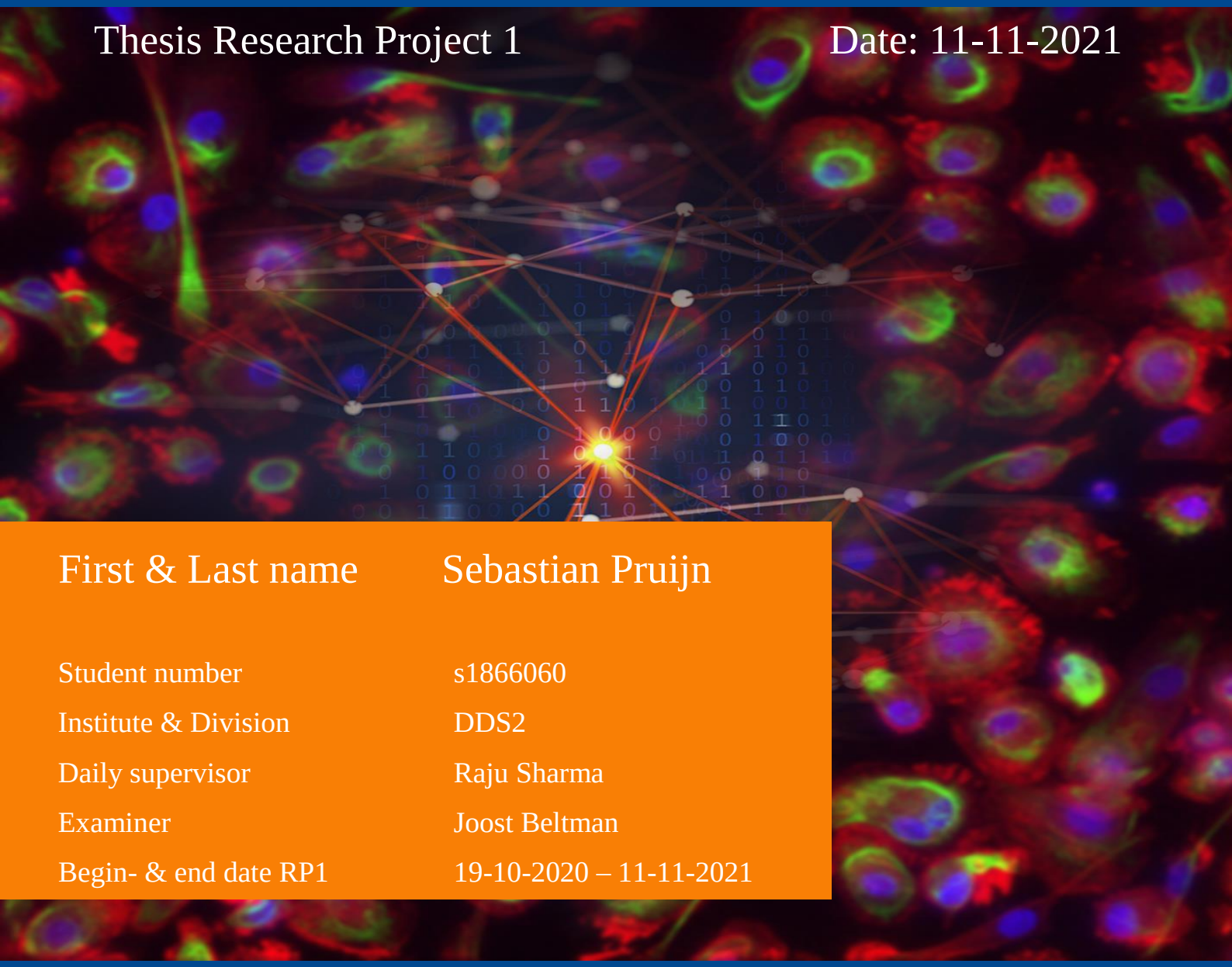
MSc Bio-Pharmaceutical Sciences

“Neural network approach to the quantitative study of the Oxidative Stress Response ”

Thesis Research Project 1

Date: 11-11-2021



First & Last name	Sebastian Pruijn
Student number	s1866060
Institute & Division	DDS2
Daily supervisor	Raju Sharma
Examiner	Joost Beltman
Begin- & end date RP1	19-10-2020 – 11-11-2021

BFW·BPS LACDR

Universiteit Leiden
The Netherlands

Discover the world at Leiden University

STATEMENT of ORIGINALITY

I hereby declare that this thesis is my own original work and confirm that:

- this work has been drafted by me without the use of any sources other than those explicitly cited in the text and acknowledgements
- I have clearly referenced all sources used in the work
- all data and findings in the work are original and have not been falsified or embellished.

I confirm that I take full responsibility of the contents of this work understand it will be checked for plagiarism by the use of plagiarism detection software.

Leiden University 11-11-2021	Sebastian Pruijn 
……	……
Place and date	Name and Signature

TABLE of CONTENTS

PREFACE

Predicting drug toxicity is a difficult task due to the vast complexity of the protein signalling networks that a drug may interact with. In this research a framework is presented for the quantitative study of protein signalling pathways using neural networks.

PUBLIC SUMMARY

Studying liver toxicity with neural networks

Cells function though very complex networks of protein interactions. In predicting drug toxicity it is necessary to study how drugs interact with these networks, something conventionally done with mathematical models. This research shows that this process can be partially automated using neural networks.

KEYWORDS

Neural networks, Nrf2, Srxn1, ODE, cell signalling

RESEARCH REPORT

ABSTRACT

Taking medications is not without its side effects, one of the most common being Drug Induced Liver Injury (DILI). DILI is mediated through various forms of stress, one of which being oxidative stress caused by an uptake in reactive metabolites. In response to oxidative stress cells activate the Oxidative Stress Response (OSR), within which an increase in Nrf2 leads to an upregulation of Srxn1. Though the OSR pathway is responsible for mitigating oxidative stress, high levels of drug induced OSR activation may lead to cellular toxicity. Understanding the mechanisms underlying the regulation of the OSR pathway and how compounds may interact with it is therefore vital for drug safety. This study has looked to machine learning as a potential alternative to the traditionally used ODE models for the study of the OSR pathway. The aims of this research were two-fold: the first was to investigate the potential of neural networks to learn the mechanism of the Nrf2-Srxn1 relationship. As GFP data is a relatively expensive form of data and error prone, the second objective was to investigate the potential to save on GFP data through learning the patterns within the data. This study has found that the neural networks had mediocre successes at learning compound exposure based Nrf2 dynamics using limited data. If a good fit could be attained however, the neural network could cast accurate predictions for single time point inputs. Through the use of artificial data the neural network has proven some capacity for learning an Nrf2-Srxn1 mechanism. Using this neural network to learn the Nrf2-Srxn1 mechanism of GFP data the neural network has hinted at preliminary evidence of a compound specific Nrf2-Srxn1 mechanism. This research has shown the potential of using neural networks for saving on GFP data through limited time point predictions, as well as providing an initial framework for a probabilistic approach to learning compound specific protein-protein mechanisms.

INTRODUCTION

When a drug enters the body it is often either activated or inactivated through metabolic processes that commonly occur within the liver. A potential side effect to these metabolic processes is liver damage or Drug Induced Liver Injury (DILI), a phenomenon responsible for many cases of acute liver failure and drug withdrawals from the market (1,2,3,4,5). One of the mechanisms through which DILI occurs is through Oxidative stress, a form of stress mediated through an increase in Reactive Oxygen Species (ROS) (6,7). These ROS have the capacity to damage various components of the cell such as the mitochondria or DNA, damage which in sufficient amounts will lead to cell death (8,9,10,11).

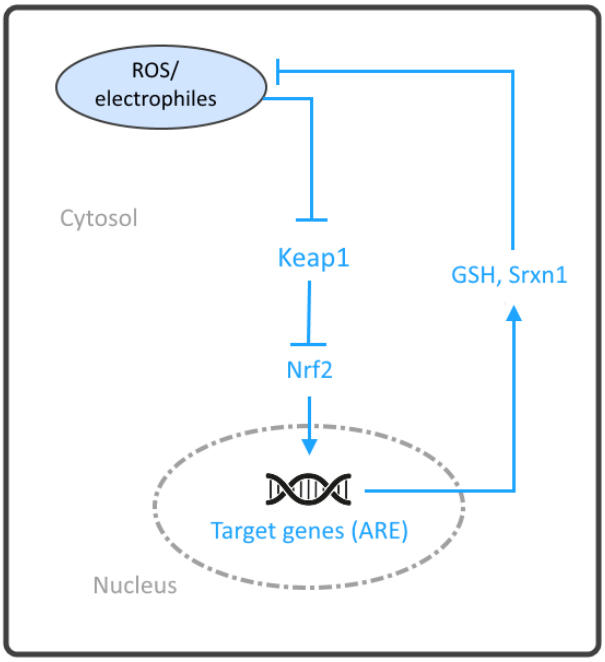


Figure 1: the Oxidative Stress Response pathway. Image adapted from Kuijper et al. (2017) (12).

As a response to this hazard cells can activate the Oxidative Stress Response (OSR) pathway. This activation is initialised upon an increase in ROS, generated by either a parent compound or its active metabolite, resulting in the inactivation of the protein Kelch-like ECH-associated protein 1 (Keap1)(13). Under normal conditions Keap1 is responsible for the sequestering and ubiquitination of nuclear factor erythroid 2–related factor 2 (Nrf2), resulting in degradation of Nrf2 by a proteasome (14,15,16). The ROS induced inactivation of Keap1 therefore has the consequence of halting the inhibition and sequestering of Nrf2, which in turn results in a rise in cellular Nrf2 concentration and an increase in Nrf2 nuclear translocation (17,18). In the nucleus, the subsequent activation of the Antioxidant Response Element by Nrf2 leads to an upregulation of antioxidant proteins like Sulfiredoxin-1 (Srxn1) responsible for maintaining a balanced redox status in the cell (19). Thus in essence Nrf2 protects the cell against oxidative stress which makes it an interesting drug target. This also leads to an interest in Nrf2 upregulating compounds like Sulforaphane as its cytoprotective effects could have significant health benefits (20). Upregulation of Nrf2 following compound exposure is far from universally beneficial however as whilst it can serve a cytoprotective function, it could also indicate the presence of heightened oxidative stress or even have tumour promoting effects (21).

That there are benefits and drawbacks to compound mediated Nrf2 upregulation serves as a prime example of the nuances and challenges in predicting DILI. In the vast complexity of signalling pathways in the body, there are my potential components a compound could either directly bind to or have an indirect interaction with. With these many interactions there are many routes through which a compound may damage and/or protect the cell. In drug development it is therefore essential for drug safety to have an understanding of the many potential interactions that a compound may have with the components of the cell and on the wider signalling pathways within the cell. In the case of the OSR pathway Nrf2 knockdown experiments have revealed that the Srxn1 expression is largely mediated by Nrf2, yet compound interactions with regulatory proteins or crosstalks with other stress response pathways may allow for different mechanisms underlying this Nrf2 mediated Srxn1 expression (22,23).

One way through which a signalling pathway such as the OSR pathway and its Nrf2-Srxn1 relationship can be described is through the use of Ordinary Differential Equation (ODE) modelling (24,25,26). Within ODE models the way that protein concentrations change over time and proteins interact with one another are described through ODEs. The ODE models can therefore be seen as a mathematical representation of the quantitative protein interactions within a system which is used for casting predictions of the change within that system. Though ODE models can become an impressive representation of the biological reality given sufficient complexity and parametrization, they can be quite time and labour intensive to develop. This is in part due to the difficulty of formulating and parameterizing ODEs for complex biological pathways. In reducing the time and labour required for designing these ODE models, the application of machine learning may provide a viable answer.

In the last decade the use of machine learning methods has seen a strong rise in usage due to computational power becoming cheaper, the amount of data rising exponentially worldwide and machine learning tools becoming more accessible (27). One common machine learning approach is that of neural networks which as the name implies consists of a network of “neurons”; functions that transform an input value into an output value. These neurons are then interlinked through parameters called “weights”. In training a neural network the weights of the network are optimized in such a way that the input values to the network are transformed into the desired output. In a properly fitted neural network, the network has learned the underlying pattern of how an input is transformed into an output. It was hypothesised that this pattern learning ability of neural networks could prove useful in studying the OSR pathway.

One way in which this neural network ability could prove useful is in the learning of the Nrf2 dynamics following compound exposure. As these dynamics are often observed from Green Fluorescent Protein (GFP) imaging data, there is a cost associated with observing the full range of a dose-response relationship in the form of the funds, time and labour required for the experiments. It was hypothesised that using neural networks this cost could be reduced by having the network learn the dose-response pattern on a scarce amount of data.
Aside from saving on data through the learning of Nrf2 dynamics, another neural network application may be the study of protein-protein relationships. Using Nrf2 as an input to the neural network and Srxn1 as an output, the neural network may be able to learn how an increase in Nrf2 quantifies an increase in Srxn1 over a specific amount of time. It was believed that a neural network capable of understanding how proteins relate to one another akin to the functioning of an ODE model may then prove useful as a tool for the evaluation of ODE models and as such aid in their development.

Moreover when compound specific GFP time series data is used, the learned Nrf2-Srxn1 mechanism would be based on the Nrf2-Srxn1 relationship as described by the data of a specific compound. If the Nrf2 data of another compound was then to be used as input data, the Srxn1 prediction by the neural network would follow the mechanism of the compound the network was trained on. A comparison between the Srxn1 prediction of the network and the Srxn1 data of another compound could then reveal to what degree compounds share in the same Nrf2-Srxn1 mechanism. As such it was hypothesised that a learned Nrf2-Srxn1 mechanism by a neural network could be used for the study of the role of compound specificity in the Nrf2-Srxn1 pathway.

In order to test these hypotheses a suitable neural network architecture needs to be considered. One neural network architecture that excels at learning time series data like GFP data is the Long-Short term Memory (LSTM) architecture (28). What sets this type of neural network apart from others is its built in memory component which allows the network to remember and use past data for its future predictions. In predicting protein dynamics or protein-protein mechanisms this ability may prove useful as it increases the amount of information based on which the network can cast a prediction.

The aim of this research was the study of the OSR pathway using neural networks, which was performed in two parts: First was the evaluation of the neural network's ability to learn the Nrf2 expression dynamics following compound exposure. Second was the study of the neural network's ability to learn a complex Nrf2-Srxn1 mechanism and to subsequently apply this mechanism to the study of the role of compound specificity in the Nrf2-Srxn1 relationship.
To achieve these aims multiple neural network configurations were applied through which the role of the LSTM's memory was tested as well as the importance of Srxn1 as an input value for the learning of a proper Nrf2-Srxn1 mechanism. Using limited amounts of Nrf2 data a neural network was trained to predict Nrf2 dynamics for varying doses of Sulforaphane resulting in learned patterns of mixed accuracy. For the testing of a learned Nrf2-Srxn1 mechanism, ODE data was generated that allowed for the testing of whether a complex Nrf2-Srxn1 relationship could be learned by the neural network. These ODE tests along with other artificial data tests accrued evidence that the neural network configuration used could learn a more complex form of the Nrf2-Srxn1 relationship. Based on this evidence compound specificity tests were performed using Sulforaphane, Andrographolide and CDDO data resulting in potential evidence that Sulforaphane and CDDO follow different Nrf2-Srxn1 mechanisms.

MATERIALS and METHODS

Data preparation and processing

The data used in this study is acquired from an imaging experiment performed in the LACDR-DDS. Experiments were conducted on HEPG2-GFP reporter cell lines for an Nrf2 response measured in nuclear and a Srxn1 response in cytoplasm. Andrographolide, Sulforaphane and CDDO-me were administered once at 0h to induce the Nrf2 response. Images were taken each hour for 32 hours to follow the reporter intensities over time. This resulted in 32h intensity-time profiles of Nrf2 and Srxn1.

Programming was performed in Python version 3.7. Full model code and neural networks versions are available on Github (see Appendix 1.). Data transformations, neural network operations and statistical tests were performed using the following packages: Numpy, Pandas, Sklearn and Scipy, Tensorflow, Keras and Pylab. Figures have been plotted using the Matplotlib package. 0-1 normalization was performed using the MinMaxScaler function from the Sklearn package (sklearn.preprocessing.MinMaxScaler). This scaling function has been set to a feature range of a minimum of 0 and a maximum of 1 and is from here referred to as '0-1scaler'.

The 32 hour imaging data were interpolated to the length of 2000 data points using an interpolation function (scipy.interpolate.interp1d). For each dose the data vectors were smoothed a first time using a Savitzky-Golay filter (polynomial value 5, window value 9) from the Scipy python package (scipy.signal.savgol_filter) (29).

For each dose of the compound exposure data the upper and lower replicate limit were calculated in the following way:
In a loop, for every iteration *i* along the replicate length (*len*) the value *r* of each replicate vector at index *i* was taken and collected in list *R*.

$$R_i = \{r_1, \ . \ . \ . \ , r_n\}$$

The upper replicate range was assembled by taking the maximum value of list *R* at every iteration.

$$UI_i = \max(R_i)$$

Subsequently assembled into list *upper*

$$upper = \{UI_1, \ . \ . \ . \ , UI_{len}\}$$

The lower replicate range was assembled by taking the minimum value of list *R* at every iteration.

$$LI_i = \min(R_i)$$

Subsequently assembled into a list *lower*

$$lower = \{LI_1, \ . \ . \ . \ , LI_{len}\}$$

The centre of the replicate range was subsequently generated as vector *Rmean*. This was done through the calculation of the mean between the upper and lower ranges for each index.

$$Rmean_i = \frac{upper_i + \ lower_i}{2}$$

Rmean vectors were subsequently smoothed using the Savitzky-Golay filter using Polynomial value 5 and Window value 1501.

