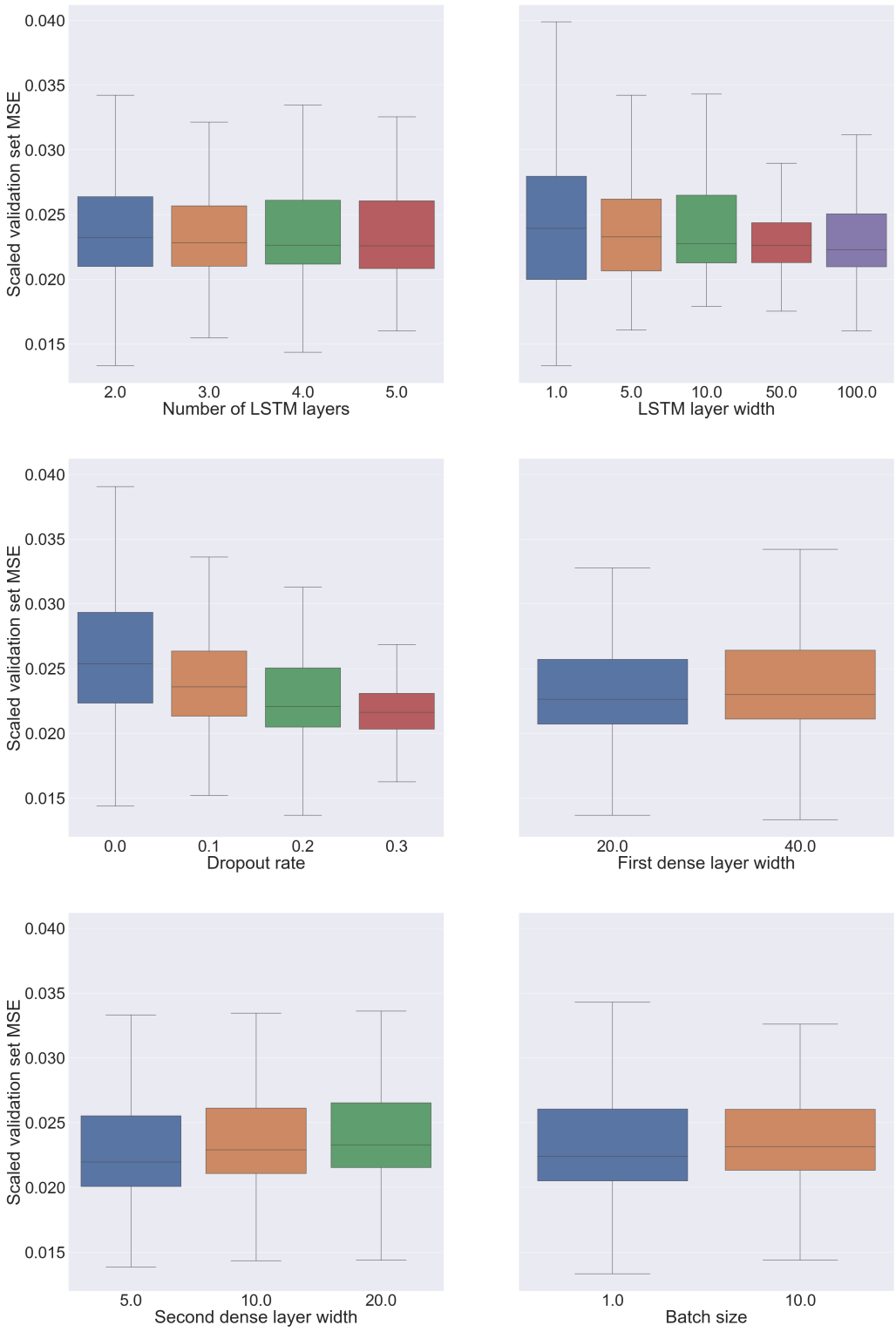


Figure A.9: Performance across explored hyperparameter space (gravity non-sequential model trained on data starting in 2010 including simulated groundwater measurements)

