Forecasting Smog Clouds With Deep Learning: A Proof-Of-Concept

Anonymous Authors[1]

Abstract

In this proof-of-concept study, we conducted a multivariate time-series forecasting for concentration of nitrogen dioxide (NO_2), ozone (O_3), and (fine) particulate matter (PM_{10} & $PM_{2.5}$) with meteorological covariates between two locations in the Netherlands using various deep learning models, with a focus on long short-term memory (LSTM) and gated recurrent unit (GRU) architectures. In particular, we propose an integrated, hierarchical model architecture inspired by air pollution dynamics and atmospheric science that employs multi-task learning and is benchmarked by unidirectional and fully-connected models. Results demonstrate that, above all, the hierarchical GRU proves itself as a competitive and efficient method for forecasting the concentration of smog-related pollutants.

1. Introduction

1.1. Motivation

The presence of hazardous atmospheric chemicals characterises the phenomenon of air pollution. Although a number of physical activities (volcanoes, fire, etc.) may release different pollutants, anthropogenic activities are the head cause of environmental air pollution (Kampa & Castanas, 2008).

Adverse air pollution effects can range from skin irritation and difficulty in breathing to an increased risk of cardiac and respiratory illnesses, cancer, and mortality overall (Brunekreef & Holgate, 2002; Kampa & Castanas, 2008; Wong et al., 2008; Orellano et al., 2020). A recent addition is its direct link to COVID-19 morbidity and severity (Zorn et al., 2024). Indeed, as stated in (Lim et al., 2012), air pollution ranks high in the general disease burden attributable to environmental factors, with 3.1 million deaths in 2012 and 3.1% of disability-adjusted life years worldwide.

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review at ICML 2024 AI for Science workshop. Do not distribute.

Fundamentally, the air pollution problem and the extent to which it spreads is evident. Indeed, the air and its contaminants are everywhere and will remain inevitably inherent to human-nature interaction in the future. Positive notions present themselves, nonetheless: (1) humans, being the primary polluters, also possess the opportunity to act as the "primary purifiers"; and (2) comprehensive fundamental problem knowledge offers positive prospects for further advancing research (Vallero, 2014).

This problem motivates the development and application of data-driven forecasting models based on multiple neural network architectures, using contaminant and meteorological data to simulate and predict air pollution and smog. In particular, we consider the modelling of nitrogen dioxide (NO_2), ozone (O_3), and (fine) particulate matter (PM_{10} & $PM_{2.5}$) with various meteorological covariates as a first proof-of-concept (PoC). By employing these weather-predictive methods, this study aims to contribute incrementally to understanding air pollution dynamics and enhance environmental conditions for improved public health.

1.2. State-of-the-art

Traditional weather systems have evolved into sophisticated models that approximate real-world weather dynamics with increasing precision (Alley et al., 2019). The systems apply numerical weather prediction (NWP), a now ubiquitous, though computationally costly, numerical method grounded on physical first-principles (Bauer et al., 2015). While applying purely natural laws as boundary conditions for predictions is theoretically possible, it presents challenges in practice: the weather system is everywhere and contains numerous complex processes that make it computationally infeasible to provide these predictions with more than a highly simplified, parameterised value. Moreover, the non-linear dynamics, exemplified by the chaotic behaviour of turbulent flow, make predictions at high resolution—spatially, temporally, and/or across variables—a lasting challenge.

The emergence of data-driven methods presents a novel approach to abstracting physical processes embedded in the weather system. Machine learning (ML) models are adept at recognising complex patterns within large datasets with unparalleled efficiency—patterns that may represent relationships and correlations between atmospheric variables

and influences not yet understood by traditional physics.

A recently undertaken application of large-scale deep learning (DL) weather forecasting is FourCastNet by (Pathak et al., 2022). FourCastNet generates global forecasts orders of magnitude faster than traditional NWP with comparable or better accuracy. It herewith demonstrates the potential of data-driven methods to make significant progress in weather forecasting without explicitly considering the underlying (known) physical processes and equations. Implications are reducing costs of the traditional NWP and, more importantly, reducing the opportunity cost of inaccurate forecasts. Its scope does not, however, encompass predicting components directly related to air pollution or smog.

More closely related state-of-the-art studies, see (Masood & Ahmad, 2021) for a review, that distinctly forecast air pollution are (Freeman et al., 2018) and (Tao et al., 2019). The former performs a forecast of surface O_3 levels using a recurrent neural network (RNN) with long short-term memory (LSTM); its approach takes as input hourly-sampled meteorological data and O_3 itself, outputting a multivariate 72-hour horizon forecast. The latter, (Tao et al., 2019), highlights a composition of 1D convnets and the bidirectional gated recurrent unit (GRU) for a multivariate short-term prediction of $PM_{2.5}$. Both studies are consistent and relevant to the purpose of this PoC in that they use RNNs, take meteorological covariates as inputs, and consequently predict air pollution. Nonetheless, as much as O_3 and $PM_{2.5}$ are influential elements, a more complete air pollution and smog prediction requires consideration of a broader and combined set of air contaminants.

1.3. Contributions

Acknowledging the recent developments (Masood & Ahmad, 2021) and state-of-the-art, the LSTM and GRU establish themselves as the appropriate choice for modelling the sequential series of components in ambient (polluted) air. This insight steers us towards contributing an attempt at getting further command of the air pollution problem through a PoC of smog modelling with LSTM and GRU models. Specifically, the combined modelling of contaminants NO_2, O_3, $PM_{2.5}$, and PM_{10} is considered.

Ultimately, this research addresses the question: *"To what extent are models with the LSTM and GRU architecture capable of the multivariate timeseries forecasting of smog-related air components?"* It is found that the LSTM and GRU can indeed accurately forecast smog-related air components, thus providing an effective method for modelling and forecasting pollutants.

2. Background

2.1. Atmospheric interactions in air pollution

The very reality of contaminant concentrations changing over time naturally focuses attention on the question of how these changes originate and evolve. Whereas the origins and sources of pollution are reasonably well understood (Vallero, 2014; Saxena & Naik, 2018), much is still unknown about its dynamics and how it evolves—hence, the subject of this research. Especially from a ML perspective, understanding the specific relationships between variables, or features, is critical for efficient learning (Li et al., 2017).

How pollutants evolve is partly explicable by their interaction with their environment. As a result, a combined modelling of atmospheric variables can be justified. The following paragraphs briefly introduce the pollutants' interconnectivity and atmospheric interaction.

Foremost, the chemical interrelation of NO_2 and O_3. Nitrogen oxides (NO_x), a mixture of the colourless nitric oxide (NO) and reddish-brown, pungent NO_2, are (mostly anthropogenically-generated) primary pollutants. When in the presence of certain volatile organic compounds (VOCs) or another initiator or catalyst, NO can oxidate into NO_2 (Atkinson, 2000).

Continuing, PM is either emitted directly into the atmosphere (primary) or formed later (secondary) and is subject to air transport, cloud processing, and removal from the atmosphere (Seinfeld & Pandis, 2016). PM_{10} and $PM_{2.5}$ are interconnected as seen empirically (Velders et al., 2015) and naturally (Rhodes & Seville, 2024), given that their distinction is their size. With its relatively large size, PM itself experiences negligible chemical reactivity with the atmosphere compared to minor compounds such as NO_2 and O_3. Nonetheless, noting its susceptibility to transport, PM and other pollutants alike are responsive to airflow and dispersion in their ambient air environment—an environment amidst all meteorological influences, without yet considering factors such as geology and topology. Therefore, many, at least implicit, parameters are required to model PM and other air components reliably.