Time and dose value input data generation
A 0-1 normalized time vector was prepared for 0-32 hours, which was used in all training configurations. This vector was prepared in a loop of *len* iterations. For each iteration a time point was generated as:

$$tp = \frac{32}{len} * i$$

In which each time point is defined as *tp* and generated on each iteration *i*. Each *tp* was added to a time vector *timevec* in order of generation.

$$timevec = \{tp_1, \ \ldots, tp_{len}\}$$

Lastly the *timevec* was 0-1 normalized using the 0-1scaler set to the minimum value of 0 and the maximum value of 96.

For the Dose-Nrf2 neural network a dose vector *dosevec* was generated. This was done by first fitting the 0-1scaler to the minimum of $\log_{10}(0.1)$ and the maximum of $\log_{10}(40)$. The desired compound exposure dose value was then 0-1 normalized using the fitted 0-1scaler to produce the normalized dose value *dv*. The *dosevec* of length *len* was assembled with value *dv* along its entire length.

$$dosevec = \{dv_1, \ \ldots, dv_{len}\}$$

Training configurations

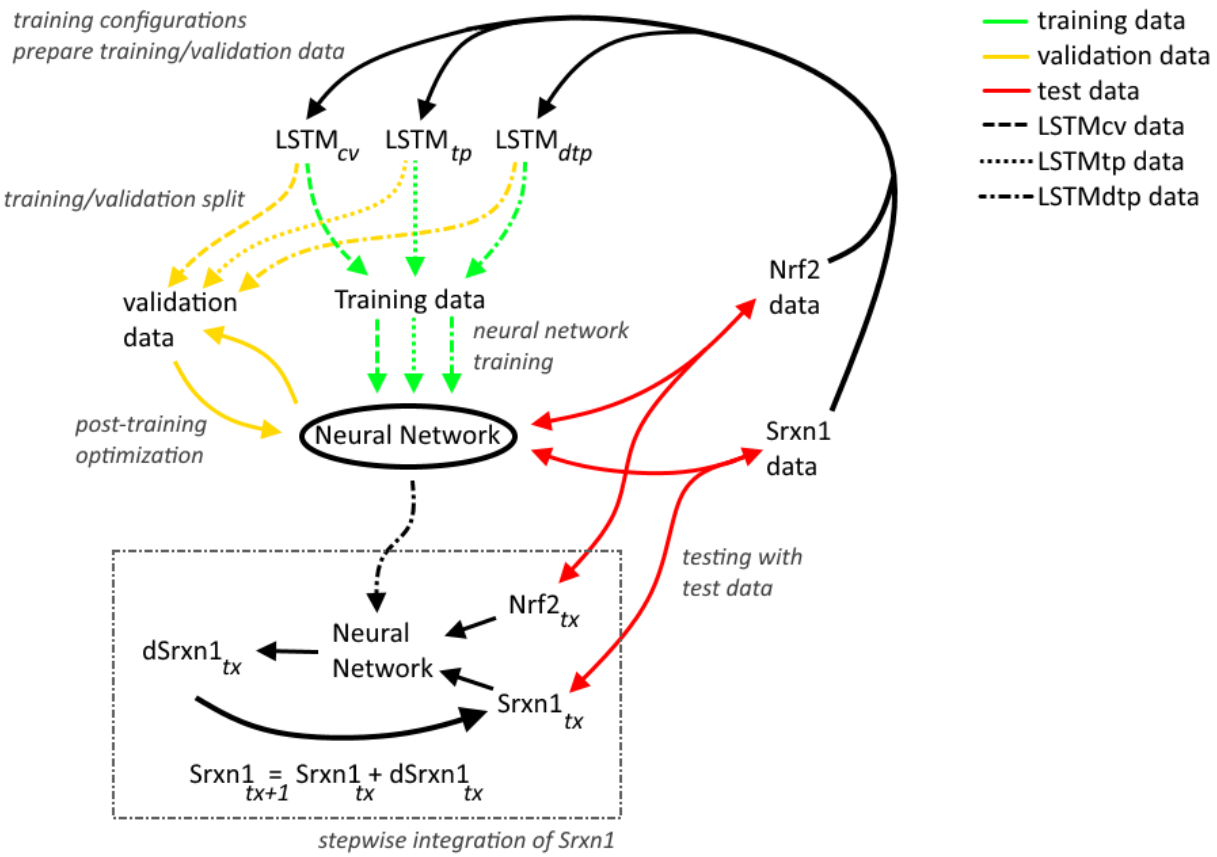


Figure 2: Model framework. Arrows are indicative of the data flows within the model. Protein data is transformed into training and validation data using either the $LSTM_{cv}$, $LSTM_{tp}$ or $LSTM_{dtp}$ training configurations. Training data is used by the neural network for parameter optimization within the network. Validation data is (optionally) used to select the hyperparameters that result in the best training fit by which overfitting on the training data is prevented. Finally test data is used for final testing to see whether the neural network has learned the proper patterns underlying the data.

For the neural network training process the data was transformed using training configurations. A more detailed description is given in Appendix 4. In these training configurations the data points *dp* consisted of a time value, dose value or protein value depending on the test performed. 0-1 normalization for Nrf2, Srxn1 and dSrxn1 has been performed within these training configurations using specific 0-1scaler settings found in appendix 3.
For the $LSTM_{cv}$ configuration each neural network sample contains a full vector of length *len* with data points *dp*.

$$Sample\ x = \{dp_1, \ldots, dp_{len}\}$$

For the $LSTM_{tp}$ and $LSTM_{dtp}$ configurations each neural network sample contains a single data point.

$$Sample\ x = dp$$

The $LSTM_{dtp}$ configuration calculates derivative values from the selected protein data, which were smoothed using the Savitzky-Golay filter (polynomial value 5, window length 1501). Using this configuration the derivative predictions of the neural network were integrated using an integrator function which is described in detail in appendix 5.

Neural network hyperparameters

The neural network seeds *S* used to achieve varying training fits have been set using the tensorflow.random.set_seed(*S*) function of the Tensorflow package. Alternative seeds have been set as constant values during training to improve reproducibility. These seeds can be found in the imports of the code on the Github. Test specific settings can be found in appendix 1.

The layers LSTM and Dense and the Sequential function have been imported from the Keras package. These layers were added by calling the Sequential.add function. The loss function and optimizer were set using the Sequential.compile function. Finally the training itself was initiated by calling the Sequential.fit function, giving the prepared input tensor for *x* and the prepared output tensor for *y*.

The $LSTM_{cv}$ configurations use the following neural network hyperparameters:
Presented as: (layer number, layer type, amount of neurons, activation function)

1. LSTM 20, tanh activation
2. LSTM 10, tanh activation
3. LSTM 1, tanh activation
4. LSTM 1, tanh activation
5. Dense 1

Batch size = 1000
Loss = Mean Absolute Error (MAE)
Optimizer = ‘Adam’

The $LSTM_{tp}$ and $LSTM_{dtp}$ configurations use the following neural network hyperparameters:

1. LSTM 100, tanh activation
2. LSTM 100, tanh activation
3. LSTM 100, tanh activation
4. LSTM 100, tanh activation
5. Dense 1

Batch size = 1000
Loss = Mean Absolute Error (MAE)
Optimizer = ‘Adam’

ODE data generation
ODE data was generated using a premade Oxidative Stress Response ODE model, listed in appendix 6. Listed below are the most essential ODEs with their respective parameter changes shown in table 1. The stress parameter S was altered from value 1 to value 4 to generate Nrf2 and Srxn1 training data. Validation data was generated by altering the logarithmic value *x* of the vMax parameter.

$$dS = -time\_constant1 * S$$
$$dRM = buildRM - confForm * Gsh * RM - degradRM * RM + S$$
$$dSrxn1 = buildSrxn1base + buildSrxn1 * \frac{nNrf2^{hill_{Srxn1}}}{K_{Srxn1}{}^{hill_{Srxn1}} + nNrf2^{hill_{Srxn1}}} - degradSrnx1 * Srxn1$$

$$dNrf2 = exportNrf2 * nNrf2 - importNrf2 * Nrf2 + buildNrf2Base - \frac{vMax * Nrf2}{km * \frac{1+RM}{kiRM} + Nrf2}$$

	S	vMax (e+*x*)
Train 1	1	08
Train 2	2	08
Train 3	3	08
Train 4	4	08
Test 1	1	06
Test 2	1	05

Table 1. Modified ODE parameters.

Simple artificial data tests
The following artificial Nrf2 data were prepared for a *len* of 2000 data points. These data were not 0-1 normalized during the process of data generation as this step was performed in the integrator.

Steady state tests (fig. 14a):
Multiple empty vectors were generated and filled with a single steady state value. These values are 0.025, 0.1, 0.2, 0.4 and 0.8.

Negative exponential Nrf2 test (fig. 14b):
The following formula was used in a loop with 2000 iterations in which the value of iteration x ranges from 0-1999. Each data point *dp* in this loop was collected in a vector.

$$dp = \frac{-1}{2 * 10^6} * (x + 1)^2 + 2$$

Positive exponential Nrf2 test (fig. 14c):
The following formula was used in a loop with 2000 iterations in which the value of iteration x ranges from 0-1999. Each data point *dp* in this loop was collected in a vector.

$$dp = \frac{1}{2 * 10^6} * (x + 1)^2$$

Steady state multiple exposure test (fig. 14d):
A vector was generated of 2000 data points in which the first 1000 data points contained the value 0.025 and the last 1000 data points contained the value 0.8.

Quantitative compound comparison t-test
For the quantitative compound comparison a t-test was performed between the CDDO MSE values and the SUL MSE values using the ttest_ind function from the Scipy package (Scipy.stats.ttest_ind).

RESULTS

$LSTM_{cv}$ compound-Nrf2 neural network attained poor fit on higher test data

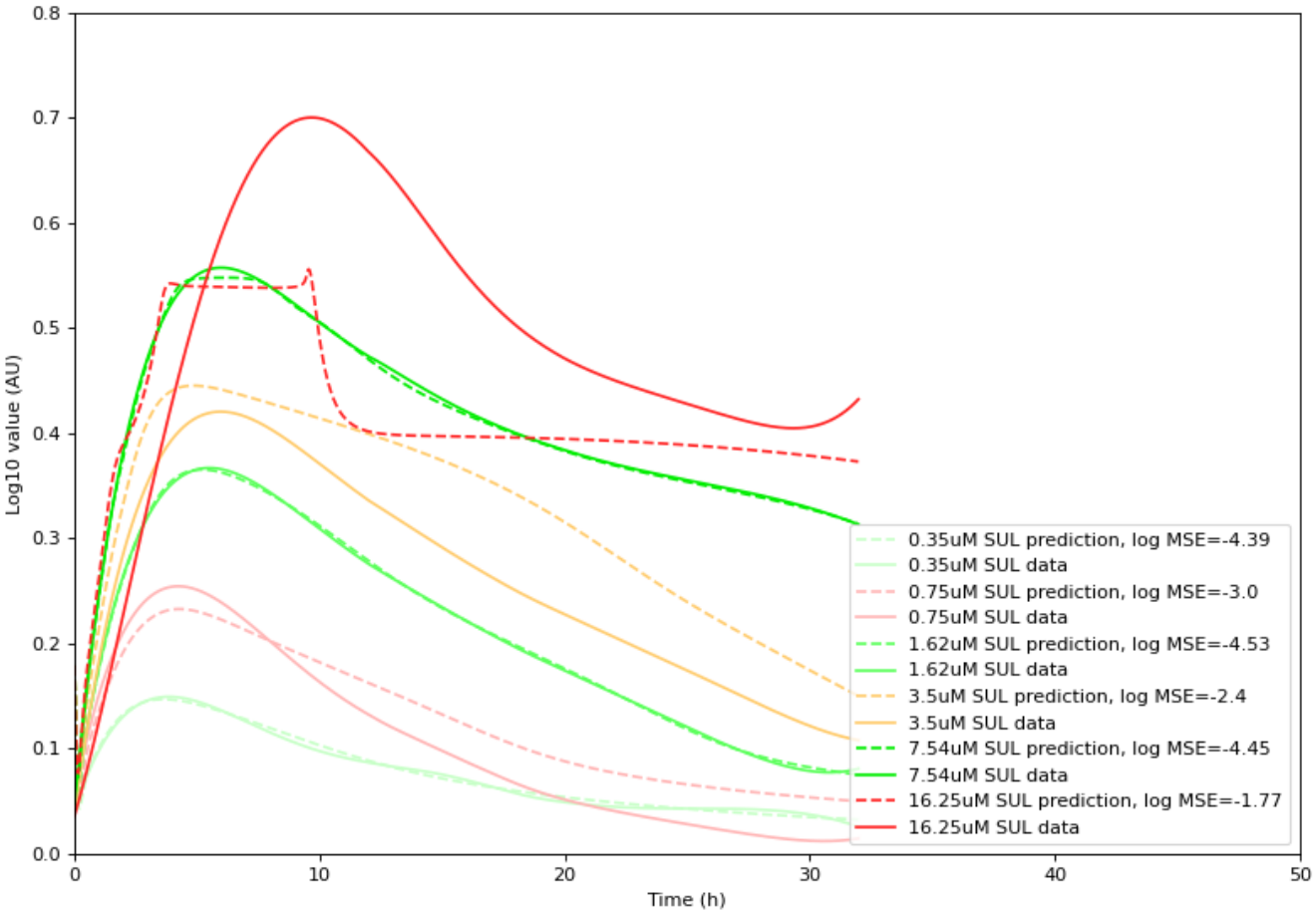


Figure 3. $LSTM_{cv}$ neural network fit on SUL Nrf2 data. Input variable include time and SUL exposure concentration. Green lines indicate training data, orange lines indicate validation data used for post training optimization and red lines indicate test data. Solid and dashed lines represent Nrf2 data and neural network predictions respectively.

The first neural network application that was tested was on the prediction of Nrf2 dynamics using limited data. This training was performed using the conventional training configuration ($LSTM_{cv}$), in which every sample to the neural network includes the full time range of values allowing for the utilization of the LSTM's memory. Input values to the neural network included time and the compound exposure concentration or 'dose', which the neural network transformed into Nrf2 values as an output.

Using the $LSTM_{cv}$ configuration the neural network was trained on 0.35 µM, 1.62 µM, and 7.54 µM SUL, using 3.5 µM SUL as validation data (figure 2). The neural network was able to attain a very good fit on the training data, with mixed results on all validation and test doses. Of note is the inability of the neural network to cast predictions above the highest value of the training data, in which case the prediction will plateau at the highest value of the training data. As it was hypothesised that this fault was due to an incompatibility of the conventional LSTM training configuration with the linear input data, another training configuration was tested that did not utilize the LSTM's memory component.

$LSTM_{tp}$ fit is dependent on training data similarity

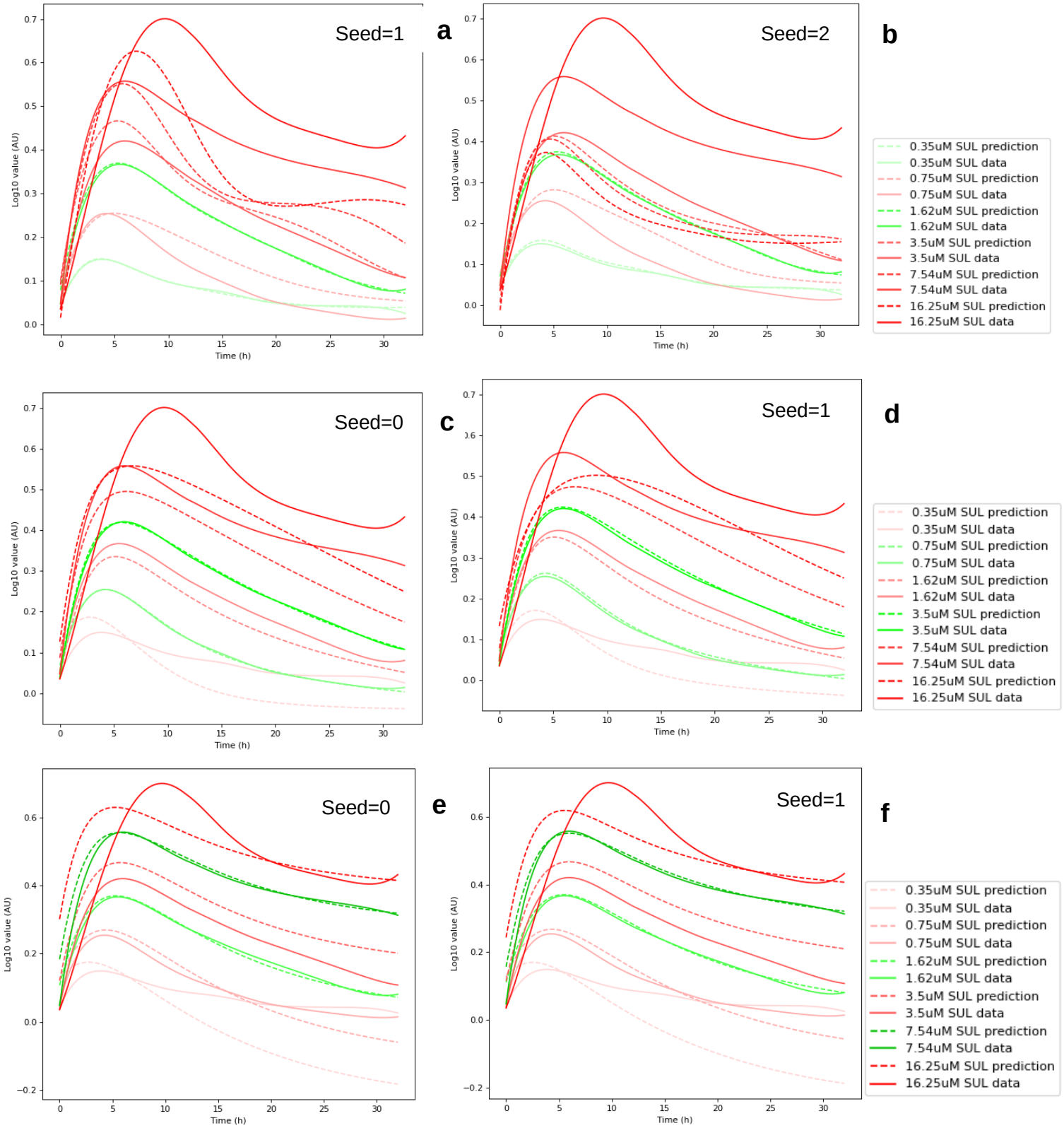


Figure 4. $LSTM_{tp}$ neural network fit on SUL Nrf2 data. a-b) fit on 0.35 µM and 1.62 µM SUL as training data. c-d) fit on 0.75 µM and 3.5 µM SUL as training data. e-f) fit on 1.62 µM and 7.54 µM SUL as training data. 'Seed' refers to both the seed used for the neural network's training process and its implicit stochastic effects on the training process.