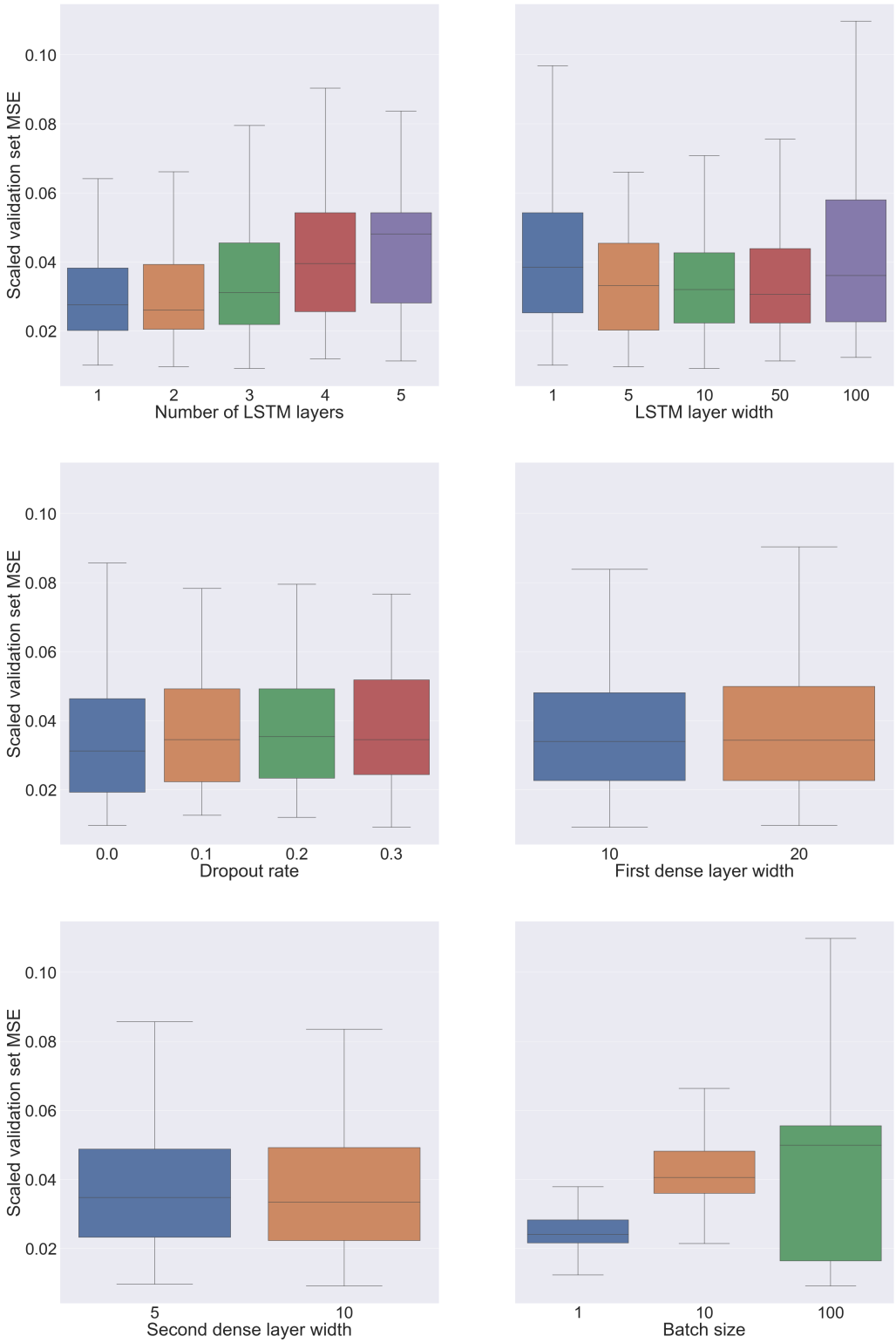


Figure A.10: Performance across explored hyperparameter space (gravity sequential model without temperature input)