In short, NO_2, O_3, PM_{10}, and $PM_{2.5}$ are subject to influences from all dimensions and thus can be broadly modelled: for modelling pollution movements, pollution can be assumed to behave as air. Furthermore, pollutants are "internally" affected by each other and externally by the atmosphere, warranting a multivariate, integrated modelling approach.

2.2. Recurrent Neural Networks

Recurrent neural networks (RNNs) host cyclical connection pathways that, mathematically speaking, represent not functions but dynamical systems (DSs). RNNs have a net-

work state $\mathbf{x}(n)$ allowing some earlier input $\mathbf{u}(n')$ to leave its traces on output $\mathbf{y}(n)$, and, therewith, the current state is influenced by past states and input. In practical terms, RNNs are tailored for sequential, fixed order data, such as timeseries. Illustratively, when data is fed non-sequentially, e.g. today's weather prior to yesterday's, the internal state becomes confounded.

Formally, the transition of an RNN network state is given by the update equations:

$$\mathbf{x}(n) = \sigma(W\mathbf{x}(n-1) + W^{\text{in}}\mathbf{u}(n), \quad (1)$$

$$\mathbf{y}(n) = f(W^{\text{out}}\mathbf{x}(n)), \quad (2)$$

where $n = 0, 1, 2, ..., n_{\text{max}}$ are the time steps, W is a matrix containing the connections weights, W^{in} and W^{out} contain the weights from/to the input/output neurons, σ is a sigmoid function, and f a function wrapping the readout $W^{\text{out}}\mathbf{x}(n)$ (Jaeger, 2023b). In particular the activation function σ, which introduces non-linearity to the evolution of the internal state (1), enables RNNs to capture (long-term) non-linear dependencies in the data. Recurrent neural networks are trained with a special technique called backpropagation through time (BPTT) to handle their sequential nature. However, BPTT suffers from vanishing gradients (Hochreiter, 1998), making it difficult to learn long-term dependencies (Bengio et al., 1994).

Long short-term memory (LSTM) networks were introduced by Hochreiter, Schmidhuber, and Gers with the intention to solve this problem (Hochreiter, 1991; Hochreiter & Schmidhuber, 1997; Gers et al., 2000). They proposed a self-connected linear unit, the LSTM *memory cell*, with a constant error flow: in the absence of new input or error signals to the cell, the local error backflow remains constant, neither growing nor decaying (Gers et al., 2000). Thus, with the LSTM, the gradient is independent of T.

A more recently proposed recurrent unit is the gated recurrent unit (GRU) by (Cho et al., 2014). The GRU uses a similar approach to solving the vanishing gradient problem but simpler. It contains only two gates, the reset gate and update gate, making it easier to compute (and implement). The former controls the degree to which the previous hidden state influences the current, and the latter combines the LSTM input and forget gate into one. Its performance has shown to be on par with the LSTM, and, in some cases, can outperform it in terms of convergence in CPU time and in terms of parameter updates and generalisation (Chung et al., 2014).

As seen in Section 1.2, the gating mechanism also proves itself in air pollution-related applications (Freeman et al., 2018; Tao et al., 2019). Still, these studies predicted one contaminant only, while LSTMs are proven to be adequate for multivariate data (Che et al., 2018).

3. Methods

3.1. Data

The proposed forecasting experiment uses hourly-sampled data from 2016 to 2023 (RIVM, 2024; KNMI, 2024), which is available under an initiative of the Dutch government and the Dutch national meteorological service, the Royal Netherlands Meteorological Institute (KNMI). The data is accreditated under NEN-EN-ISO/IEC 17025 standards and is technically and substantively validated (and possibly rejected) before release (KNMI, 2023).

Figure 1. Utrecht area with markers indicating the AWS locations.

3.1.1. Spatiotemporal context

This experiment involves forecasting with data from two locations, a source location (A) and a target location (B). The source location is in Utrecht, the Netherlands, and here, pollutant data is combined with meteorologically related covariates, from a sensor located in De Bilt, to forecast pollutant data at the relatively northwestern target location in Breukelen. Their relative positions are best illustrated in Figure 1.

3.1.2. Inspection

Table 1 details the predictive variables used in this experiment, along with the initially considered meteorological parameters. The rationale and relevance of the meteorological parameters in the context of pollution prediction are discussed in Appendix A.1, coupled with an inspection.

3.2. Preprocessing

Preprocessing starts with tidying the raw data, followed by a train-validation-test split, feature engineering, normalisation, and ends with generating (input, output)-pairs.

Firstly, the raw data was cleaned to make it utilisable, for example by solving erronous (split) rows and columns, con-

Table 1. Predictive variables and initially considered meteorological variables (in alphabetic order). Some units are multiplied by 0.1 for simplification without losing significance.

Variable	*Unit*
Nitrogen dioxide (NO_2)	$\mu g\,m^{-3}$
Ozone (O_3)	$\mu g\,m^{-3}$
PM $\leq 10\,\mu m$ (PM_{10})	$\mu g\,m^{-3}$
PM $\leq 2.5\,\mu m$ ($PM_{2.5}$)	$\mu g\,m^{-3}$
Air pressure (AP)	hPa
Dew point temperature (DPT)	°C
Global radiation (GR)	$J\,cm^{-2}$
Maximum wind gust (MWG)	$m\,s^{-1}$
Mean wind direction (MWD)	$0 - 360°$
Mean wind speed (MWS)	$m\,s^{-1}$
Precipitation amount (PA)	mm
Precipitation duration (PD)	h
Sunshine duration (SD)	h
Temperature (T)	°C

verting encodings, extracting metadata, and the exclusion of data disqualified due to outliers. Next, there were missing values (Table 4). These were assumed to be missing completely at random (MCAR) and were imputed by linear interpolation.

Secondly, the tidy data is split into a training, validation, and testing set. Granting the heterogeneous nature of the data from year to year, but also the fact that forecasting the future using information from the future is fallacious, a sampling balance has to be struck. The resulting in the training/validation/testing split was 76.3%, 11.9%, 11.9%.

Thirdly, the newly acquired training set undergoes feature selection. As described in (Hall, 1999), good feature sets contain features that are highly correlated with the class, yet uncorrelated with each other. Thus, to assess the features—the pollutants and meteorological parameters listed in Table 1—their intercorrelations are assessed and compared to a threshold r_{th} using the absolute value Pearson correlation coefficient r_{xy}:

$$r_{xy} = \left| \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \right|, \quad (3)$$

where, given paired data $(x_i, y_i)_{i=1}^{n}$, n is the sample size, and $\bar{x}$, $\bar{y}$ are their sample means. It must be noted, however, that the calculation assumes linear relationships, heteroskedasticity, and a Gaussian distribution. The correlations are plotted in Figure 2.