The alternative training configuration used will be referred to as the time point LSTM configuration ($LSTM_{tp}$). This configuration takes individual data points as samples, effectively nullifying the LSTM's memory. The $LSTM_{tp}$ neural network has been trained on varying dose pairs, the results of which can be seen in figure 4 (training doses listed in green). Additionally training has been performed on varying neural network seeds resulting in a variety of learned patterns for the same training conditions. Training data was limited to two doses as to test the potential in capturing a full range of dose based dynamics using limited data. The $LSTM_{tp}$ trained neural network has shown improvements in regards to upward dose based extrapolations as higher predictions do not share in the plateauing effect showcased by the $LSTM_{tp}$ configuration. Of note is that in all cases, the single dose used for interpolation between the training doses perfectly falls in the centre of the range between those training doses. In regards to extrapolation the neural network achieves a better fit when the training data is similar in shape (fig. 3e-f) compared to training data that differs in shape (fig. 3a-b). The similarity in shape between the training doses therefore appear to influence the overall dynamics learned by the model.

$LSTM_{tp}$ allows for limited data tests

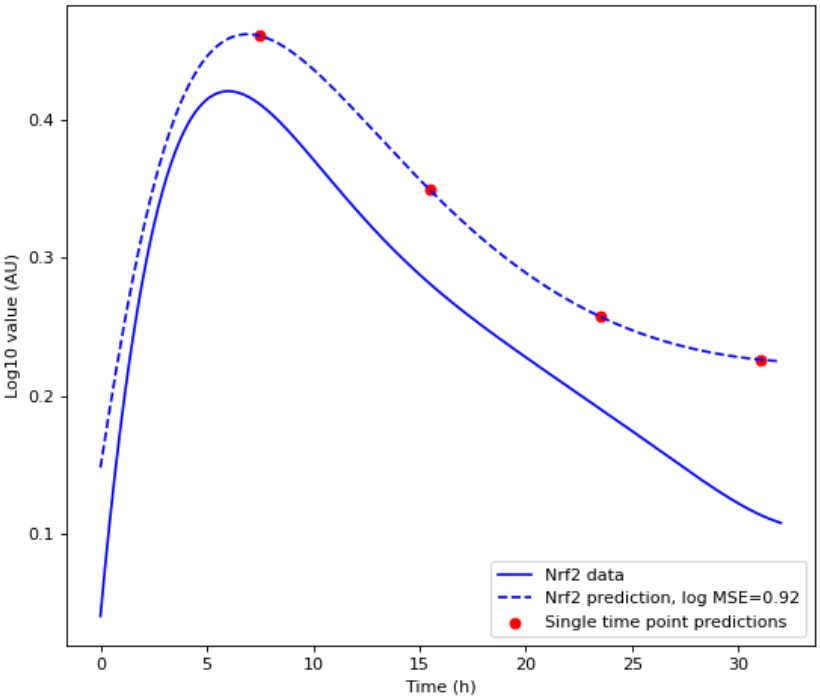


Figure 5. Single time point predictions of the $LSTM_{tp}$ neural network. Comparison between 3.5uM SUL prediction (dashed) and 3.5uM SUL data (solid) shown in blue. Data points in red show Nrf2 predictions for input tensors consisting of a single time and dose value of which the time values consisted of 0-1 normalized 7.5, 15.5, 23.5 and 31 hour values respectively per input tensor and dose values were constant at a 0-1 normalized value of 3.5uM.

An alternative way of saving on the generation of experimental data through neural networks is through single time point predictions. If an experiment requires the value of Nrf2 at a specific time and for a specific dose, a neural network trained on a compound dose-Nrf2 relationship might replace the need for the generation of new lab data. To test whether the $LSTM_{tp}$ configuration would be compatible with these types of tests a set of limited data tests were performed (fig. 5). These limited data tests have shown that the prediction accuracy of limited data inputs align perfectly with the prediction for a full 0-32 hour data vector, indicating that the accuracy is solely dependent on the training fit of the neural network.

$LSTM_{cv}$ Neural network is able to learn SUL Srxn1 dynamics

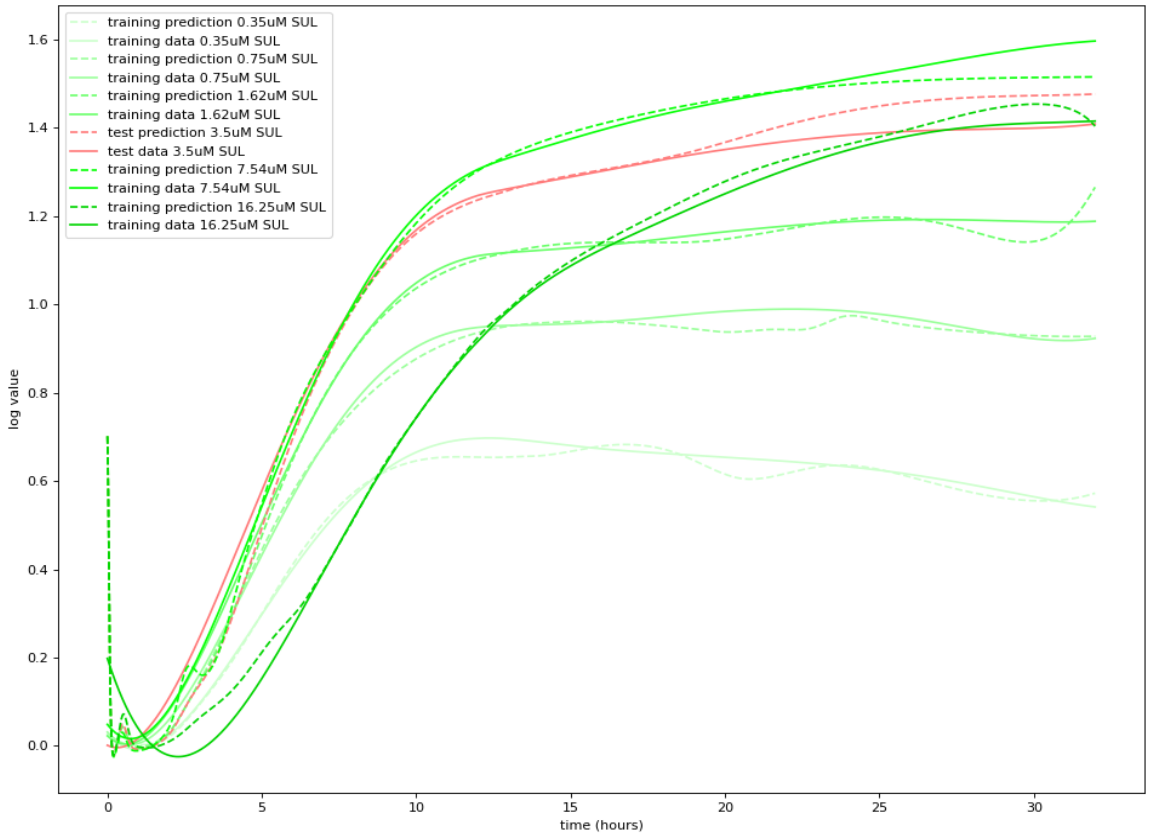


Figure 6. $LSTM_{cv}$ fit on Srxn1 data. Input variables include time and Nrf2 data. Green lines indicate training data and red lines indicate test data. Solid and dashed lines represent Srxn1 data and corresponding neural network predictions respectively.

Aside from using a neural network to study dose-Nrf2 dynamics, another potentially useful application is that of studying protein-protein relationships such as the Nrf2-Srxn1 relationship. A first attempt at studying this Nrf2-Srxn1 relationship has been done using the $LSTM_{cv}$ configuration, with time and Nrf2 as input values and Srxn1 as output values (fig. 6). Though the $LSTM_{cv}$ configuration had proven to be suboptimal in the dose-Nrf2 neural network, it was theorized that the more detailed Nrf2 input data would prove more valuable to the LSTM's memory as opposed to the linear data of the time and dose vector inputs of the dose-Nrf2 neural network.

Looking at the results it can be observed that both the training and test data predictions have good fits on the SUL Srxn1 data. However it is noticeable that even though the predictions follow the macro patterns of the respective data, there is a decent amount of noise present in these predictions. This is likely the result of the memory component that LSTM neural networks possess which may put emphasis on trying to learning patterns within the data, even when there are few. Slight amounts of noise within the data may be learned by the LSTM's memory and subsequently exacerbated.
Additionally the prediction of 16.25 µM SUL has a substantially higher initial value than would be expected. This is likely to be an indirect consequence of smoothing and log scaling the data, as these two factors have led to higher initial values than were present before these data transformations. The observed spike in initial value of the 16.25 µM prediction could then be an exaggeration of the increased initial value of the 16.25 µM data as a consequence of the data transformations.

$LSTM_{cv}$ neural network cannot learn artificial data mechanism

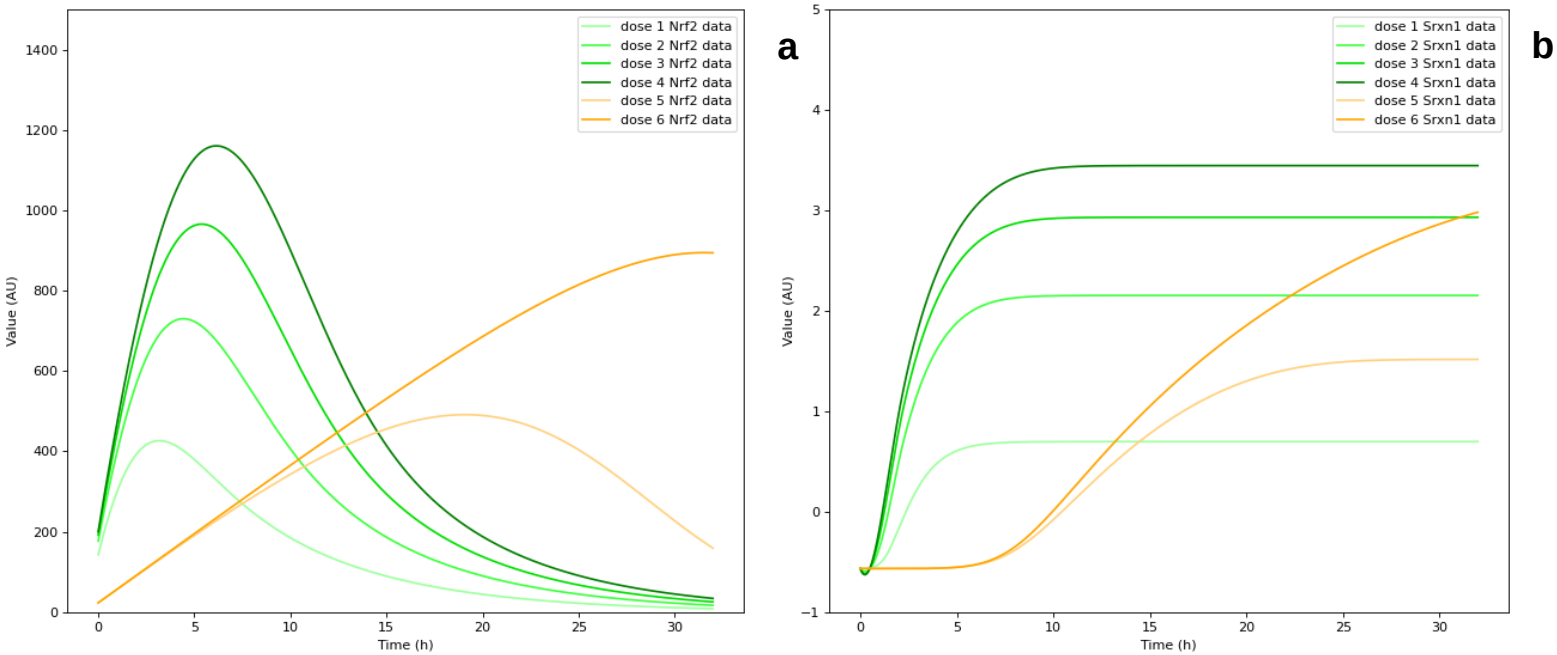


Figure 7. OSR ODE model generated data a) Artificially generated Nrf2 data with training data (green) and validation data (orange). b) ODE generated Srxn1 data with training data (green) and validation data (orange).

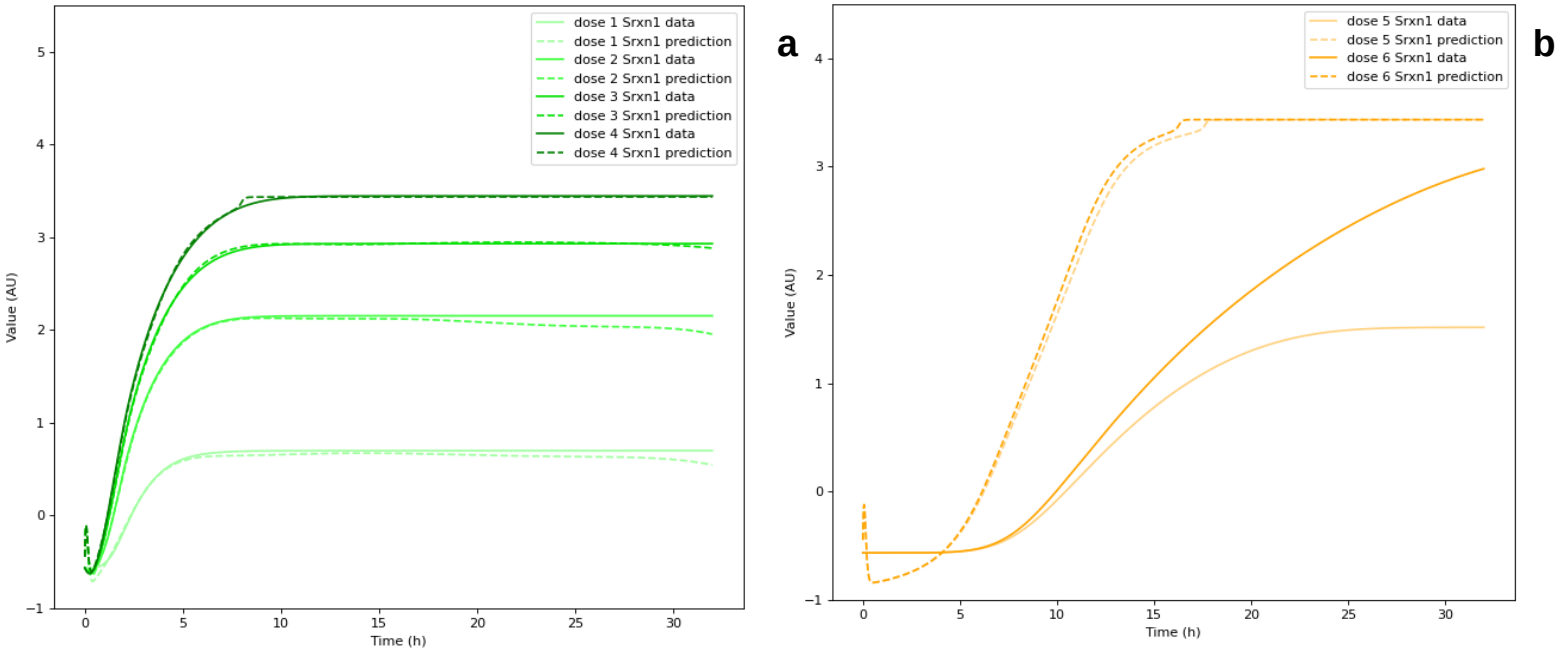


Figure 8. $LSTM_{cv}$ neural network fit on artificial Srxn1 data a) Neural network prediction fit on the Srxn1 training data of the ODE generated data**.** b) Neural network prediction fit on the Srxn1 validation data of the ODE generated data.

Using the $LSTM_{cv}$ configuration it was able to learn the dynamics of the Nrf2-Srxn1 relationship. However this tells little of how well the neural network has learned the underlying Nrf2-Srxn1 mechanism. To test whether the neural network could learn the true Nrf2-Srxn1 relationship of a compound as opposed to merely mimicking the dynamics of the training data, it was theorised that the neural network learning a Srxn1 Ordinary Differential Equation (ODE) would equate to it being able to learn a mechanism as an ODE could pose as an adequate representation of a mechanism.

To this end artificial data was generated using an OSR ODE model. Within this model the effect of varying compound exposure concentrations was mimicked by generating outputs for varying stress values, the results of which can be seen as the green lines in figure 7. These data which resembled the compound exposure data of Nrf2 and Srxn1 were used as training data for the neural network. In order to validate whether the neural network had learned the Srxn1 ODE, artificial data was generated using the same Srxn1 ODE but regulated by different Nrf2 dynamics as an Nrf2 ODE with lowered *vMax* values was used. This altered parameter led to a slower degradation of Nrf2 being simulated by the ODE model. This subsequently resulted in Nrf2 dynamics that had a strong deviation from the dynamics of the training data as it had more gradual gradients and later peak values. The Srxn1 output that resulted from the different Nrf2 dynamics also deviated strongly from the training data as it either plateaued 20 hours later or not at all.

The $LSTM_{cv}$ was able to learn the dynamics of the training data (fig. 8a) yet had a bad fit to the validation data as the fit on the validation data shows a repeat of the training data dynamics (fig 8b). As with the SUL fit (fig. 6) there appears to be noise present in the predictions, most notably around the initial values.
One notable difference between the training fit and the validation fit is that the validation fit shows a delay in the time to reach the plateauing phase of the training dynamic, reaching a plateau around 15 hours for the validation fit compared to around 6-9 hours for the training data fit. Additionally there is a delay in the time it takes the validation fit to reach the steepest rise in value, the gradient of which is lower than that of the training fits that reach a similar max value. This reveals that the validation fit does adapt to the validation data to some degree. However there is little difference between the two predictions of the validation data despite the validation data having a notable difference in shape. This could indicate that the validation fits shown are a demarcation of the $LSTM_{cv}$ neural network's ability to deviate from the training data fit.

$LSTM_{cv}$ configuration training did not learn proper time dynamics

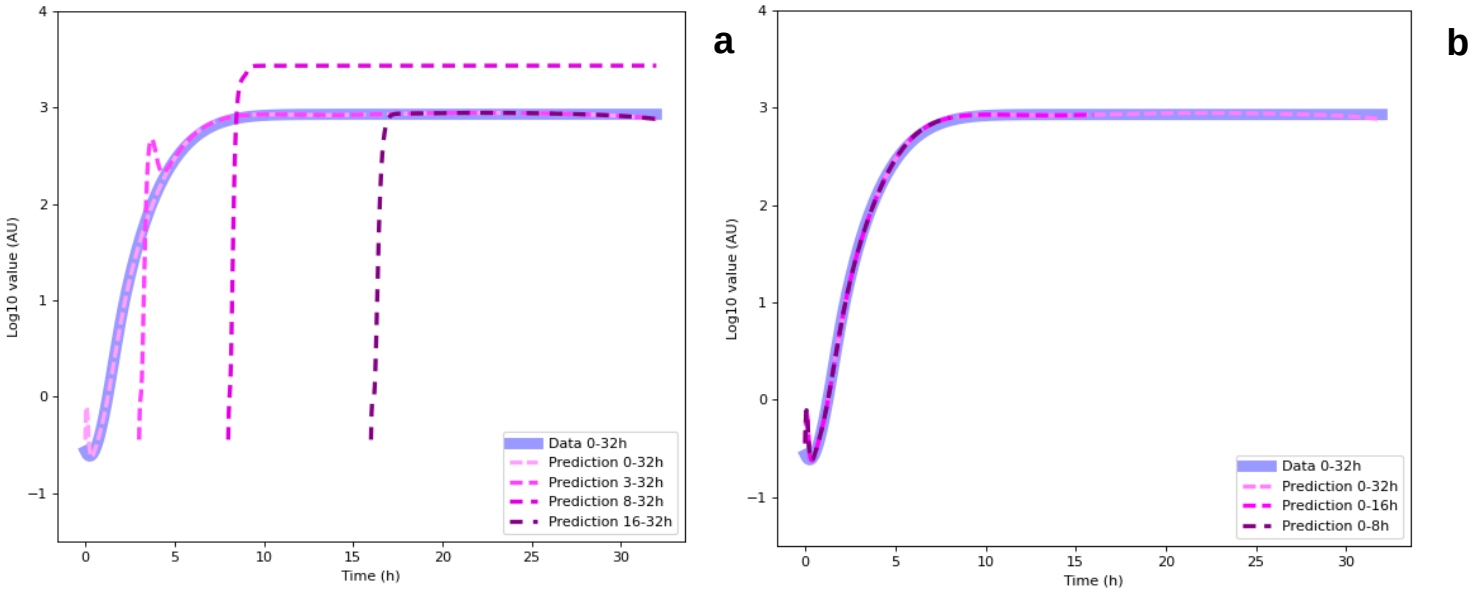


Figure 9. $LSTM_{cv}$ configuration ODE time tests. a) predictions for varying starting times (purple) compared to the data (blue). b) predictions for varying end times (purple) compared to the data (blue).

One benefit of this neural network approach could be casting accurate predictions for varying timeframes. An example of this application could include Nrf2 data ranging from 4-32 hours with the 0-4 hour data missing and a neural network trained on 0-32 hours subsequently casting an accurate Srxn1 prediction for the 4-32 hour timeframe. Tests were therefore performed using the ODE generated data to discover whether the $LSTM_{cv}$ configuration could cast accurate Srxn1 predictions for timeframes deviating from the 0-32 hour training timeframe.

These tests were performed by generating time vectors for varying time frames. Emphasis was placed on timeframes with varying initial time points but the same end time point. Timeframes with the same initial time point but varying end time points were used as a control. The corresponding Nrf2 data for these timeframes were added to the tensors with the time vectors and used as input data. The resulting predictions are shown in figure 9. in which it is observed that there were no good fits for input data timeframes that deviated from the training data. Predictions for different time frames show some resemblance to that of the actual data as after an hour the predictions will follow the proper trajectory. The initial values of the predictions however appear to remain unchanged. Of note are the predictions for 3-32 hours which shows an under prediction followed by an overshoot, after which it attains a good fit to the data, and the prediction for 8-32 hours which attains a good qualitative fit after the initial hour but has an overshoot in value. It is observed that the $LSTM_{cv}$ training has difficulty with initial data points, yet will quickly correct. The time component is therefore partially functioning.

$LSTM_{tp}$ configuration could not learn mechanism yet allows for Nrf2-Srxn1 limited data predictions

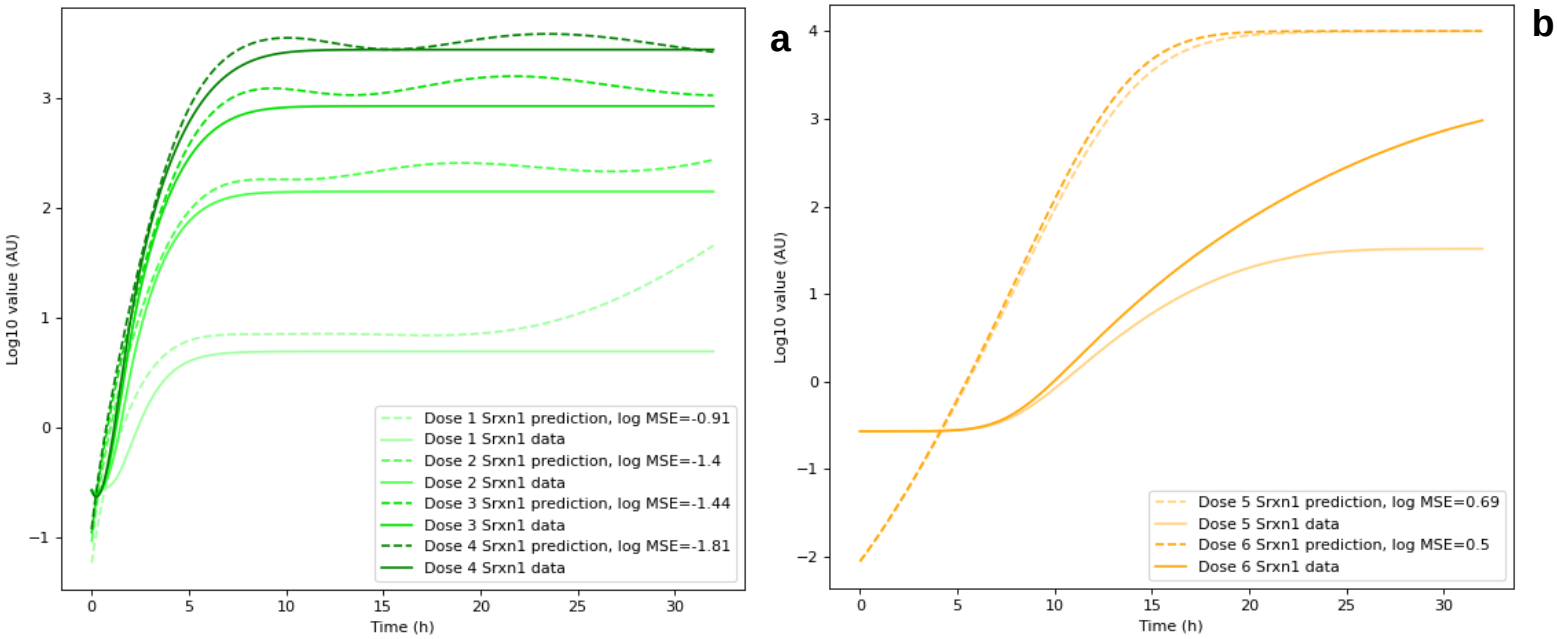


Figure 10. $LSTM_{tp}$ configuration ODE data fit a) Neural network Srxn1 prediction (dashed lines) fit on the training data (solid lines). b) Neural network Srxn1 prediction (dashed lines) fit on the validation data (solid lines)

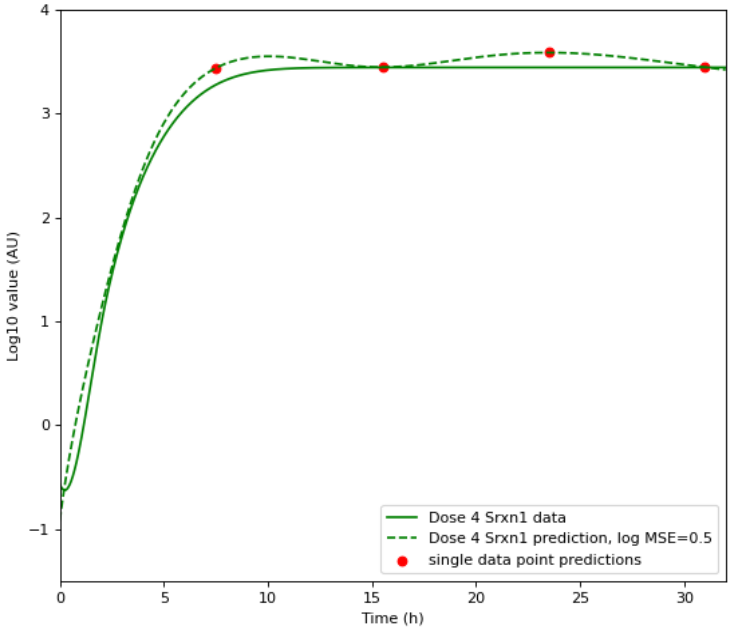


Figure 11. $LSTM_{tp}$ training Srnx1 predictions Solid line represents the Srxn1 ODE data, dashed line represents the corresponding Srxn1 prediction. Red dots are predictions for a single time and Nrf2 input of their respective values at 7.5, 15.5, 23.5 and 31 hours.

From the results in figure 8b it was hypothesised that the $LSTM_{cv}$ configuration could not learn the Nrf2-Srxn1 mechanism due to the memory component of the neural network's LSTM cells. It was likely that the memory component biased past data as opposed to present time data. Seeing as the Srxn1 ODE predicts future values based on the current state of a system, having an element that looks at the past state of a system may hinder the neural network's capacity to learn the ODE. Subsequent tests were therefore performed using the $LSTM_{tp}$ configuration, for which time and Nrf2 were used as input data and Srxn1 was used as output data. As can be seen in figure 10a the neural network learned the Nrf2-Srxn1 dynamics up to a decent accuracy, with higher doses generally having a better fit. The validation data had a poor fit however, being very similar in shape to the validation fit of the $LSTM_{cv}$ configuration (fig. 8b) yet without noise.

Similar to the dose-Nrf2 $LSTM_{tp}$ (fig. 5) this configuration allows for limited time point predictions as are seen in figure 11, using a single time point and a single Nrf2 data point as an input. As with the aforementioned neural network, the accuracy of these predictions are dependent on the training fit. All in all whilst a decent training fit could be achieved, the validation fit again has shown a similar fit to that of the $LSTM_{cv}$ configuration indicating that the LSTM's memory component is not solely at fault for the neural network not learning the ODE's mechanism. Using the $LSTM_{tp}$ configuration again allowed for limited time point predictions of which the accuracy appears to be solely determined by the fit on the training data. The $LSTM_{tp}$ configuration therefore has the benefit of flexible usage as it does not require a multi hour timeframe worth of data in order to cast predictions.

Derivative time point configuration can learn Srxn1 ODE

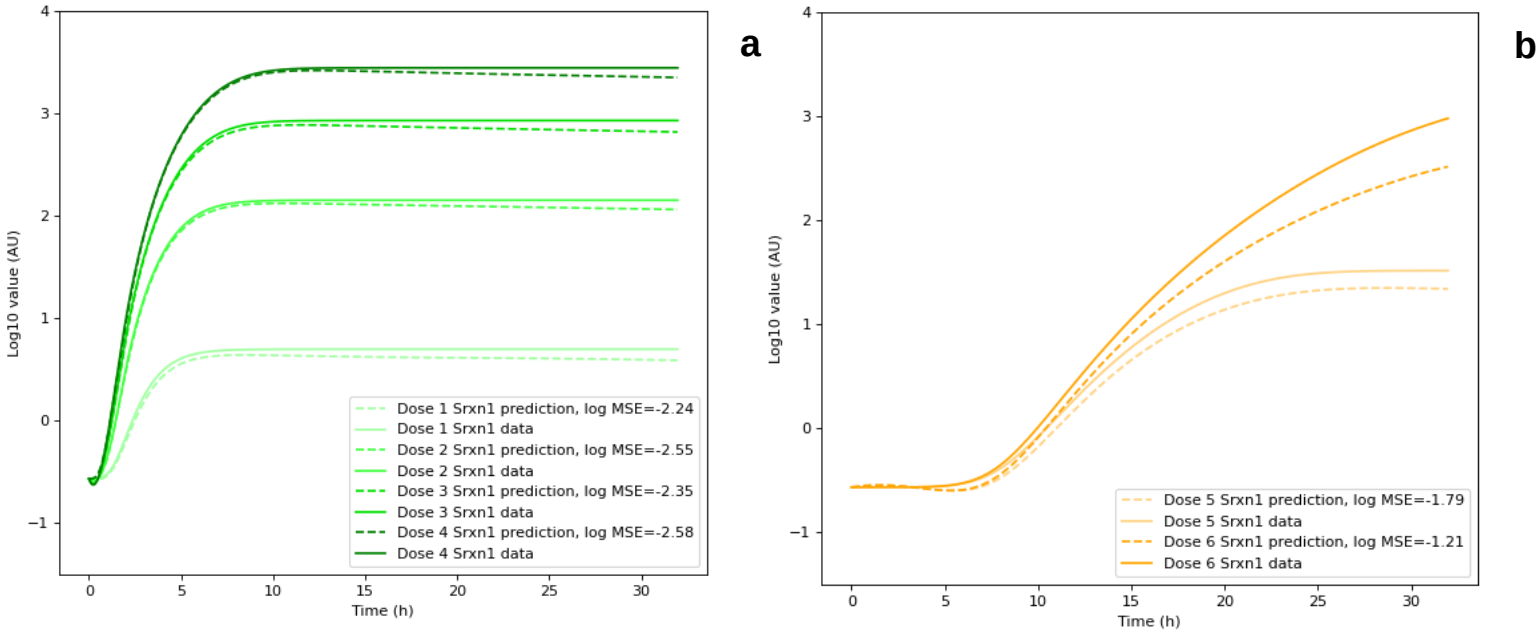


Figure 12. $LSTM_{dtp}$ fit on Srxn1 ODE data a) Training fit (dashed lines) on the data (solid lines). b) Validation fit (dashed lines) on the data (solid lines) resulting from an altered Nrf2 dynamic compared to the training data.

As training with the $LSTM_{cv}$ and $LSTM_{tp}$ configurations did not show potential in learning the Srxn1 ODE, a training configuration was used that is more in line with the functioning of ODEs. This configuration, referred to as the derivative time point configuration ($LSTM_{dtp}$), uses tensors which are generated similarly to $LSTM_{tp}$ tensors in which each time point is its own sample. It differs from the $LSTM_{tp}$ configurations previously used however as it uses time, Nrf2 and Srxn1 as input data and the derivative of Srxn1 as output data. Similarly to the Srxn1 ODE, the network calculates the derivative of Srxn1 through both the current Nrf2 value as well as the current Srxn1 value. It was hypothesised that this similarity between the Srxn1 ODE and neural network configuration would yield better results regarding learning the mechanism as opposed to configurations solely reliant on Nrf2 data as an input.