Fourth, normalisation. Normalising promotes generalisation, stabilises gradients and the learning process, and can produce faster convergence (Ioffe & Szegedy, 2015). The

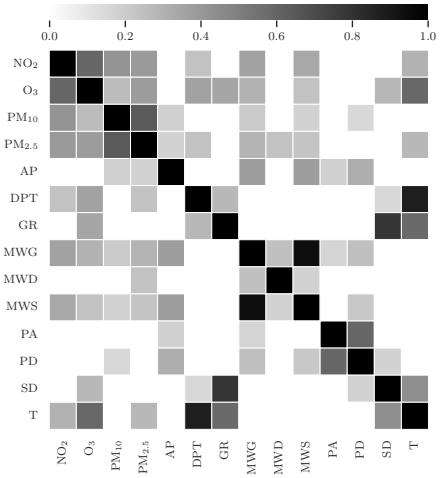

Figure 2. Coefficient matrix for the initially considered features. A threshold $\mathbf{r_{th}}$ for the absolute Pearson coefficient is set at $\mathbf{r_{th} = 0.15}$. When not met, the entry remains white.

selected features are normalised to a range of $[0, 1]$ with min-maxscaling

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}}, \quad (4)$$

where x_{min} and x_{max} are the minimum and maximum value for each feature in the training set.

Lastly in preparing the data for the models, pair generation. In order to obtain static input-output pairs from a discretised hourly-sampled temporal training sequence, one segments the input timeseries $(\mathbf{u}(n))_{n\in[0,N_u]}$ for Utrecht, and input timeseries $(\mathbf{y}(n))_{n\in[0,N_y]}$ for Breukelen with $N_u = N_y$, into sliding windows of length $l_{\text{in}} = 72$ and $h = 24$, respectively, obtaining input-output pairs $(\mathbf{u}_i, \mathbf{y}_i)_{i=1,...,P}$ consisting of input

$$\mathbf{u}_i = (\mathbf{u}(n_i), ..., (\mathbf{u}(n_i + l_{\text{in}})), \quad (5)$$

and output

$$\mathbf{y}_i = (\mathbf{y}(n_i + \delta), ..., (\mathbf{y}(n_i + \delta + h)), \quad (6)$$

where n_i represents the starting index of the i-th pair, P denotes the number of pairs as defined by $P = \lfloor \frac{N_u+1}{\Delta n} \rfloor$ with sampling step size Δn, and δ is defined as $\delta = l_{in} - 24 + 1$, meaning $\mathbf{y}_i$'s output is considered from the 48th hour on, plus a 1-hour window for the spatial prediction from $\mathbf{u}_i$ to $\mathbf{y}_i$. To expand on the latter, and as seen in (1) and (2), RNNs process values one-by-one, which for this case means for l_{in} iterations—δ, however, selects only the last 24 readouts for predicting and, thus, learning (facilitated by the loss function (7) discussed in Section 3.4). Step size Δn is set at $\Delta n = 24$ for computational efficiency, and because trial-and-error testing showed no or minor upside to a smaller Δn.

Essentially, this means that for each pair, an hour of pollutant concentrations at B will be predicted 24 times in sequence, with the preceding hours of A available as the ground for prediction. Thus, the data sets the models up to learn to model the pollutants using their covariates for the spatial prediction task from Utrecht to Breukelen.

3.3. Model architecture

The multivariate one-dimensional forecasting task of smog clouds, i.e., modelling the four pollutants from Utrecht to Breukelen, is taken on using six models: an ordinary multilayer perceptron (MLP), a hierarchical MLP (HMLP), an LSTM and GRU, and, as main contenders, a hierarchical LSTM (HLSTM) and GRU (HGRU). This section will outline the modelling types and set-ups, followed by their hyperparameter optimization procedure.

3.3.1. Types

As touched upon in Section 2.2, the MLP models approximate not DSs but functions. Where RNNs have a state $\mathbf{x}(n)$ allowing some earlier input $\mathbf{u}(n')$ to leave its traces on output $\mathbf{y}(n)$, MLPs learn to approximate a (nonlinear) function $f : \mathbb{R}^{L^0} \rightarrow \mathbb{R}^{L^k}$, where L^0 and L^k represent the neurons in the input- and output layer, and lack an explicit mechanism for retaining sequences over extended periods. In practice, they cannot utilise the sequence-spanning BPTT; they propagate errors solely through the network. Hence, the MLP and HMLP are less suited for this task than RNNs and serve as benchmarks.

In terms of their specific architecture, the input and output layer are of size $L^0 = 10$ (ten features) and $L^k = 4$ (four predictive variables). Unidirectional layers knit these together. For the MLP, these are standard fully-connected layers. Its counterpart, the HMLP, is of the type of hierarchical models—a term introduced in Section 2.2 which describes non-homogeneous, modular neural circuits. HNNs can, depending on the task, perform multi-task learning (MTL), a method whose principle goal is to improve generalisation performance (Caruana, 1997).

Furthermore, with HNNs, the hierarchical organisational structure is in hands of the model designer and offers an opening for a priori knowledge to be embodied in the neuronal arrangement as inductive bias or regularising factor, guiding the model in a preferred direction. In the context of this study, we aim to predict the four pollutants, each of which can be regarded as a distinct subtask. Recognising both the intercorrelations of the pollutants (as depicted, for example, in Figure 2) and the fact that they all live a life of their own, it seems reasonable to mirror this reality in a model's architecture. To achieve this, we employ one shared layer to establish shared representation and subsequently partition the network flow into a modular branch per subtask to reduce the interference between tasks. This design, including this nuanced regularising factor, confers HNNs a hypothetical advantage over fully-connected nets, which neglect an explicit internal-external balance.

Next are the RNNs. The RNNs use the PyTorch implementation of LSTM and GRU memory cells introduced in Section 2.2. The fully-connected RNNs are similar to the MLP, except for their gating mechanisms and recurrent synaptic connections, and vice versa for the hierarchical RNNs in relation to the HMLP.

Following up on identifying model types, hyperparameters reveal in more detail how these types are shaped into complete architectures. The following sections discuss how they are established.

3.3.2. Hyperparameter optimization

Hyperparameters can be used to control the behaviour of a learning algorithm and are not adapted by the algorithm itself (Goodfellow et al., 2016). For the DL models at hand, examples are the number of hidden layers and hidden units, the learning rate, choice of optimizer, and the regularisation term. An overview of the used hyperparameters per model can be found in Appendix C—this section will chiefly explore the methodology behind their selection.

To determine their values, a hyperparameter search procedure consisting of a grid search and cross-validation (CV) schemes is used. This procedure aims to find a hyperparameter configuration c with a minimised loss, while also testing c's generalisation capabilities. The loss is calculated on distinct validation sets generated by CV from all available training data, i.e. a concatenation of the training and validation set, to test this generalisation performance and prevent overfitting. Nested within the hyperparameter search and within the CV scheme, is the models' training algorithm, which, together with the loss, is specified in the next subsection, Section 3.4.

Continuing, the traditional grid search was used because of its ease suiting this PoC; it essentially does a brute-force search through the parameter space H. Here, H is defined as the Cartesian product of the finite sets S containing possible values for each parameter. Because H grows exponentially, a large S is not feasible, and smaller S are already computationally expensive. Hence, as measures, some initial trial runs were executed to get a feel of which options to include and the many models were computed on an HPC cluster.