In order to cast predictions with a neural network that gives derivatives as an output an integrator was written. For casting a prediction, the integrator uses the initial Srxn1 value of the corresponding Srxn1 data and the entirety of the Nrf2 data. Through the dSrxn1 outputs, new Srxn1 values are generated by the combination of the previous Srxn1 value and the dSrxn1 prediction.

Training on the artificial data was performed using the $LSTM_{dtp}$ configuration of which the integrated predictions can be seen in figure 12a. The neural network acquired a good fit on the training data despite a minor deviation in slope towards the latter end of the curve. In addition, the network acquired a good qualitative fit on the validation data with each prediction matching the shape of its respective validation data. The validation predictions do show a minor dip around 7 hours not seen in the validation data. This may be explained by dose 4 of the training data having a very minor dip after its initial value. From a quantitative perspective the validation data is notably less accurate than the training data, having a log MSE value around 1.5 to 2 times lower than that of the training data. The validation fit has a high degree of reproducibility however as for all varying neural network seeds validation fits were achieved of identical qualitative accuracy and comparable quantitative accuracy.

$LSTM_{dtp}$ configuration has learned proper time dynamics

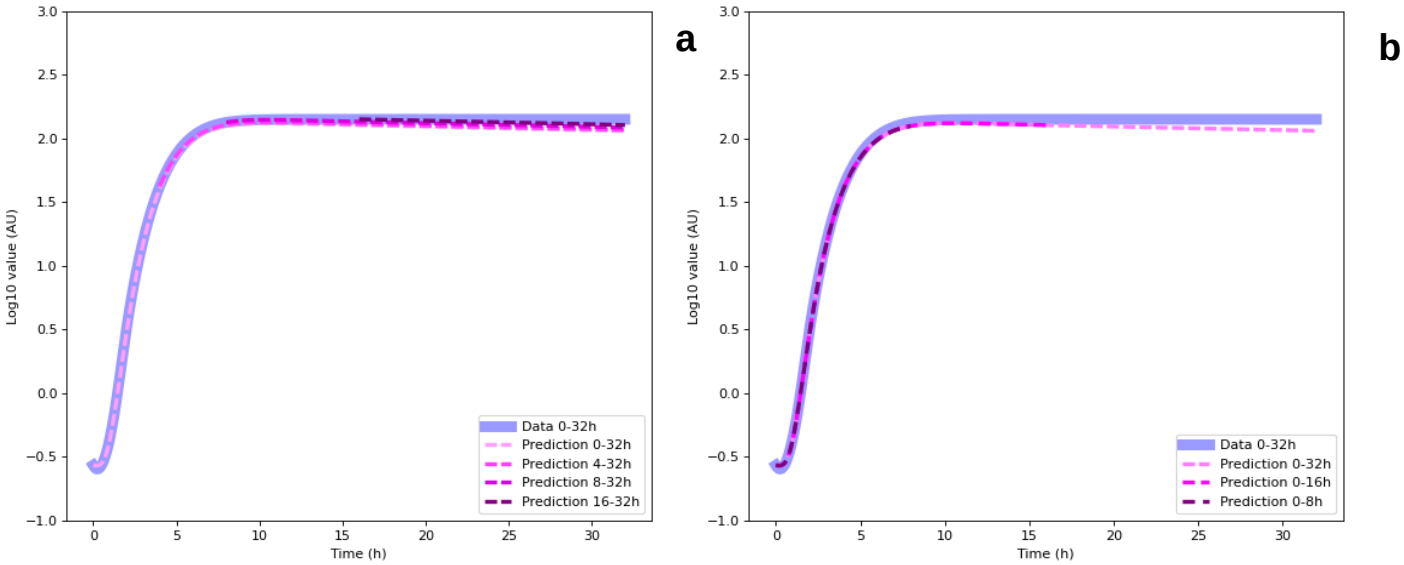


Figure 13. $LSTM_{dtp}$ time tests. a) integrated predictions for varying initial times (purple) compared to the data (blue). b) integrated predictions for varying final times (purple) compared to the data (blue).

Time frame tests were run for the $LSTM_{dtp}$ configuration with 0, 4, 8 and 16 hours as varying initial time points (fig. 13a) and 32, 16 and 8 hours as varying end time points (fig. 13b) as a control. As opposed to the $LSTM_{cv}$ configuration (fig. 9a), this neural network configuration shows a good fit to alternating initial starting points. Of note however is a very mild vertical shift between the predictions of the varying initial time points, the cause of which is likely to be traced back to an error in the integrator function leading to a mild shift in data points. These tests have shown that the $LSTM_{dtp}$ configuration can adapt to varying starting times, allowing application for alternate timeframes than the timeframe of the training data.

$LSTM_{dtp}$ configuration neural network learned general pattern of SUL Srxn1 data

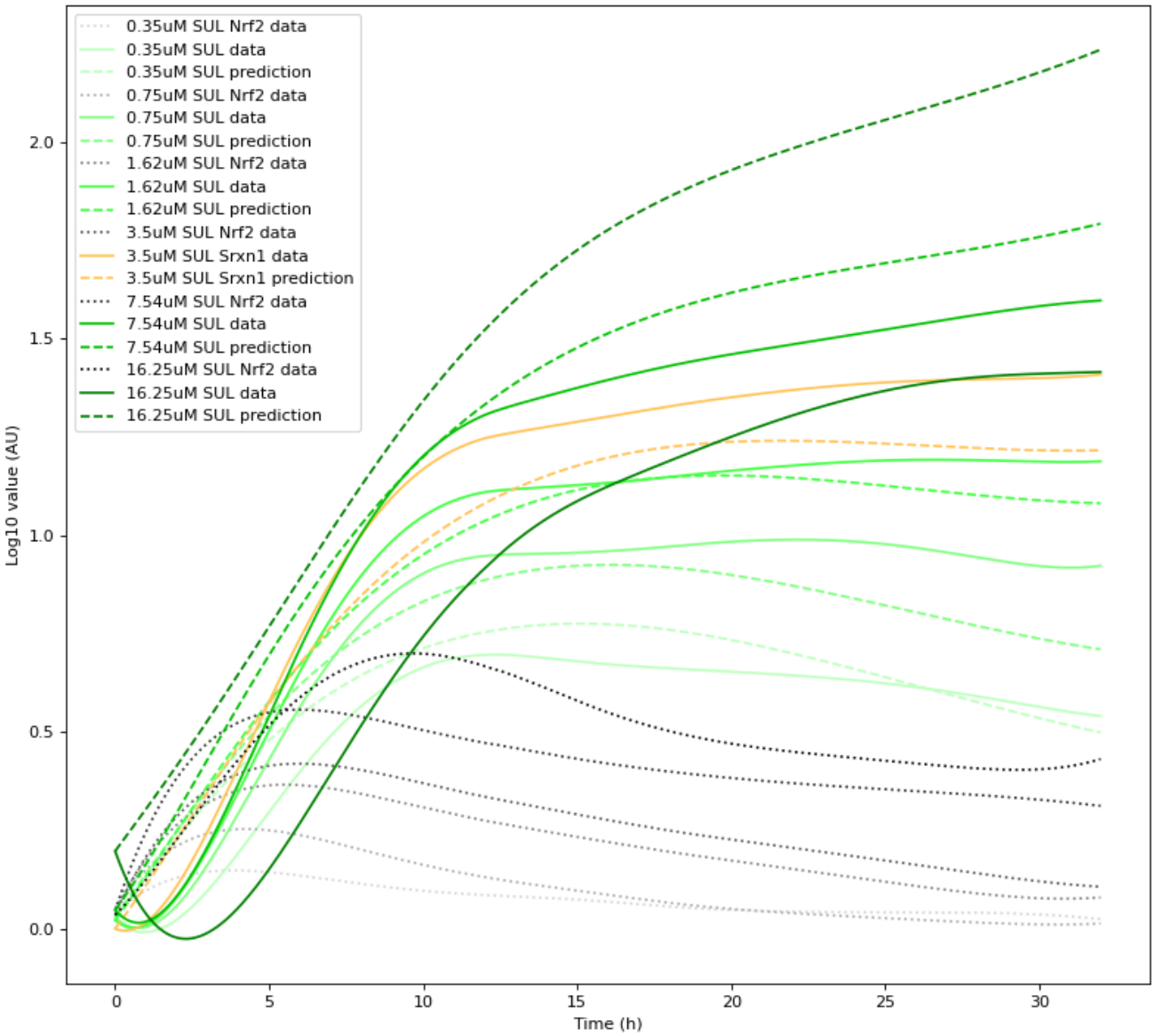


Fig 14. $LSTM_{dtp}$ configuration fit on SUL Srxn1 data. Dashed lines represent predictions for training (green) and validation (orange) Srxn1 data. Solid lines represent training (green) and validation (orange) Srxn1 data. Black dotted lines represent the Nrf2 data for their respective doses.

As the $LSTM_{dtp}$ configuration had proven capable of understanding a mechanism as presented by a Srxn1 ODE (fig. 12b) the configuration was applied to SUL compound exposure data. Using SUL doses 0.35 µM, 0.75 µM, 1.62 µM, 7.54 µM and 16.25 µM as training data and 3.5 µM as validation data, the neural network was able to attain a decent qualitative fit for all doses except 16.25 µM, which has a deviating pattern. All predictions matched the macro patterns of their respective data to a greater or lesser degree. From a qualitative perspective the later time points of the predictions have a better fit to the data compared to the earlier time points as the neural network was largely accurate in predicting whether there was a positive or negative slope at around 32 hours. It however did not predict the minor dip at 0-3 hours which is likely to be a by-product of smoothing and log scaling the data rather than a part of the biological reality. That the neural network predictions do not follow into the dip as presented by the data could be seen as a positive sign that the neural network relates rising Nrf2 levels to a rising Srxn1 as opposed to overfitting on a data artefact.

Neural network predictions show sensitivity to Nrf2 decay

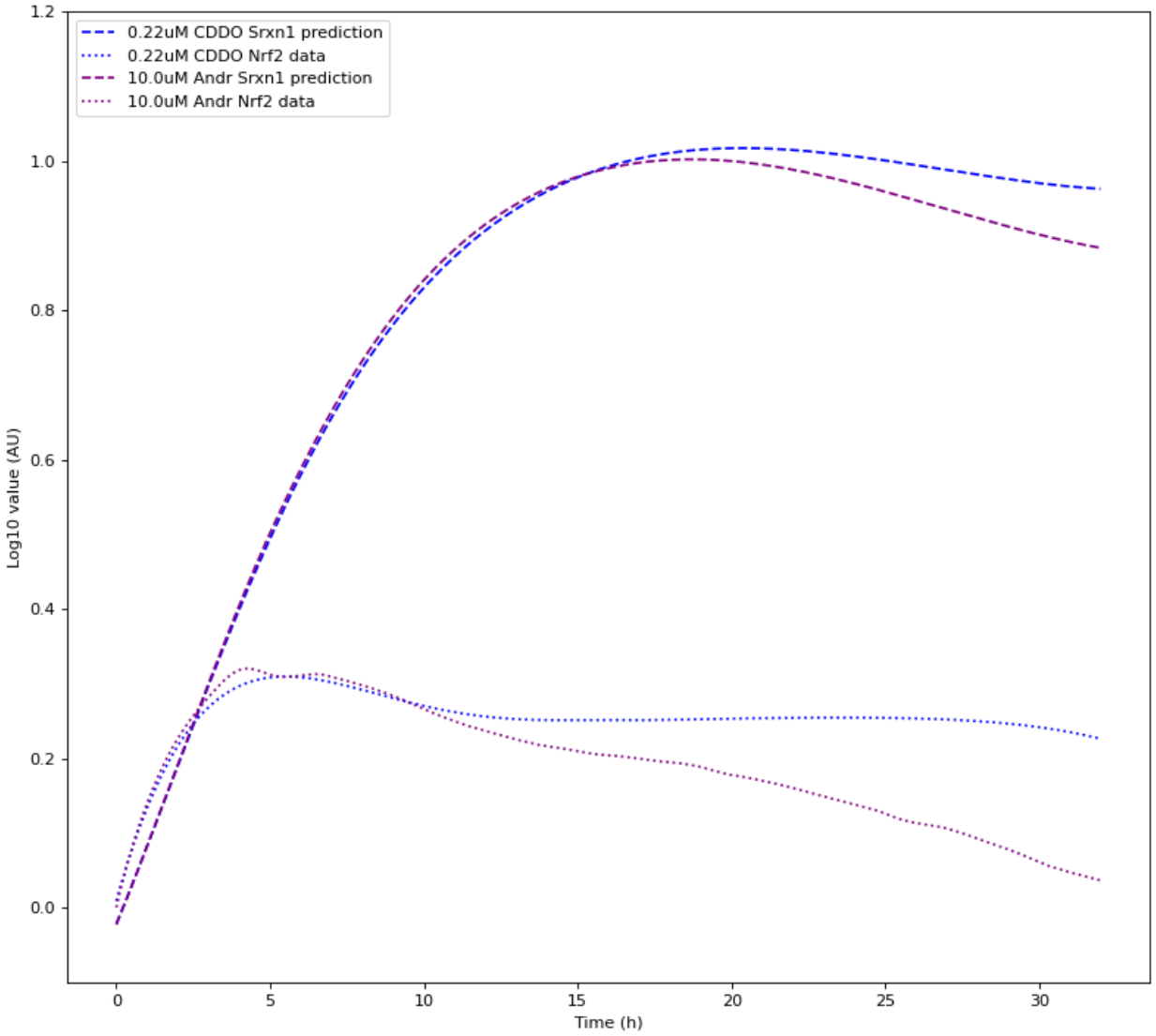


Figure 15. $LSTM_{dtp}$ model testing using similar data Predictions were cast using a $LSTM_{dtp}$ model trained on SUL data. Dotted lines represent Nrf2 compound exposure data for CDDO (blue) and ANDR (purple). Dashed lines represent integrated Srxn1 predictions corresponding to the CDDO (blue) or ANDR (purple) Nrf2 input data.

Another way of gaining insight into the mechanism learning of the compound data by the neural network is by analysing predictions for almost similar data. It is assumed that if the neural network had learned a more advanced Nrf2-Srxn1 relationship than the simple replication of the training data dynamics, compound doses with similar looking Nrf2 data would result in similar looking Srxn1 predictions, with any deviations in the Srxn1 predictions able to be traced back to specific deviations in the Nrf2 data. This would be opposed to bad fits that may show overfitting to identical points or patterns despite varying input data calling for different points or patterns.
To test this Nrf2 data was used following compound exposure to Andrographolide (ANDR) and CDDO (CDDO). As is shown in figure 15, Nrf2 data of ANDR and CDDO with overlapping peaks result in Srxn1 predictions that also reach very similar peaks. Additionally, the faster decay in ANDR Nrf2 results in a faster decay in ANDR Srxn1 as compared to the CDDO prediction with its slower decaying Nrf2 values. This demonstrates that the neural network is sensitive to the changes in slope of the Nrf2 input data.

Artificial data tests hint at learning beyond training dynamics

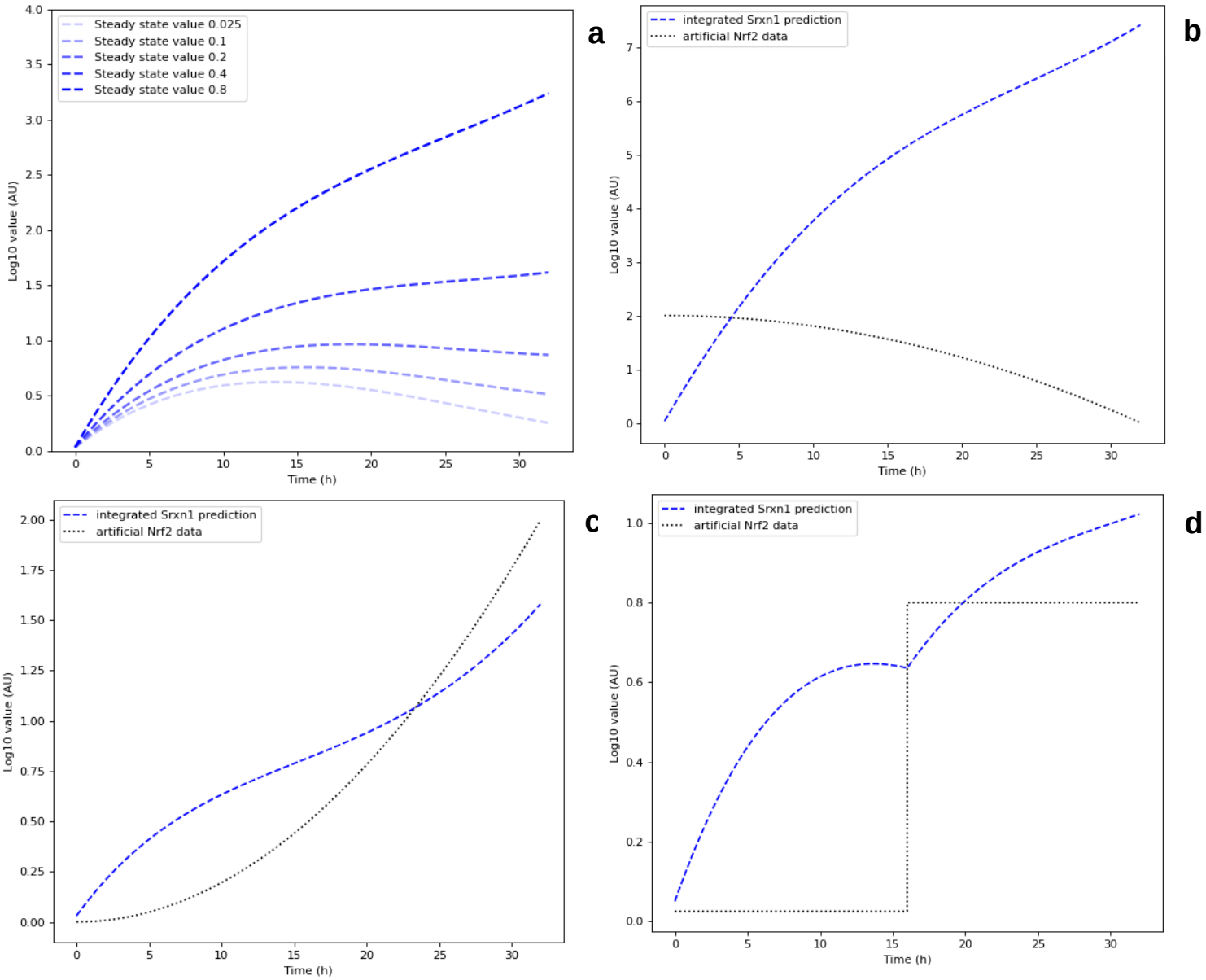


Figure 16. $LSTM_{dtp}$ model testing using artificial data. a) Integrated Srxn1 predictions for Nrf2 steady state inputs of varying values. b) Integrated Srxn1 prediction for exponentially decaying Nrf2 values. c) Integrated Srxn1 values for exponentially increasing Nrf2 values. d) Integrated Srxn1 predictions for an Nrf2 steady state of varying levels.

To gain further understanding into the Nrf2-Srxn1 mechanism that has been learned by the neural network, several tests have been performed using artificial data vectors. Initial testing has been performed on a simulated Nrf2 steady state, which was achieved through filling the Nrf2 input vector to the neural network integrator with a constant value. These tests were run for 5 steady state values largely covering the value range of the SUL Nrf2 GFP data. The results are shown in figure 16a, where it can be seen that the predictions follow an arching pattern with an initial rise largely independent of Nrf2 concentration, followed by a slope of which its gradient is correlated with the Nrf2 steady state value.

Though the steady state predictions show little noise or abnormal patterns, the steady state tests did mimic the SUL data tests (fig. 14) which is reason to suspect overfitting on the dynamics of the training data. Additional tests were therefore performed with exponentially decreasing and exponentially increasing Nrf2 input data in an attempt to break these dynamics, the results of which can be seen in figure 16b and 16c respectively. Of note here is that the exponentially decreasing Nrf2 (fig. 16b) yielded a prediction leading up to 5 fold the value of the exponentially increasing Nrf2 (fig. 16c), a result that may be attributed to a higher AUC of the respective Nrf2 data. Additionally, the exponential increasing Nrf2 test reveals that there is an initial rise in Srxn1 at t0 regardless of the corresponding Nrf2 values as Nrf2 concentrations start at near zero. This finding is indicative to partial overfitting on the training data dynamics. The exponential increase test did lead to an exponential increase in Srxn1 at the later time points, which can be seen as a break in the arch shaped dynamics seen in earlier tests (fig. 14, 16a).
To further attempt to disrupt the training data dynamics a test was performed with a simulated steady state scenario with varying Nrf2 levels, the result of which can be seen in figure 16d. For this test the Nrf2 input vector to the integrator was filled with the value of 0.025 for all data points up to 16 hours, and likewise filled with the value of 8.0 for all data points from 16 to 32 hours. From this result it is observed that the downward slope of Srxn1 around 16 hours is mitigated by a steep rise in Nrf2 resulting in a positive slope for the Srxn1 prediction. This test has shown that the prediction's dynamics are adaptable to disruptions in the Nrf2 data input.

All in all the difficulty in discovering whether the neural network has learned the underlying mechanism of the Nrf2-Srxn1 relationship from compound exposure data comes from the absence of proper validation data. This is because the compound exposure data shows one set of dynamics for Nrf2 and one set for Srxn1, resulting in difficulty in judging whether the neural network could accurately predict Srxn1 for varying Nrf2 dynamics or whether it has overfitted on the patterns of the Srxn1 data. Using ODE generated artificial data the neural network has proven capable of learning the underlying Nrf2-Srxn1 mechanism as represented by an ODE as it could accurately predict Srxn1 patterns following alternative Nrf2 dynamics (fig. 12b). After training on SUL data, testing on similarly looking Nrf2 data (fig. 15) has proven that the compound data trained neural network is sensitive to minor changes in input data. In addition, testing the SUL trained model with simple artificial data (fig. 16) has proven that the pattern of the Srxn1 training data dynamics could be broken with varying patterns of Nrf2 input data. In the absence of proper validation for testing the learning of a mechanism, these tests have provided evidence that the neural network can learn the Nrf2-Srxn1 relationship to a greater complexity than the mere mimicry of the training dynamics.

Qualitative compound comparison tests give indication of CDDO compound specificity

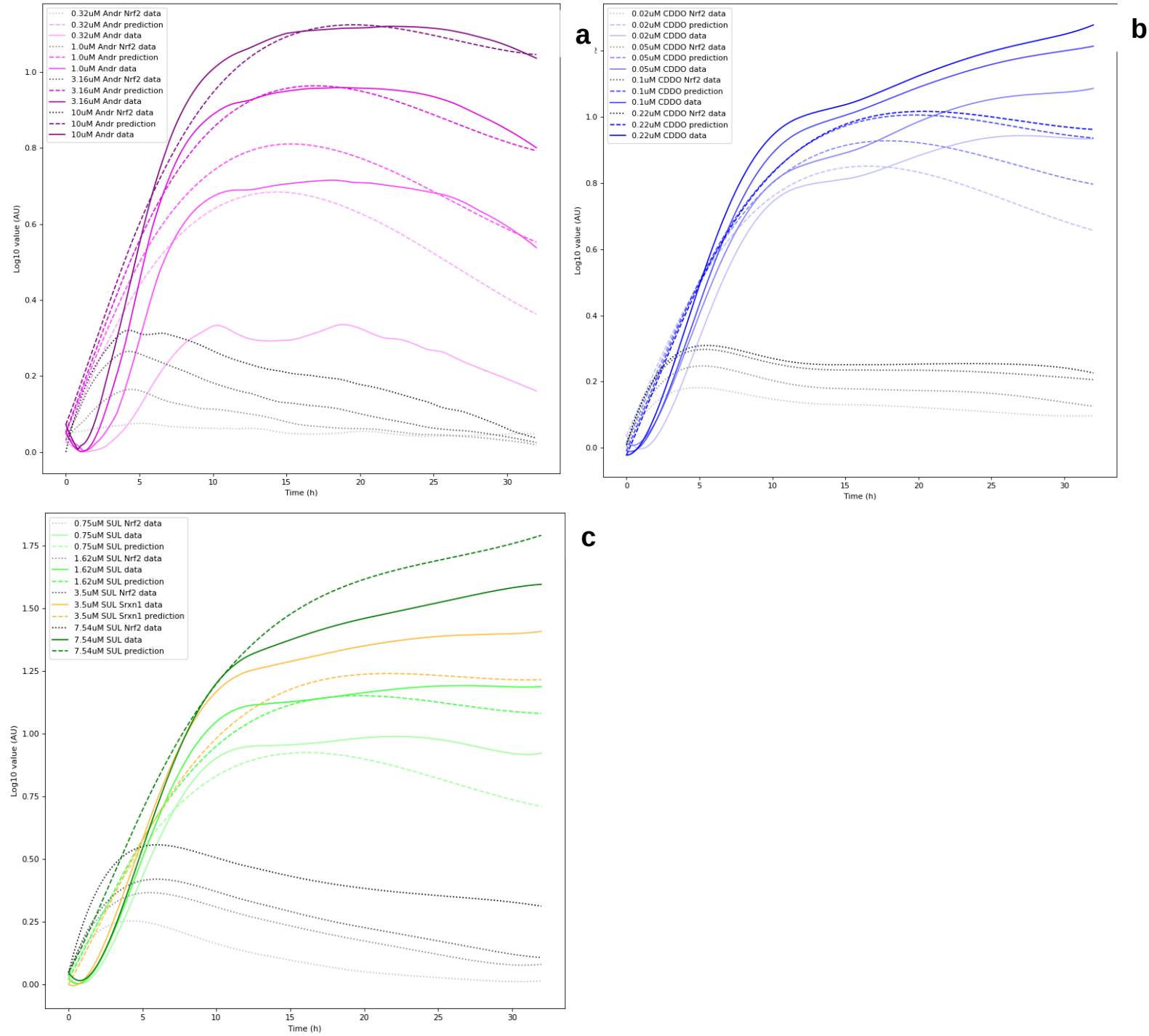


figure 17. $LSTM_{dtp}$ **qualitative compound comparison** Compound specificity tests using a model trained on SUL data. a) SUL model fit on ANDR Srxn1 using ANDR Nrf2 input data. b) SUL model fit on CDDO Srxn1 using CDDO Nrf2 input data. c) model fit on SUL training (green) and validation (orange) data. Dashed lines present Srxn1 predictions with their corresponding data presented by solid lines of the same colour. Respective Nrf2 input data is presented by the black dotted lines.

Having accrued evidence that the $LSTM_{dtp}$ neural network could potentially learn the underlying Nrf2-Srxn1 mechanism based on compound exposure data the model was applied to the study of the role of compound specificity in the Nrf2-Srxn1 mechanism. This was done through first training the model on the SUL exposure data so as to learn the implicit Nrf2-Srxn1 mechanism of the SUL data. It was hypothesized that using Nrf2 data of different compounds as an input to the neural network would result in a Srxn1 prediction that followed the SUL Nrf2-Srxn1 mechanism, despite the Nrf2 input potentially presenting different patterns compared to the SUL Nrf2 data. From here it was assumed that if these predictions matched the Srxn1 data of those alternative compounds then that would imply that the Nrf2-Srxn1 relationship of those compounds followed the same mechanism as that of SUL Nrf2-Srxn1.

These tests have been run using the SUL trained model initially presented in figure 14 for compounds ANDR and CDDO, the results of which can be seen in figure 17. Of note for every compound fit is the difference between the initial slope of the data and the initial slope of the predictions. The minor dip in value that can be observed in the data around 1 hours is likely to be a result from log scaling and smoothing the data. The predictions do not show such a pattern due to their integration process, as dSrxn1 is calculated from the initial Srxn1 and the immediately rising Nrf2. As a consequence the Srxn1 predictions are more a product of the rising Nrf2 value than the result of overfitting on the SUL training data.

For ANDR the predictions (fig. 17a) for the higher doses are within the same range as their respective data with only the lowest dose presenting a strong deviation. The Nrf2 concentrations corresponding to this dose maintains very low values over the 0-32 hour period, which may lead to the suspicion that the model is not as sensitive to lower values of Nrf2 thus resulting in an overprediction of Srnx1. Despite that the model seems to perform generally well for ANDR data with the highest doses having an accurate fit for their highest values. Additionally the model seems to scale the patterns of its Srxn1 predictions well for ANDR. This is exemplified by the predictions reaching their peaks in an accurate time range and also the final slopes around 28 hours somewhat matching that of their respective data.

Regarding the CDDO predictions (fig. 17b), there is an apparent difference in gradients between the predictions and the data occurring at the 20-32 hour timeframe. The perpetually rising Srxn1 concentration presented by the data may be explained by the sustained Nrf2 concentration, a phenomenon that is less present in the SUL and ANDR Nrf2 data (fig. 17a,c). This raises the question of whether the discrepancy between the Srxn1 predictions and the data is actually the product of poor mechanism learning on the side of the model or whether there really is compound specificity at play. The neural network fit on the SUL data has shown that Srxn1 concentrations can be sustained after the Nrf2 concentrations dips below the value of 0.25, as is showcased by the predictions of 1.62 µM and 3.5 µM SUL. Looking at the 0.1 µM and 0.22 µM data of CDDO however, the neural network predicts a slightly negative gradient for CDDO Srxn1 starting from ~23 hours, this despite the Nrf2 concentration ranging around the value of 0.25. One difference between the SUL Nrf2 and the CDDO Nrf2 is that is that the CDDO Nrf2 reaches lower peak values around 5 hours. The aforementioned difference in slope from ~23 hours between the CDDO and SUL predictions may therefore be explained not by the Nrf2 concentration at ~23 hours, but by the lower peaks in CDDO Nrf2 reached at ~5 hours.

Quantitative compound comparison supports CDDO specific mechanism

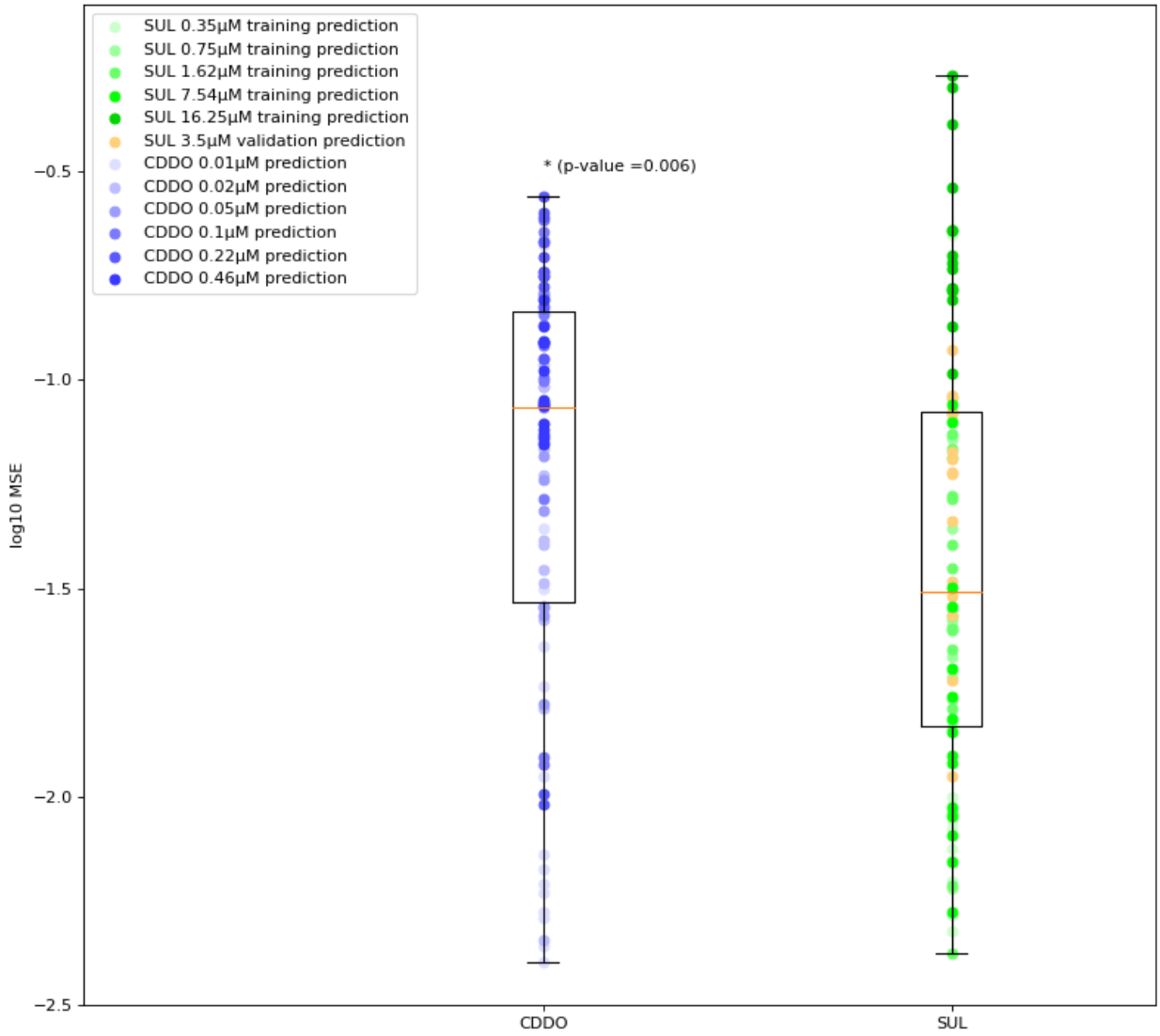


Figure 18. Quantitative compound comparison. Each dot represents the log scaled MSE value of an Nrf2 based Srxn1 prediction compared to its respective Srxn1 data. SUL data was used for model training and optimization, of which green and orange represent training and validation data respectively. Blue dots represent MSE values of Srxn1 predictions for CDDO exposure based Nrf2 as an input to the SUL trained neural network, calculated by comparison to CDDO Srxn1 data for their respective dose. Multiple dots for the same dose value are the result of using varying neural network fits through the use of different seeds.

Though the compound specificity test for CDDO hinted at a potentially different Nrf2-Srxn1 mechanism, the neural network fit on the data was questioned as the network may have biased earlier time points. Another compound comparison approach was therefore utilized that incorporated multiple neural network fits in its comparison. This quantitative compound comparison utilized a SUL trained $LSTM_{dtp}$ neural network that had been trained for 15 different neural network seeds, result in 15 different neural network fits on the training data. Of each fit predictions were cast for all CDDO and SUL doses of which the MSE was calculated in comparison to the Srxn1 data of its respective dose. The collected MSE values of all CDDO and SUL predictions for all 15 different neural networks fits are plotted in figure 18.