Then, CV is run for each c, where c is a unique configuration within H. For the stateless MLPs, regular k-fold cross-validation, with $k = 5$, is used—with the perk of maximal data usage. RNNs, conversely, do have a state and allow memory trace of past sequences, as aforementioned in Section 2.2. A variation of k-fold CV, called sliding window

Table 2. List of the hyperparameters included in the grid search.

Hyperparameter	*Symbol*
Hidden layers	k
Hidden units	L^{κ}
Learning rate	μ
Learning rate, shared	μ_{shared}
Learning rate, branches	μ_{branch}
Weight decay	λ

CV, accommodates this: it samples training and validation sets—with, in contrast to pair generation, superposition of intervals—using a sliding window approach, thus not allowing validation of trained models with out-of-sample data directly from the past.

With these schemes, grid search with k-fold CV for the MLPs and sliding-window CV for the RNNs, values for the hyperparameters listed in Table 2 were determined—forging the model types into architectures. (Appendix C presents complete model architecture summaries.)

3.4. Training

In this section, we explain the training procedure used during hyperparameter optimization and the final training itself by defining the optimization goal and method, followed by some anti-overfitting measures. The final models are trained using the training and validation set created in Section 3.2.

To approximate a model m_θ parametrised by tuneable parameters collected in a vector θ, given a search space $\theta \in \Theta$ of target models Θ within the same architecture; training pairs $(\mathbf{u}_i, \mathbf{y}_i)_{i=1,\dots,P}$; and the mean squared error (MSE) loss function defined as

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \lambda \mid\mid \theta \mid\mid_2^2, \tag{7}$$

where n denotes sample amount, y the ground truth, and $\hat{y}$ the prediction—one has to solve the optimization problem

$$\theta_{\text{opt}} = \underset{\theta\in\Theta}{\text{argmin}} \frac{1}{P}\sum_{i=1,\dots,P} \text{MSE}(m_\theta(x_i), y_i), \tag{8}$$

where θ_{opt} denotes a model with a minimised empirical risk (Jaeger, 2023a;b). MSE punishes extreme values quadratically more, suiting the context of air pollution where extremes are of greater concern.

With an initial model $\theta^{(0)}$, initialised by the PyTorch-default Kaiming initialisation (He et al., 2015), optimization of $\theta^{(n)}$ is performed by the *Adam* (ADAptive Momentum) optimizer (Kingma & Ba, 2014). Adam differentiates itself from, e.g., stochastic gradient descent (SGD) by using momentum (Sutskever et al., 2013) and adaptive learning rates per parameter while only requiring first-order gradients and little memory (Kingma & Ba, 2014). This study uses its implementation in PyTorch. An implemented, assisting add-on is a scheduler: it reduces the learning rate by a factor of 0.1 when the validation loss stagnates for a set number of epochs (defined per model in Table 8).

Adam does its work everytime a batch B of 16 $(\mathbf{u}, \mathbf{y})$-pairs is passed. $B = 16$ was plainly adopted from (Masters & Luschi, 2018), who found smaller batch sizes $(2 \leq B \leq 32)$ to provide benefits in terms of convergence stability and overall test performance for a given number of epochs. Batches are randomly sampled (while in sequence order) from the available pairs, introducing stochasticity (and efficiency over, e.g., one-by-one calculation). Internally, this adds the batch dimension to the pairs, creating the tensors $[B, l_{in}, L^0]$ for $\mathbf{u}$ and $[B, h, L^k]$ for $\mathbf{y}$. When $\mathbf{u}$ is fed, the models spit out forecasts $\mathbf{y}'$ in the form of such tensor, which is subsequently compared to the ground truth $\mathbf{y}$ yielding the loss withwhich θ can be updated.

Updating θ, however, proceeds quite differently for the two main types of models. Whilst for the fully-connected models, this proceeds as usual with one optimizer updating θ, the modular models require a different approach. As they essentially consist of multiple core components (one shared layer, four branches) with different search spaces and convergence qualities, the process capitalises on this: all five components have their own optimizer and matched scheduler. In addition, they have two separate (initial) learning rates, as seen in Table 2 and Table 7. Distributing the learning tasks helps each model part stably reach an optimum.

Lighting this in terms of implementation, the shared and branched parts do epochs in turns, seeing all the batches separately while the other is *frozen*. Frozen, as in, the parameters cannot update but can infer. A con here is efficiency: the batches are passed through the model once more for every epoch.

When zooming out and looking at when learning should finalise, early stopping comes in: it finishes training when for some number of epochs (defined in Table 8) the validation loss does not decrease by $\geq 1 \times 10^{-5}$. Another anti-overfitting measure, or regulariser, is the $L2$-norm added to (7). As effect, larger weights are penalised and smaller weights are encouraged, preventing some set of weights dominating the model.

In summary, the training process seeks to find an optimal set of model weights θ_{opt}, and to regularise, the learning process early stops, the batches introduce stochasticity, the regularisation term balances weight values, and the hierarchical nets incorporate MTL. With these regularisation

steps, the training procedure should yield models fulfilling the ultimate objective of generalisation, tested in next section.

3.5. Evaluation

For evaluation of the models, the held-out test set is used. The test set is unseen by the models before evaluation to properly assess their generalisation capabilities. The predictions were first post-processed using inverse minmax-scaling, sampled in batches without shuffling to eliminate any randomness, and then evaluated using the root mean squared error (RMSE) and symmetric mean percentage error (sMAPE) metric, which both serve a different interpretation of model performance. In addition, the inference speed of each model is evaluated, as this is one of the unique advantages of data-driven methods over first-principle methods like NWP.

The RMSE, defined as

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}, \tag{9}$$

provides a measure of the average magnitude of the error with, due to its squaring operation, larger errors getting penalised more. This fits the context of smog modelling, where higher values are especially harmful.

The other metric, the sMAPE:

$$\text{sMAPE} = \frac{2}{n}\sum_{i=1}^{n}\frac{|y_i - \hat{y}_i|}{(|y_i| + |\hat{y}_i|)} \times 100, \tag{10}$$

is an accuracy measure based on percentage (or relative) errors, providing a scale-independent, well-interpretable metric. The sMAPE complements the RMSE by taking into account the individual and different distributions of the pollutants. Therefore, the metric allows for a fair relative comparison of the models' performance.

Finally, it is worth emphasising that, generally speaking, the RMSE serves a practical purpose because it tells about the deviation in $\mu\text{g}\,\text{m}^{-3}$ and has a quadratically progressive penalty. The sMAPE mainly fulfils a "scientific" purpose due to the possibility of comparing models, though the two metrics are not mutually exclusive. This idea acts as a guide to interpret the results meaningfully.

4. Results

Following training, the models are evaluated on out-of-sample data. It is found that the models provide an effective method for the modelling and forecasting of the pollutants. Quantitative results by RMSE and sMAPE are listed in Table 3.

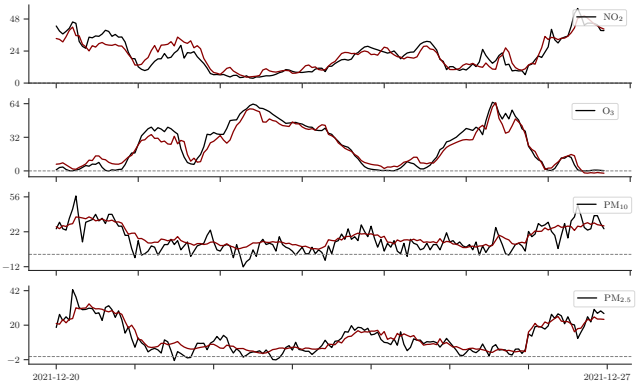

Figure 3. HGRU forecasts for NO_2, O_3, PM_{10}, and $PM_{2.5}$ taken for a week from the evaluation set. Black indicates the ground truth and maroon the forecasts. Dashed lines indicate zero.

Considering the subtask-specific lowest sMAPE values, NO_2 is predicted most accurately. Following NO_2 is O_3, then $PM_{2.5}$, and the models were least successful in predicting PM_{10}. Nonetheless, the lowest sMAPEs, as well as the RMSEs—**which are primarily generated by the HGRU**—confirm the models' suitability for forecasting the pollutants at B using data at A. Meanwhile, the models also differed in performance.

Measured by RMSE, the non-hierarchical fully-connected RNNs perform predominantly better than the MLPs, but also utilise many parameters to do so. Measured by sMAPE, they do too, despite HMLP's sMAPE ($M = 46.274, SD = 45.344$) being slightly inferior to the LSTM's ($M = 46.321, SD = 46.515$): a paired t-test with $\alpha < .05$ suggests there is no sufficient evidence to reject the null hypothesis of no difference, $t(8927) = 1.011, p = 3.12 \times 10^{-1}$.

Furthermore, the GRU yields the lowest errors of the non-hierarchical RNN models. The HLSTM ranks second, and, as per RMSE and sMAPE, the HGRU performs best, establishing the hierarchical models as the top performers. A paired t-test confirms the HGRU's ($M_{\text{RMSE}} = 5.468, SD_{\text{RMSE}} = 4.906, M_{\text{sMAPE}} = 44.519, SD_{\text{sMAPE}} = 44.519$) significant predictive ability on the testing set over the HLSTM ($M_{\text{RMSE}} = 5.633, SD_{\text{RMSE}} = 4.935, M_{\text{sMAPE}} = 44.981, SD_{\text{sMAPE}} = 45.850$) both by RMSE ($t(8927) = -5.922, p = 3.30 \times 10^{-9}$) and sMAPE ($t(8927) = -2.855, p = 4.32 \times 10^{-3}$), as well as on the other models. Moreover, for all individual pollutant subtasks by RMSE, and most by sMAPE, the HGRU exhibits the highest predictive precision, where it is only surpassed repeatedly with the sMAPE of the $PM_{2.5}$-subtask.

A visual representation of the HGRU's forecasts is with a lineplot, shown in Figure 3. Consistent with the numerical interpretation of the RMSE and sMAPE, the patterns of NO_2 and O_3 seem to be most closely captured. The PMs show

Table 3. Results of each model, evaluated and compared on performance (RMSE and sMAPE) and efficiency (inference speed and parameter count). The error metrics are calculated per pollutant and combined, with the lowest error in **bold**. Inference speed t_{inf} is the time in milliseconds for one inference of one 24-hour lead time prediction (processed on an Intel Core i7-8565U CPU, 8GB RAM, 64-bit OS).

Models	Performance										Efficiency	
	RMSE ($\mu g\,m^{-3}$)					**sMAPE** (%)					t_{inf} (ms)	**Param #**
	NO_2	O_3	PM_{10}	$PM_{2.5}$	Total	NO_2	O_3	PM_{10}	$PM_{2.5}$	Total		
MLP	6.63	7.53	7.82	4.85	6.71	35.89	41.90	65.24	53.15	49.04	27.2	17 604
HMLP	5.99	6.83	7.95	4.62	6.35	31.84	39.44	65.42	**48.00**	46.27	135.2	15 620
LSTM	5.97	6.39	7.48	4.32	6.04	32.09	38.09	63.40	51.70	46.32	14.4	572 640
HLSTM	5.36	6.57	6.60	4.00	5.63	**28.53**	38.83	60.80	51.76	44.98	18.7	72 244
GRU	6.01	6.18	6.84	3.94	5.74	32.62	38.46	61.15	49.67	45.47	47.9	363 360
HGRU	**5.35**	**6.01**	**6.59**	**3.92**	**5.47**	28.78	**36.97**	**59.92**	52.40	**44.52**	77.4	74 948

more short-term fluctuations, which are infrequently caught. This proves the most challenging with PM_{10}. Altogether, it can be stated that the HGRU is well equipped to use data at A for forecasting at B.

In terms of efficiency, the inference speed t_{inf} of the models, as also seen in Table 3, shows that efficiency is high: a 24-hour prediction is generated with negligible delay on a relatively inefficient processor. Counterintuitively, the model with the most parameters is the quickest, though the margins are small. As also discussed in (Pathak et al., 2022) and Section 1.2, the speeds, apart from the initial training cost, highlight the operational advantage of DL models over traditional first-principle methods: they are orders of magnitude faster and more efficient. Last to note on efficiency is that the best-performing models, the HLSTM and HGRU, require significantly fewer parameters (due to reduced parameter sharing) than the non-hierarchical RNNs as well.

5. Conclusions

In this paper, multivariate timeseries forecasting of smog clouds, represented by NO_2, O_3, PM_{10}, and $PM_{2.5}$ concentrations, using RNNs is conducted. Specifically, meteorological and pollution data at A is used to forecast air pollution levels at B. The most sophisticated models, the HLSTM and HGRU, are benchmarked with unidirectional and fully-connected DL architectures.

The research question, "To what extent are models with the LSTM and GRU architecture capable of the multivariate timeseries forecasting of smog-related air components?" is answered by the fact that the models are indeed highly adequate. Results demonstrate that, above all, the HGRU is suitable and competitive at this task. Reasons include the sequence-processing prowess of RNNs, a GRU's simplicity, and an integrated design streamlined to the very nature of the pollutants.

To sum up, our study contributes a PoC of smog cloud modelling using RNNs, providing a basis for advancements in pollution and weather forecasting to improve future public health.

Broader Impact Statement

Air pollution stands as a critical global challenge to humanity (UN, 2015). The rise of large-scale combustion and anthropogenic polluting activities has led to significant increases in air pollutant concentrations over the last century, leaving a heavy burden on human health (Kampa & Castanas, 2008). An unmistakable manifestation of these developments is the occurrence of smog: a noxious mixture of air pollutants that obstructs visibility and severely impairs human health in various ways (Brunekreef & Holgate, 2002). Given the detrimental effects, it is imperative to be able to predict when harmful pollutant levels might occur. This research proposes different methods to gain insight into air pollutant levels through timeseries forecasting and the application of multiple DNN architectures, notably including RNNs.

This research's limitations are summarised by simplifying measures to keep it within a scope appropriate for a PoC and by conceptually inherent limitations. Notable inherent limitations of this study include: the data being limited to merely two sensors, which fails to honour the complexity (e.g. the multidimensionality, emission sources, geographical features) of the modelled system; the non-stationarity of the data not being explicitly taken into account neither in preprocessing nor in model design; and modelling at a location where the air pollution and smog clouds problem is almost absent, thus limiting the direct impact.

Recurrent deep architectures offer a promising addition or augmentation to traditional NWP, given their adequacy and efficiency. Additionally, the dataset's minimal transformation makes real-time and continuous predictions possible.