As can be seen in this figure, there is a significant difference between the means and the distribution of the SUL and CDDO MSE values. Despite having lower MSE values on average however the SUL data presents a higher maximum with its upper whisker filled with 16.25 µM predictions. This phenomenon may be explained by the Srxn1 data for 16.25 µM having lower values as could be expected based on its Nrf2 values being the highest of all doses (fig. 14). Overall this result shows that even accounting for the variability in learned training patterns, the CDDO predictions show a discrepancy with their respective data. To further elaborate on the significance of these findings it is necessary to state that for every neural network fit a very general pattern was observed with overfitting on the training data being close to non-existent. This therefore reduces the odds of the result shown in figure 18 being the product of overfitting.

DISCUSSION

The main objective of this research was the study of the OSR pathway using neural networks, the first part of which included learning Nrf2 dynamics using limited data. Though the neural network had shown some potential in learning the compound exposure based Nrf2 dynamics, the results were mediocre as neither the $LSTM_{cv}$ (fig. 3) nor the $LSTM_{tp}$ (fig. 4) configuration could scale well across the entire dose-response dynamic. In the case of the $LSTM_{cv}$ configuration this is because the training resulted in an unnatural plateauing effect of Nrf2 around the highest values of the training data. This phenomenon immediately restricts the usage of the neural network to interpolation between training doses and extrapolations for lower doses than the training data as predictions for higher doses would result in a plateau. A potential cause of this phenomenon could be the nature of the input data as the memory component of LSTM cells may require more intravectorial variation in order to function properly as opposed to linear time and dose values. Using the time point configuration the neural network shows more potential in learning the dose based Nrf2 dynamics as it didn't result in the plateauing effect of the $LSTM_{cv}$ configuration therefore also allowing for higher dose extrapolations.

From the results it can be observed that the neural network learns a pattern based on the training data it is given, as the data with good training fits resulted in inter- and extrapolations that resembled the training data in shape (fig. 4). From this perspective the learning process of this neural network could be interpreted as the learning of a dose-response curve, in which the neural network learns the underlying pattern of how a higher response differs in shape from a lower response. The accuracy of the interpolation and extrapolation are subsequently dependent on how many doses are used and where these doses are on the dose-response curve, as that would determine how well the network can scale its predictions in accordance with the dose-response curve associated with the data. If training was done on two doses, it is assumed that the neural network would scale linearly between where these two points are on the dose-response curve.

Where the $LSTM_{tp}$ configuration could see some utility however is in the potential for using limited data point predictions (fig. 5,11). As the neural network configuration is not limited by the dimensions of its training samples (unlike the $LSTM_{cv}$ configuration that requires all samples to have the specific length of the training data), it would theoretically allow for model validation using but a few data points in the place of a full length vector. This then translates to the potential learning of proper dose dynamics using fewer time measurements as validation (i.e. 4 and 8 hours) as compared to validation data of the full time range (i.e. 0-32 hours).

The second part of studying the OSR pathway using neural networks included the study of the Nrf2-Srxn1 relationship. Here the $LSTM_{cv}$ configuration has shown its limitations in regards to predictions for differing timeframes (likely due to the lack of samples with different starting times) (fig. 9a) as well as the learning of the compound mechanism.
In regards to the learning of a protein-protein mechanism as presented by a Srxn1 ODE, the predictions on the validation data (fig. 8b) were very similar to that of the $LSTM_{tp}$ validation data fit (fig. 10b). This may indicate that the shared pattern could be the extent to which the predictions can deviate from the training dynamics using just time and Nrf2 as an input, and as such that the addition of Srxn1 is necessary for the proper understanding of the ODE.

Of all training configurations the $LSTM_{dtp}$ configuration has proven to be the most capable in learning a more detailed form of the Nrf2-Srxn1 relationship as it goes beyond the mimicry of training dynamics (fig. 12). Using the $LSTM_{dtp}$ configuration the model was used to study the role of compound specificity on the Nrf2-Srxn1 mechanism. Both qualitative (fig. 17) and quantitative (fig. 18) tests have revealed that compound specificity could play a role in the Nrf2-Srxn1 relationship, in which case there are multiple potential mechanisms mediating this compound specificity. For instance, it has been demonstrated that NF-κB is capable of inducing Srxn1 expression, yet also that Nrf2 expression leads to an inhibition of NF-κB activation (30,31).

Certain compounds could potentially nullify Nrf2 mediated inhibition of NF-κB activation, leading to Srxn1 upregulation through both Nrf2 and NF-κB. Alternatively there are the acetyltransferases CPB and P300 which are responsible for the acetylation of Nrf2, in turn resulting in increased DNA binding and transcriptional strength of Nrf2 (32,33). A compound's interaction with P300/CBP could therefore influence Nrf2's ability to increase Srxn1 expression. If there is a compound specific Nrf2-Srxn1 mechanism there is also the degree to which such a mechanism may be observed that needs to be accounted for. If a compound for example interacts with a Srxn1 regulatory protein, the compound kinetics like its stability may determine the degree to which it influences the course of the Srxn1 concentration.

Unfortunately the compound specificity tests have not yielded very robust evidence of the presence of compound specificity. This is largely due to discrepancies between Srxn1 predictions for similar Nrf2 values presenting reason to question the accuracy of the learned mechanism. Seeing as a higher initial peak in Nrf2 presented a greater predictor of the Srxn1 gradient at later time points than the Nrf2 value at later time points, it could be assumed that the neural network incorporates a delay between the Nrf2 value and the Srxn1 value that follows from it. The steady state multiple exposure test (fig. 16d) however has provided evidence that a rise in Nrf2 has an immediate effect on Srxn1 concentration. In addition, the steady state Nrf2 test (fig. 16a) and the exponentially increasing Nrf2 test (fig. 16c) have showcased that there is partial overfitting on the training data leading to an unconditional initial rise in Srxn1. This opens up the possibility that the neural network gives greater priority to earlier time points, becoming less sensitive to changes at later time points. This problem could potentially be solved by adding an Nrf2 and Srxn1 steady state baseline to the training data as a separate dose.

Where the $LSTM_{dtp}$ model could also see improvement is in the neural network's architecture which (aside from optimizing the hyperparameters) has not been properly developed for the function that it serves. A custom built neural network architecture could improve the predictive accuracy of a learned ODE system as is evidenced by a 2019 study (34). In this study a continuous depth neural network was developed that had shown far more accurate predictions for ODE systems than the standard discretely layered neural networks.

Another architectural improvement that may bring the model closer to its function of delivering mechanistic insight lies in moving away from the standard neural network approach of mapping input to output. Though this standard approach may come close to representing an equation, it is never more than a correlation with no true implicit equation being present in the trained model. A 2018 study solves this by incorporating algebraic operations into the neural network layers, effectively reducing part of the neural network´s black box nature to take the form of equations (35). Utilizing an architecture that learns implicit equations would be a step towards deciphering the equations underlying protein interactions.

However even with this approach, the learned equation would still be hidden inside the neural network. The last component needed for gaining proper mechanistic insight into protein-protein relationships would therefore be a model that transforms learned mechanisms into mathematical equations as a text based output. It is believed that an ODE can produce a wide variety of patterns depending on its parameters and variables, but also that there is a limit to the range of patterns an ODE can produce (assuming controlled data of its variables) that is inherent to the structure of the formula. If this assumption is correct for a range of ODEs then it opens up the possibility of developing a neural network that learns how to transform the limitation in patterns inherent to an ODE´s formula into the ODE´s formula as a text based output.

It is proposed that a two part model is needed for proper machine learning mediated mechanistic understanding of protein systems. The first part of this model would be filled by a robust model that understands protein-protein relationships of which a potential framework is featured in this study and a second part by a model that translates these relationships into mathematical equations as a text based output. If successful, this proposed two part model could one day transform raw protein expression data into an ODE model using limited human intervention.

CONCLUSION

Using neural networks the oxidative stress response has been studied in a quantitative way covering the compound exposure to Nrf2 relationship and the Nrf2 to Srxn1 relationship. Using limited training data the neural network has proven some capacity in inter- and extrapolating Nrf2 dynamics based on compound exposure values with its limited data predictions allowing for a flexible practical application of the model. Using a derivative calculating configuration, the neural network has proven capable of understanding an Nrf2-Srxn1 relationship of greater complexity than the mimicry of training data dynamics. Though compound comparison tests hint at the presence of compound specificity, a more robust model is needed in order to rule out overfitting on initial time points. Regardless this model provides a framework for a neural network based approach for the quantitative study of protein signalling pathways.

REFERENCES

1. Suk, K. T., & Kim, D. J. (2012). Drug-induced liver injury: present and future. *Clinical and molecular hepatology*, **18**(3), 249–257.
2. David, S., & Hamilton, J. P. (2010). Drug-induced Liver Injury. *US gastroenterology & hepatology review*, **6**, 73–80.
3. Onakpoya, I. J., Heneghan, C. J., & Aronson, J. K. (2016). Post-marketing withdrawal of 462 medicinal products because of adverse drug reactions: a systematic review of the world literature. *BMC medicin*e, **14**, 10.
4. Fung M, Thornton A, Mybeck K, Wu, J. H., Hornbuckle, k., Muniz, E. (2001). evaluation of the characteristics of safety withdrawal of prescription drugs from worldwide pharmaceutical markets-1960 to 1999. *Drug Inf J*., **35**, 293–317.
5. David, S., & Hamilton, J. P. (2010). Drug-induced Liver Injury. *US gastroenterology & hepatology review*, **6**, 73–80.
6. Donato, M., & Tolosa, L. (2021). High-Content Screening for the Detection of Drug-Induced Oxidative Stress in Liver Cells. *Antioxidants (Basel, Switzerland)*, **10**(1), 106.
7. Deavall, D. G., Martin, E. A., Horner, J. M., & Roberts, R. (2012). Drug-induced oxidative stress and toxicity. *Journal of toxicology*, 645460.
8. Guo, C., Sun, L., Chen, X., & Zhang, D. (2013). Oxidative stress, mitochondrial damage and neurodegenerative diseases. *Neural regeneration research*, **8**(21), 2003–2014.
9. Zorov, D. B., Juhaszova, M., & Sollott, S. J. (2014). Mitochondrial reactive oxygen species (ROS) and ROS-induced ROS release. *Physiological reviews*, **94**(3), 909–950.
10. Cooke, M.S., Evans, M.D., Dizdaroglu, M. and Lunec, J. (2003), Oxidative DNA damage: mechanisms, mutation, and disease. *The FASEB Journal*, **17**, 1195-1214.
11. Tubbs A., Nussenzweig A. (2017), Endogenous DNA Damage as a Source of Genomic Instability in Cancer. *Cell*, **168**(4), 644-656
12. Kuijper, I. A., Yang., H., Van De Water, B., Beltman, J.B., (2017). Unraveling cellular pathways contributing to drug-induced liver injury by dynamical modeling, *Expert Opinion on Drug Metabolism & Toxicology*, **13**(1), 5-17
13. Kobayashi, A., Kang, M. I., Okawa, H., Ohtsuji, M., Zenke, Y., Chiba, T., Igarashi, K., & Yamamoto, M. (2004). Oxidative stress sensor Keap1 functions as an adaptor for Cul3-based E3 ligase to regulate proteasomal degradation of Nrf2. *Molecular and cellular biology*, **24**(16), 7130–7139.
14. Dayalan Naidu, S., & Dinkova-Kostova, A. T. (2020). KEAP1, a cysteine-based sensor and a drug target for the prevention and treatment of chronic disease. *Open biology,* **10**(6), 200105.
15. Zhang, D. D., Lo, S. C., Cross, J. V., Templeton, D. J., & Hannink, M. (2004). Keap1 is a redox-regulated substrate adaptor protein for a Cul3-dependent ubiquitin ligase complex. *Molecular and cellular biology*, **24**(24), 10941–10953.
16. Cullinan, S. B., Gordan, J. D., Jin, J., Harper, J. W., & Diehl, J. A. (2004). The Keap1-BTB protein is an adaptor that bridges Nrf2 to a Cul3-based E3 ligase: oxidative stress sensing by a Cul3-Keap1 ligase. *Molecular and cellular biology*, **24**(19), 8477–8486.
17. Wu, J. H., Miao, W., Hu, L. G., & Batist, G. (2010). Identification and characterization of novel Nrf2 inducers designed to target the intervening region of Keap1. *Chemical biology & drug design*, **75**(5), 475–480. https://doi.org/10.1111/j.1747-0285.2010.00955.x
18. Zhang, D. D., & Hannink, M. (2003). Distinct cysteine residues in Keap1 are required for Keap1-dependent ubiquitination of Nrf2 and for stabilization of Nrf2 by chemopreventive agents and oxidative stress. *Molecular and cellular biology*, **23**(22), 8137–8151.

19. Reddy S. P. (2008). The antioxidant response element and oxidative stress modifiers in airway diseases. *Current molecular medicine*, **8**(5), 376–383.
20. de Figueiredo, S. M., Binda, N. S., Nogueira-Machado, J. A., Vieira-Filho, S. A., & Caligiorne, R. B. (2015). The antioxidant properties of organosulfur compounds (sulforaphane). *Recent patents on endocrine, metabolic & immune drug discovery*, **9**(1), 24–39.
21. Wu, S., Lu, H., & Bai, Y. (2019). Nrf2 in cancers: A double-edged sword. *Cancer medicine*, **8**(5), 2252–2267.
22. Singh, A., Ling, G., Suhasini, A. N., Zhang, P., Yamamoto, M., Navas-Acien, A., Cosgrove, G., Tuder, R. M., Kensler, T. W., Watson, W. H., & Biswal, S. (2009). Nrf2-dependent sulfiredoxin-1 expression protects against cigarette smoke-induced oxidative stress in lungs. *Free radical biology & medicine*, **46**(3), 376–386.
23. Apopa, P. L., He, X., & Ma, Q. (2008). Phosphorylation of Nrf2 in the transcription activation domain by casein kinase 2 (CK2) is critical for the nuclear translocation and transcription activation function of Nrf2 in IMR-32 neuroblastoma cells. *Journal of biochemical and molecular toxicology*, **22**(1), 63–76.
24. Yang, H., Niemeijer, M., Van de Water, B., Beltman, J.B. 2020. ATF6 Is a Critical Determinant of CHOP Dynamics during the Unfolded Protein Response. *iScience*, **23**(2), 100860.
25. Wijaya, L. S., Trairatphisan, P., Gabor, A., Niemeijer, M., Keet, J., Alcalà Morera, A., Snijders, K. E., Wink, S., Yang, H., Schildknecht, S., Stevens, J. L., Bouwman, P., Kamp, H., Hengstler, J., Beltman, J., Leist, M., Le Dévédec, S., Saez-Rodriguez, J., & van de Water, B. (2021). Integration of temporal single cell cellular stress response activity with logic-ODE modeling reveals activation of ATF4-CHOP axis as a critical predictor of drug-induced liver injury. *Biochemical pharmacology*, **190**, 114591.
26. Kapuy, O., Papp, D., Vellai, T., Bánhegyi, G., & Korcsmáros, T. (2018). Systems-Level Feedbacks of NRF2 Controlling Autophagy upon Oxidative Stress Response. *Antioxidants (Basel, Switzerland)*, **7**(3), 39.
27. Chollet, F. (2018) *Deep Learning with Python*, 1, 3-116
28. Hochreiter, S., Schmidhuber, J. (1997). Long Short-term Memory. *Neural computation*, **9**(8), 1735-1780
29. Savitzky, A. & Golay, M. J. E. (1964). Smoothing and Differentiation of Data by Simplified Least Squares Procedures. *Anal. Chem*. **36**, 1627–1639
30. McElwee, M. K., Song, M. O., & Freedman, J. H. (2009). Copper activation of NF-kappaB signaling in HepG2 cells. *Journal of molecular biology*, **393**(5), 1013–1021.
31. Herpers, B., Wink, S., Fredriksson, L., Di, Z., Hendriks, G., Vrieling, H., de Bont, H., & van de Water, B. (2016). Activation of the Nrf2 response by intrinsic hepatotoxic drugs correlates with suppression of NF-κB activation and sensitizes toward TNFα-induced cytotoxicity. *Archives of toxicology*, **90**(5), 1163–1179.
32. Sun, Z., Chin, Y. E., & Zhang, D. D. (2009). Acetylation of Nrf2 by p300/CBP augments promoter-specific DNA binding of Nrf2 during the antioxidant response. *Molecular and cellular biology*, **29**(10), 2658–2672.
33. Kawai, Y., Garduño, L., Theodore, M., Yang, J., & Arinze, I. J. (2011). Acetylation-deacetylation of the transcription factor Nrf2 (nuclear factor erythroid 2-related factor 2) regulates its transcriptional activity and nucleocytoplasmic localization. *The Journal of biological chemistry*, **286**(9), 7629–7640.
34. Chen, R. T. Q., Rubanova, Y., Bettencourt, J., Duvenaud, D. (2018). Neural Ordinary Differential Equations. *arXiv.org*
35. Sahoo, S. S., Lampert, C. H., Martius, G. (2018). Learning Equations for Extrapolation and Control. *arXiv.org*

ACKNOWLEDGMENTS

Special thanks to Raju Sharma and Joost Beltman for making this project possible and granting access to the lab data. Additionally thanks are in place for their advice and in depth discussions which were decisive in moving the project in the right direction.