References

Alley, R. B., Emanuel, K. A., and Zhang, F. Advances in weather prediction. *Science*, 363(6425):342–344, January 2019. ISSN 1095-9203. doi: 10.1126/science.aav7274.

Atkinson, R. Atmospheric chemistry of vocs and nox. *Atmospheric Environment*, 34(12–14):2063–2101, 2000. ISSN 1352-2310. doi: 10.1016/s1352-2310(99)00460-4.

Bauer, P., Thorpe, A., and Brunet, G. The quiet revolution of numerical weather prediction. *Nature*, 525 (7567):47–55, September 2015. ISSN 1476-4687. doi: 10.1038/nature14956.

Bengio, Y., Simard, P., and Frasconi, P. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, March 1994. ISSN 1941-0093. doi: 10.1109/72.279181.

Brunekreef, B. and Holgate, S. T. Air pollution and health. *The Lancet*, 360(9341):1233–1242, October 2002. ISSN 0140-6736. doi: 10.1016/s0140-6736(02)11274-8.

Caruana, R. Multitask learning. *Machine learning*, 28: 41–75, 1997.

Che, Z., Purushotham, S., Cho, K., Sontag, D., and Liu, Y. Recurrent neural networks for multivariate time series with missing values. *Scientific Reports*, 8(1), April 2018. ISSN 2045-2322. doi: 10.1038/s41598-018-24271-9.

Cho, K., Van Merriënboer, B., Bahdanau, D., and Bengio, Y. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

Freeman, B. S., Taylor, G., Gharabaghi, B., and Thé, J. Forecasting air quality time series using deep learning. *Journal of the Air and Waste Management Association*, 68(8):866–886, May 2018. ISSN 2162-2906. doi: 10.1080/10962247.2018.1459956.

Gelbard, F. and Seinfeld, J. H. The general dynamic equation for aerosols. theory and application to aerosol formation and growth. *Journal of Colloid and Interface Science*, 68(2):363–382, 1979.

Gers, F. A., Schmidhuber, J., and Cummins, F. Learning to forget: Continual prediction with lstm. *Neural computation*, 12(10):2451–2471, 2000.

Goodfellow, I., Bengio, Y., and Courville, A. *Deep learning*. MIT press, 2016.

Hall, M. A. *Correlation-based feature selection for machine learning*. PhD thesis, The University of Waikato, 1999.

He, K., Zhang, X., Ren, S., and Sun, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

Hochreiter, S. Untersuchungen zu dynamischen neuronalen netzen. *Diploma, Technische Universität München*, 91 (1):31, 1991.

Hochreiter, S. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116, 1998.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Holton, J. R. and Hakim, G. J. *An introduction to dynamic meteorology*. Academic press, 2012.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pp. 448–456. pmlr, 2015.

Irwin, J. and Williams, M. Acid rain: Chemistry and transport. *Environmental Pollution*, 50(1–2):29–59, 1988. ISSN 0269-7491. doi: 10.1016/0269-7491(88)90184-4.

Jaeger, H. Lecture Notes: Machine Learning. `www.ai.rug.nl/minds/uploads/LN_ML_RUG.pdf`, 2023a. Accessed on 18-04-2023.

Jaeger, H. Lecture Notes: Neural Networks. `www.ai.rug.nl/minds/uploads/LN_NN_RUG.pdf`, 2023b. Accessed on 02-07-2023.

Kampa, M. and Castanas, E. Human health effects of air pollution. *Environmental pollution*, 151(2):362–367, 2008.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

KNMI. Luchtkwaliteitsdata als csv bestanden, versie 7, 2023. URL `data.rivm.nl/data/luchtmeetnet/readme.pdf`. Accessed on 14/10/2023.

KNMI. KNMI dataplatform, 2024. URL `dataplatform.knmi.nl/`. Accessed on 15/03/2024.

Li, J., Cheng, K., Wang, S., Morstatter, F., Trevino, R. P., Tang, J., and Liu, H. Feature selection: A data perspective. *ACM computing surveys (CSUR)*, 50(6):1–45, 2017.

Lim, S. S., Vos, T., Flaxman, A. D., Danaei, G., Shibuya, K., Adair-Rohani, H., AlMazroa, M. A., Amann, M., Anderson, H. R., Andrews, K. G., et al. A comparative risk assessment of burden of disease and injury attributable to 67 risk factors and risk factor clusters in 21 regions, 1990–2010: a systematic analysis for the global burden of disease study 2010. *The lancet*, 380(9859):2224–2260, 2012.

Masood, A. and Ahmad, K. A review on emerging artificial intelligence (ai) techniques for air pollution forecasting: Fundamentals, application and performance. *Journal of Cleaner Production*, 322:129072, November 2021. ISSN 0959-6526. doi: 10.1016/j.jclepro.2021.129072.

Masters, D. and Luschi, C. Revisiting small batch training for deep neural networks. *arXiv preprint arXiv:1804.07612*, 2018.

Orellano, P., Reynoso, J., Quaranta, N., Bardach, A., and Ciapponi, A. Short-term exposure to particulate matter (pm10 and pm2.5), nitrogen dioxide (no2), and ozone (o3) and all-cause and cause-specific mortality: Systematic review and meta-analysis. *Environment International*, 142:105876, September 2020. ISSN 0160-4120. doi: 10.1016/j.envint.2020.105876.

Pathak, J., Subramanian, S., Harrington, P., Raja, S., Chattopadhyay, A., Mardani, M., Kurth, T., Hall, D., Li, Z., Azizzadenesheli, K., et al. Fourcastnet: A global data-driven high-resolution weather model using adaptive fourier neural operators. *arXiv preprint arXiv:2202.11214*, 2022.

Rhodes, M. J. and Seville, J. *Introduction to particle technology*. John Wiley & Sons, 2024.

RIVM. RIVM luchtmeetnet datasets, 2024. URL `data.rivm.nl/data/`. Accessed on 15/03/2024.

Saxena, P. and Naik, V. *Air pollution: sources, impacts and controls*. Cabi, 2018.

Seinfeld, J. H. and Pandis, S. N. *Atmospheric chemistry and physics: from air pollution to climate change*. John Wiley & Sons, 2016.

Sutskever, I., Martens, J., Dahl, G., and Hinton, G. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pp. 1139–1147. PMLR, 2013.

Tao, Q., Liu, F., Li, Y., and Sidorov, D. Air pollution forecasting using a deep learning model based on 1d convnets and bidirectional gru. *IEEE Access*, 7:76690–76698, 2019. ISSN 2169-3536. doi: 10.1109/access.2019.2921578.

UN. Transforming our world: the 2030 agenda for sustainable development, 2015. URL `https://sdgs.un.org/2030agenda`.

Vallero, D. A. *Fundamentals of air pollution*. Academic press, 2014.

Velders, G., Aben, J., Geilenkirchen, G., Den Hollander, H., van der Swaluw, E., de Vries, W., and van Zanten, M. *Grootschalige concentratie-en depositiekaarten Nederland: Rapportage 2015*. Rijksinstituut voor Volksgezondheid en Milieu RIVM, 2015.

Wong, C.-M., Vichit-Vadakan, N., Kan, H., and Qian, Z. Public health and air pollution in asia (papa): A multicity study of short-term effects of air pollution on mortality. *Environmental Health Perspectives*, 116(9):1195–1202, September 2008. ISSN 1552-9924. doi: 10.1289/ehp.11257.

Zorn, J., Simões, M., Velders, G. J., Gerlofs-Nijland, M., Strak, M., Jacobs, J., Dijkema, M. B., Hagenaars, T. J., Smit, L. A., Vermeulen, R., Mughini-Gras, L., Hogerwerf, L., and Klinkenberg, D. Effects of long-term exposure to outdoor air pollution on covid-19 incidence: A population-based cohort study accounting for sars-cov-2 exposure levels in the netherlands. *Environmental Research*, 252:118812, July 2024. ISSN 0013-9351. doi: 10.1016/j.envres.2024.118812.

A. Data insights

This Appendix section provides some additional insight into the data by exploring the meteorogical variables and discussing the correlations of all features. In addition, an overview of the data availabilities is displayed, some data statistics are presented, and an extraordinary outlier is visualised.

A.1. Exploration of meteorological data

This subsection provides an overview of all the initially considered meteorological variables accompanied by an explanation of why these variables might be helpful for modelling smog clouds, i.e. the pollutants NO_2, O_3, PM_{10}, $PM_{2.5}$. Important to emphasise is that not all of these rationales have held up in feature selection (or, worded differently, showed in the data).

It is important to stress that besides the individual characteristics of the pollutants, they are located in the tropospheric sky, and have such a low mass they can be assumed to behave as air in terms of their interaction with large-scale meteorological processes. We will go through the variables from top to bottom to explain their relevance:

- **Air pressure** can indicate dispersion and transport of large air currents, for example by low and high-pressure areas, and thus also influence the air currents of pollutants (Holton & Hakim, 2012).
- **Dew point temperature, precipitation sum, and precipitation amount** are indicators of atmospheric moisture content levels. These levels can say something about, for example, the rate of condensation (which, via nucleation, can lead to the formation of fine aerosols (PM) (Gelbard & Seinfeld, 1979)), any scavenging and cleansing of the air by rainfall (lowering the pollutant concentrations) (Vallero, 2014), and the formation of acid rain (where acidic gases SO_2 and NO_x that are (related to) the predictive variables, get washed out, thus lowering the concentrations (Irwin & Williams, 1988)).
- **Global radiation and sunshine**, which signify the presence of solar energy in the form of photons, serve as fundamental drivers of low-entropy energy input on Earth.
- **Temperature** is an essential factor in chemical processes seen by its role as accelerator in the formation of secondary pollutants. In addition, temperature plays a role in atmospheric stability, with, for example, (suddenly) high temperatures signifying increased convective activity. Furthermore, temperature influences state changes, and is also tightly connected with global radiation and sunshine, therewith also indirectly contributing to their effects. For more context on atmospheric chemistry and physics, refer to the extensive (Seinfeld & Pandis, 2016).
- **Mean wind direction, mean wind speed, and maximum wind gust** all tell about the wind's properties, which in turn carries the pollutants through the atmosphere. In context of the experiment, wind direction, for example, tells about the relative directional relationship between A and B. Out of the pollutants, the wind especially plays a role for the PMs, as they have a bigger surface.

A.2. Data availability

Here is a short summary of the availability of the data used in the experiment. Missing data was interpolated with linear interpolation.

Table 4. Data availability percentage for the modelled pollutants for each year. The meteorological abbrevations are defined in Table 1. The meteorological data was completely available for all years—for the pollutants, it was not. Given the strict procedures by the KNMI (KNMI, 2023), this is no surprise.

	NO_2	O_3	PM_{10}	$PM_{2.5}$	AP	DP	MWD	MWS	SD	T
2017	97.64%	96.85%	97.62%	99.10%	100%	100%	100%	100%	100%	100%
2018	99.67%	98.52%	97.70%	99.29%	100%	100%	100%	100%	100%	100%
2020	98.60%	98.05%	99.34%	99.31%	100%	100%	100%	100%	100%	100%
2021	99.67%	98.55%	98.88%	99.29%	100%	100%	100%	100%	100%	100%
2022	96.90%	97.62%	95.56%	99.62%	100%	100%	100%	100%	100%	100%
2023	98.60%	97.15%	98.46%	97.97%	100%	100%	100%	100%	100%	100%

A.3. Data allocation and quantity

This section provides transparency on how much data was used and in what proportions. Table 5 shows the number of hours of data before pair generation, and Table 6 the data after pair generation.

Table 5. Hours of data for each feature per year in the training, validation, and testing sets before pair generation, illustrating the data balance between the different sets, and their amounts. Divide these by 24 for the amount of days. (Meteorological abbrevations are defined in Table 1).

	NO_2	O_3	PM_{10}	$PM_{2.5}$	AP	DP	MWD	MWS	SD	T
Train '17	3648	3648	3648	3648	3648	3648	3648	3648	3648	3648
Train '18	3648	3648	3648	3648	3648	3648	3648	3648	3648	3648
Train '20	3648	3648	3648	3648	3648	3648	3648	3648	3648	3648
Train '21	2640	2640	2640	2640	2640	2640	2640	2640	2640	2640
Train '22	2640	2640	2640	2640	2640	2640	2640	2640	2640	2640
Validation '21	504	504	504	504	504	504	504	504	504	504
Validation '22	504	504	504	504	504	504	504	504	504	504
Validation '23	1512	1512	1512	1512	1512	1512	1512	1512	1512	1512
Test '21	504	504	504	504	504	504	504	504	504	504
Test '22	504	504	504	504	504	504	504	504	504	504
Test '23	1512	1512	1512	1512	1512	1512	1512	1512	1512	1512

Table 6. Table with numerical descriptions of the used datasets, after pair generation performed in Section 3.2 (with a Δn of only 24 hours). Due to the overlapping nature of the pair generation algorithm, "more" usable data was generated compared to the original data. The amount of pairs P is displayed, the total amount of hours, total datapoints, datapoints passed through the model as input $\mathbf{u}$, and the ground truth $\mathbf{y}$ datapoints used for the loss function during training, giving an indication of the amount of computations needed for one training epoch. (With the "optimal" $\Delta n = 1$, the training set would grow to the impractical amount of 12,847,104 datapoints.)

	P	hrs_{total}	n_{total}	$n_{\mathbf{u}}$	$n_{\mathbf{y}}$
Training set	656	47 232	535 296	472 320	62 976
Validation set	93	6696	75 888	66 960	8928
Testing set	93	6696	75 888	66 960	8928

B. Training insights

The subplots in Figure 4 show the training and validation loss development during final training of the six models. Figure 5 shows how both the shared and branched part of the HLSTM contributed to its training loss.

Training the models took an hour maximum, using the hyperparameters listed in Table 7 and 8 and processed locally on an Intel Core i7-8565U CPU, 8GB RAM, 64-bit OS.