Appendix 1: data management

Data, model code and neural network versions are available on https://github.com/Sjvpruijn/OSR-neural-network-project. Access to which may be requested through the author.

Listed in table 2 are the neural network versions used to generate the figures listed in this paper. "Run nr." refers to the number assigned to a neural network training session, which itself can consist of multiple seeds. A neural network run trained on a specific seed is referred to as a version. For the scaler configurations see appendix 3. Trained neural network models corresponding to a run number and their seed values can be found on the Github under "versions".

Test name	Figure nr.	Run nr.	Epochs	seed	Scaler configuration
LSTMcv dose-Nrf2	3	216	4000	0	0.16
LSTMtp dose-Nrf2	4a	222	3000	1	0.17
LSTMtp dose-Nrf2	4b	222	3000	2	0.17
LSTMtp dose-Nrf2	4c	223	3000	0	0.17
LSTMtp dose-Nrf2	4d	223	3000	1	0.17
LSTMtp dose-Nrf2	4e	224	3000	0	0.17
LSTMtp dose-Nrf2	4f	224	3000	1	0.17
Dose-Nrf2 limited time points	5	224	3000	2	0.17
LSTMcv Nrf2-Srxn1	6	212.1	2000	0	0.15
LSTMcv ODE data	8	213	4000	2	1
LSTMcv ODE time test	9	213	4000	2	1
LSTMtp ODE data	10	215	1000	4	1
LSTMtp ODE limited time points	11	215	1000	4	1
LSTMdtp ODE data	12	207	1000	1	1
LSTMdtp ODE time test	13	207	1000	1	1
LSTMdtp SUL fit	14, 17c	210	1000	1	0.13
LSTMdtp Similar data tests	15	210	1000	1	0.13
LSTMdtp steady state	16a	210	1000	1	0.13
LSTMdtp exponential decrease	16b	210	1000	1	0.13
LSTMdtp exponential increase	16c	210	1000	1	0.13
LSTMdtp steady state multiple exposure	16d	210	1000	1	0.13
SUL model ANDR prediction	17a	210	1000	1	0.13
SUL model CDDO prediction	17b	210	1000	1	0.13
SUL model quantitative comparison	18	210	1000	0 to 14	0.13

Table 2: Test specific neural network settings.

Appendix 2: Task risk analysis

Name task: Computational work

Nr.	Action	Hazard	Risk classification (before)				Measures: -source -collective	Measures: -procedures -Instructions -PBM's	Risk classification (after)			
			S	E	P	Score			S	E	P	Score
1	Typing	wrist pain	**3**	**10**	**6**	180		perform wrist exercises	**3**	**10**	**1**	30
2	Sitting	back pain	**3**	**10**	**6**	180		Stretching, exercise, proper posture	**3**	**10**	**1**	30

Table 3: task risk analysis.

Appendix 3: 0-1scaler settings

All protein data (both compound exposure and artificial data) were 0-1 normalized using the 0-1 scaler fitted to the configurations listed in tables 4, 5, 6, 7 and 8. Nrf2 data was normalized using Nrf2scaler, Srxn1 data was normalized using Srxn1scaler and Srxn1 differential values were 0-1 normalized using Diffscaler.

0.13	min	max
Nrf2scaler	-3	3
Srxn1scaler	-3	3
Diffscaler	-0.005	0.005

Table 4. Scaler configuration 0.13.

0.17	min	Max
Nrf2scaler	0	1
Srxn1scaler	-3	3
Diffscaler	-0.005	0.005

Table 5. Scaler configuration 0.17.

1	min	Max
Nrf2scaler	0	1200
Srxn1scaler	-1	4
Diffscaler	-0.001	0.005

Table 6. Scaler configuration 1.

0.15	min	Max
Nrf2scaler	0	1
Srxn1scaler	-1	2
Diffscaler	-0.005	0.005

Table 7. Scaler configuration 0.15.

0.16	min	Max
Nrf2scaler	0	1.5
Srxn1scaler	-3	3
Diffscaler	-0.005	0.005

Table 8. Scaler configuration 0.16.

Appendix 4: training configurations

Input data to the neural network was prepared using the below listed training configurations, which transformed the data corresponding to a group of doses *d* into training or validation data. Tensor shapes are described as: (*first axis length, second axis length, third axis length*). Training functions take a list of doses as an input of which the corresponding data is to be incorporated in a tensor. The order in which data is added to a tensor is determined by the index of the corresponding dose value in the list.

Dose-Nrf2 neural network $LSTM_{cv}$ input tensor preparation

For the conventional LSTM training method a 0-1 scaler was fitted to the values of 0 and 2, and subsequently used to normalize the Nrf2 *Rmean* vectors of the run specific training and test doses.

An empty three dimensional tensor was generated using the numpy *zeros* function of shape (*d, len,* 2). This tensor was filled as follows: on each index of the first axis, the full second axis and the first index of the third axis was a *timevec* vector for each of *d* doses. On each index of the first axis, the full second axis and the second index of the third axis was a *dosevec* vector for each of *d* doses.

Dose-Nrf2 neural network $LSTM_{cv}$ output tensor preparation

Nrf2 *Rmean* vectors corresponding to the selected doses were 0-1 normalized using the Nrf2scaler. An empty three dimensional tensor was generated using the numpy *zeros* function of shape (*d, len,* 1) This tensor was filled as follows: On each index of the first axis, the full second axis and the first index of the third axis was a Nrf2 *Rmean* vector for each of *d* doses.

Dose-Nrf2 neural network $LSTM_{tp}$ input tensor preparation

Time and dose values were prepared as input data in the following way: An empty three dimensional tensor was generated using the numpy *zeros* function of shape (*d*len*, 1, 2). This tensor was filled as follows: on the full first axis and the first index of the third axis were *timevec* vectors appended together for each of the doses used for training. On the full first axis and the second index of the third axis were the *dosevec* vectors appended together in order of the selected doses.

Dose-Nrf2 neural network $LSTM_{tp}$ output tensor preparation

Nrf2 *Rmean* vectors corresponding to the selected doses were 0-1 normalized using the Nrf2scaler. An empty three dimensional tensor was subsequently generated using the numpy *zeros* function of shape (*d*len*, 1, 1). This tensor was filled as follows: on the full first axis were Nrf2 *Rmean* vectors appended together in order of the selected doses.

Nrf2-Srxn1 neural network $LSTM_{cv}$ input tensor preparation

Nrf2 *Rmean* vectors corresponding to the selected doses were 0-1 normalized using the Nrf2scaler.

An empty three dimensional tensor was generated using the numpy *zeros* function of shape (*d, len,* 2). This tensor was filled as follows: on each index of the first axis, the full second axis and the first index of the third axis was a *timevec* vector for each of *d* doses. On each index of the first axis, the full second axis and the second index of the third axis was an Nrf2 *Rmean* vector for each of *d* doses.

Nrf2-Srxn1 neural network $LSTM_{cv}$ output tensor preparation

Srxn1 *Rmean* vectors corresponding to the selected doses were 0-1 normalized using the Srxn1scaler.

An empty three dimensional tensor was generated using the numpy *zeros* function of shape (*d, len,* 1). This tensor was filled as follows: On each index of the first axis, the full second axis and the second index of the third axis was a Srxn1 *Rmean* vector corresponding to each of the selected doses, in order of the selected doses.

Nrf2-Srxn1 neural network $LSTM_{tp}$ input tensor preparation

Nrf2 *Rmean* vectors corresponding to the selected doses were 0-1 normalized using the Nrf2scaler.

An empty three dimensional tensor was generated using the numpy *zeros* function of shape (*d***len*, 1, 2).

This tensor was filled as follows: on the full first axis and the first index of the third axis were *timevec* vectors appended together for each of the doses used for training. On the full first axis and the second index of the third axis were the Nrf2 *Rmean* vectors appended together in order of the selected doses.

Nrf2-Srxn1 neural network $LSTM_{tp}$ output tensor preparation

Srxn1 *Rmean* vectors corresponding to the selected doses were 0-1 normalized using the Srxn1scaler.

An empty three dimensional tensor was generated using the numpy *zeros* function of shape (*d***len*, 1, 1).

This tensor was filled as follows: On the full first axis and the first index of the third axis were the Srxn1 *Rmean* vectors appended together in order of the selected doses.

Nrf2-Srxn1 neural network $LSTM_{dtp}$ input tensor preparation

Nrf2 *Rmean* vectors corresponding to the selected doses were 0-1 normalized using the Nrf2scaler.

Srxn1 *Rmean* vectors corresponding to the selected doses were 0-1 normalized using the Srxn1scaler.

Nrf2 *Rmean* vectors, Srxn1 *Rmean* vectors and *timevec* vectors were shortened to the length of (*len* -1) by deletion of the last data point.

An empty three dimensional tensor was generated using the numpy *zeros* function of shape: (*d**(*len*-1), 1, 3).

This tensor was filled as follows: On the full first axis and the first index of the third axis were the *timevec* vectors appended together in order of the selected doses.

On the full first axis and the second index of the third axis were the Nrf2 *Rmean* vectors appended together in order of the selected doses.

On the full first axis and the third index of the third axis were the Srxn1 *Rmean* vectors appended together in order of the selected doses.

Nrf2-Srxn1 neural network $LSTM_{dtp}$ output tensor preparation

Srxn1 *Rmean* vectors corresponding to the selected doses were 0-1 normalized using the Srxn1scaler. Srxn1 *Rmean* vectors were then differentiated using numpy's diff function. The resulting dSrxn1 vectors of length (*len* -1) were smoothed using the Savitzky-Golay filter (poly 5, window length 1501). dSrxn1 vectors were then scaled using the Diffscaler.

An empty three dimensional tensor was generated using the numpy *zeros* function of shape: (*d**(*len*-1), 1, 1).

This tensor was filled as follows: On the full first axis and the first index of the third axis were the smoothed and 0-1 normalized dSrxn1 vectors appended together in order of the selected doses.

Appendix 5: $LSTM_{dtp}$ integrator

The derivative time point configuration uses an integration function for its predictions. The function takes four inputs: an Nrf2 data vector, an initial Srxn1 data point, the initial time (in hours) *starttime* and the final time (in hours) *endtime*. For all tests the first index of the corresponding Srxn1 data was used as the initial Srxn1. At its core this integrator tracks four lists:
Timelist: A time list with its initial value being index *inittime* of the *timevec* vector.

$$inittime = \frac{len}{32} * starttime$$

Nrf2list: an Nrf2 list with its initial value being index *initNrf2* of the corresponding Nrf2 data input.

$$initNrf2 = \frac{len}{32} * starttime$$

Srxn1list: A Srxn1 list starting with an initial Srxn1 value
dSrxn1list: An (initially empty) dSrxn1 list.
dSrxn1 values were generated in a loop of length *Intrange* with iterations *i*:

$$Intrange = \left(\frac{len}{32} * (endtime - starttime)\right) - 1$$

For every iteration along *Intrange* the following steps were performed in order:

1. Indexing the last value of *Nrf2list* to gain a Nrf2 data point
2. 0-1 normalization of the Nrf2 data point
3. Indexing the last value of *Srxn1list* to gain a Srxn1 data point
4. 0-1 normalization of the Srxn1 data point
5. Tensor generation of shape (1, 1, 3)
6. Appending the last index of *Timelist* to the first index on the third tensor axis
7. Appending the Nrf2 data point to the second index on the third tensor axis
8. Appending the Srxn1 data point to the third index on the third tensor axis
9. Generating dSrxn1 predictions using the model.predict function on the input tensor
10. Inverse transforming the tensor using the Diffscaler
11. Generating the next Srxn1 value through addition of dSrxn1 to the value of the last *Srxn1list* index
12. Appending next Srxn1 value to *Srxn1list*
13. Indexing the next time value from *timevec* using index *timeindex*:

$$timeindex = i + 1 + \frac{len}{32} * starttime$$

14. Appending next time value to *Timelist*
15. Indexing the next Nrf2 value from the Nrf2 data vector using index *Nrf2index*:

$$Nrf2index = i + 1 + \frac{len}{32} * starttime$$

16. Appending next Nrf2 value to *Nrf2list*

Appendix 6: OSR ODE model

Code run in R version 3.6.3
Created by Isoude Kuijper (2020)

```
## LACDR model for oxidative stress response (OSR)
## Created by Isoude Kuijper and made available for BOO 2020
## Date: 06/04/2020
##
## This model contains 5 state variables, namely
## - Nrf2 (Nrf2)
## - nuclear Nrf2 (nNrf2)
## - Gsh (Gsh)
## - Srxn1 (Srxn1)
## - reactive metabolites (RM)

# First load the deSolve package. If this gives an error, you probably need
# to install it first with install.packages("deSolve")
library('deSolve')
library(tidyr)


### SECTION I: Build the model and assign the parameter values ###

# Function that defines the ddr model including the stress input function
osr <- function(t, inistate, parameters) {
  with(as.list(c(inistate, parameters)), {

    dS = -time_constant1 * S
    dNrf2 = exportNrf2 * nNrf2 - importNrf2 * Nrf2 + buildNrf2Base - (vMax
* Nrf2) / (km * (1 + RM / kiRM) + Nrf2)
    dnNrf2 = importNrf2 * Nrf2 - exportNrf2 * nNrf2
    dGsh = buildGshBase + buildGsh * nNrf2**hill_Gsh / (K_Gsh**hill_Gsh +
nNrf2**hill_Gsh) - conjFormToDegrGsh * conjForm * Gsh - conjForm * Gsh * RM
    dSrxn1 = buildSrxn1Base + buildSrxn1 * nNrf2**hill_Srxn1 /
(K_Srxn1**hill_Srxn1 + nNrf2**hill_Srxn1)  - degradSrxn1 * Srxn1
    dRM = buildRM - conjForm * Gsh * RM - degradRM * RM + S

    list(c(dS, dNrf2, dnNrf2, dGsh, dSrxn1, dRM))
  })
}

# Make a function that defines the parameters and
# runs the ddr model with those parameters
osrdose <- function(stress, tspan){
  if(1){
    pars.nostim = c(

      importNrf2 = 4.251955e-04,
      exportNrf2 = 4.672772e+00,
      buildNrf2Base = 2.297422e+01,
      vMax = 1.345040e+08,
      km = 3.816342e+02,
      kiRM = 1.338881e-06,
      buildGshBase = 5.530482e-02,
      buildGsh = 1.063481e+00,
      hill_Gsh = 6.499604e+00,
      K_Gsh = 4.580483e+00,
      conjForm = 1.240263e-04,
```

```
      buildSrxn1Base = 9.996659e-09,
      buildSrxn1 = 3.405193e+06,
      hill_Srxn1 = 6.003976e+00,
      K_Srxn1 = 4.833310e-01,
      conjFormToDegrGsh = 1.709325e+03,
      buildRM = 3.386074e-07,
      degradSrxn1 = 3.687709e-06, #train: 08, test3 = 02
      degradRM = 9.137437e-02,

      time_constant1 = 0.75
    )
  }
  inistate = c(
    Nrf2 = 2.455416e-03,
    nNrf2 = 2.234288e-07,
    Gsh = 2.608704e-01,
    Srxn1 = 2.710805e-01,
    RM = 3.704403e-06
  )
  #
  tspan_pre <-  seq(0, 1000, by = 1)
  ic <-  ode(
    y = c(S = 0, inistate), # stress is equal to 0
    times = tspan_pre,
    func = osr,
    parms = pars.nostim
  )

  out <-  ode(
    y = c(S = stress, ic[nrow(ic),3:ncol(ic)]), # stress is equal to
stresslevel
    times = tspan,
    func = osr,
    parms = pars.nostim
  )
  return(out)
}

### SECTION II: Run the ddr model ###

# Select the time span for which you want to run the model
tspan <-  seq(0, 49, by = 1)

# Run the model and save the simulation in variable 'xx'
xx <- osrdose(stress = 1, tspan) #test1 = 1, test2 = 1

# Rescale the model simulation
Nrf2 <- data.frame(xx)$Nrf2
nNrf2 <- data.frame(xx)$nNrf2
Gsh <- data.frame(xx)$Gsh
Srxn1 <- data.frame(xx)$Srxn1
RM <- data.frame(xx)$RM

### SECTION III: Plot the output ###

par(mfrow=c(2,3))

plot(tspan, data.frame(xx)$S, xlab="time (hours)", ylab="Stress (a.u.)",
ylim = c(0,1))
plot(tspan, Nrf2, xlab="time (hours)", ylab="Nrf2 (a.u.)")
plot(tspan, nNrf2, xlab="time (hours)", ylab="nNrf2 (a.u.)")
```

```
plot(tspan, Gsh, xlab="time (hours)", ylab="Gsh (a.u.)")
plot(tspan, Srxn1, xlab="time (hours)", ylab="Srxn1 (a.u.)")
plot(tspan, RM, xlab="time (hours)", ylab="RM (a.u.)")


Nrf2 %>% write.table(.,"E:/BFW/Master BPS/RP1/Neural Network/R - OSR
model/Training set 5/Nrf2_test3.txt")
Srxn1 %>% write.table(.,"E:/BFW/Master BPS/RP1/Neural Network/R - OSR
model/Training set 5/Srxn1_test3.txt")
```

LACDR Office
Gorlaeus laboratories
Einstein weg 55
2333 CC Leiden
www.lacdr.nl

Universiteit Leiden
The Netherlands

Discover the world at Leiden University