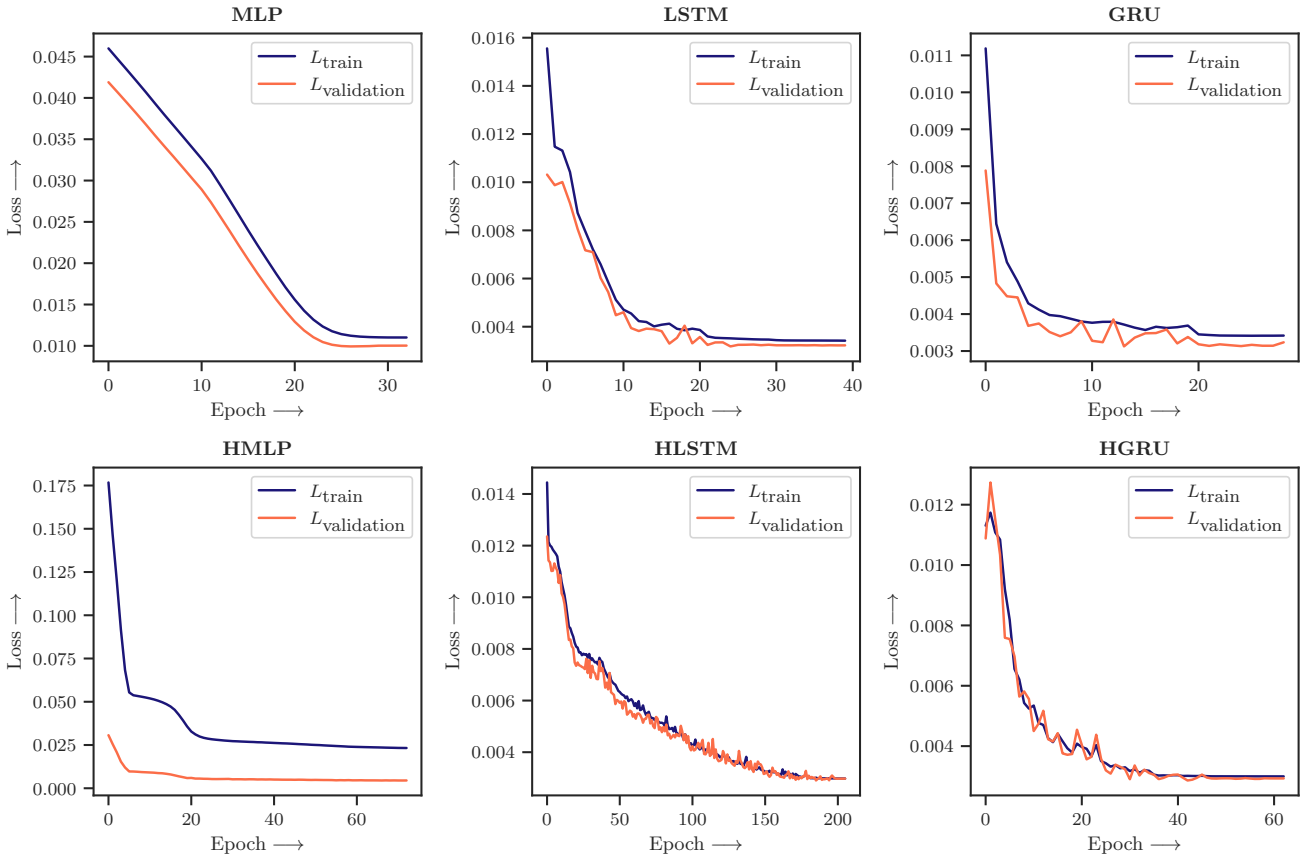

Figure 4. Loss plots for all models, showing the training versus validation losses over epochs.

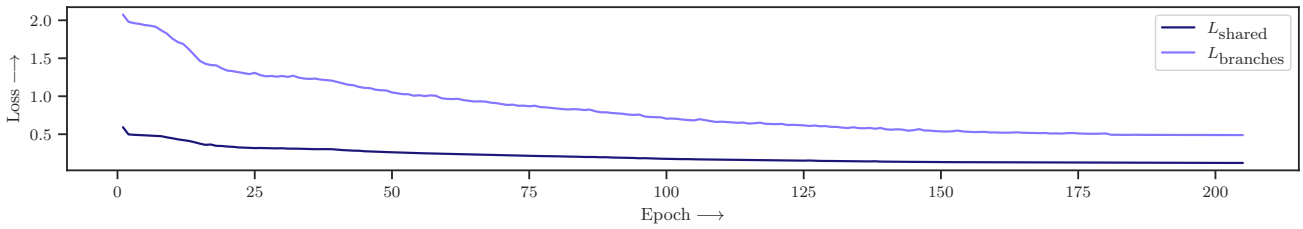

Figure 5. Training loss plotted of the shared and branched part of the HLSTM. For illustrative purposes, the first epoch is left out from the plot. Both model parts have different complexities (see Section 3.3), causing their learning process to be different as well. The branches were more complex, causing its learning process to be less stable, visible by the small ”bumps” in its descend.

C. Architecture details

This Appendix section provides some additional detail on the employed architectures by specifying the used hyperparameters and model parameters. In all of the model summaries, pass size denotes the size of a forward/backward pass in megabytes (MB). All activation functions employed are ReLU, including the readout. The high number of parameters for the RNNs are caused by the BPTT procedure, with the number being less for the hierarchical recurrent nets due to reduced parameter sharing.

Table 7. Overview of the hyperparameters that were determined through grid search and consequently used in the models. Their abbrevations are listed in Table 2. Also, note that the fully-connected models have just one learning rate, while the hierarchical models have two: one for their shared layer and one for the optimizers of each of their branches. Another thing to note here is that the ratio between these two, μ_{shared} and μ_{branch}, is equivalent to k. This was done, after lots of test runs, with the idea of a "power ratio" between the two: the branches needed a higher μ to let them converge in harmony with the shared layer. Another reason was that by interlinking the two, H was significantly reduced. A last thing to note is λ of the HLSTM being zero. During training, the HLSTM struggled to get momentum and to start learning, resulting in the hyperparameter search choosing a model with optimal flexibility—a regularisation term of zero.

	k	L^{κ}	μ	μ_{shared}	μ_{branch}	λ
MLP	4	64	1×10^{-5}			1×10^{-5}
HMLP	7	64		1×10^{-4}	7×10^{-4}	1×10^{-5}
LSTM	6	112	1×10^{-3}			1×10^{-6}
HLSTM	7	48		1×10^{-4}	7×10^{-4}	0
GRU	4	128	1×10^{-3}			1×10^{-5}
HGRU	4	64		1×10^{-3}	4×10^{-3}	1×10^{-7}

Table 8. Overview of other training settings (or "hyperparameters") that were determined through trial-and-error (and not through exhaustive search). All models used the Adam optimizer, reduced their learning rates when the validation loss reached a plateau (ReduceLROnPlateau), had a batch size ($|B|$) of 16, and used k = 5 in their k-fold cross-validation schemes. The MLPs had a patience of 6 and the RNNs of 15 to accomodate for their differences in convergence speed.

	optimizer	$\mu_{\text{scheduler}}$	patience	$\lvert B\rvert$	k_{folds}
MLP	Adam	ReduceLROnPlateau	6	16	5
HMLP	Adam	ReduceLROnPlateau	6	16	5
LSTM	Adam	ReduceLROnPlateau	15	16	5
HLSTM	Adam	ReduceLROnPlateau	15	16	5
GRU	Adam	ReduceLROnPlateau	15	16	5
HGRU	Adam	ReduceLROnPlateau	15	16	5