Exploring interpretable LSTM neural networks over multi-variable data

With continuous advancements in technology that allow the development of complex models even on commercially available hardware, time series forecasting - one of the tasks that are currently at the forefront of machine learning research - reached new heights as far as prediction accuracies are concerned. Still, the general lack of explainability in deep learning models is not an aspect that can be easily overlooked, especially when physically significant results need to be extracted. Based on this, the present report reviews the concept of the Interpretable Multi-Variable LSTM neural network. The two versions of the model (IMV-Tensor and IMV-Full) are re-developed in PyTorch and their features are extended, with the main addition being that they are able to perform predictions in broader time windows. They are then evaluated using all of the features of the PM2.5 Beijing Dataset, and are compared as far as their training times and 1-hour-depth predictions are concerned. The IMV-Full version achieves superior results in shorter training times, although both versions outperform the corresponding ones from the original paper. For this reason, the IMV-Full version is trained to perform predictions in 2-hour time windows and achieves RMSE and MAE scores that are comparable to the ones of the original paper, even though the time-window used in our analysis was double. When it comes to interpretability, all models are consistent in recognizing the PM2.5 measurements as the most prominent feature, followed by the Temperature and the accumulated rainfall. Consistency is also observed when it comes to the models' results on the features' temporal importance.

I. Introduction

The problem of time series forecasting has been at the forefront of science and economics for centuries - from efforts to predict the intra-day variance of the environment's temperature or pressure (or, in general, weather forecasting), to attempts at estimating the stock market's daily trends (stock market predictions). However, it was only relatively recently that machine learning techniques started being applied in order to study time series: the paradigm shift in computer science towards what has come to be known as the field of machine learning was the result of a series of revolutionary technological advancements, despite the fact that the theoretical foundations were set as early as in 1943 with the conception of the mathematical neuron [1]. Up to that point, time series data were either treated using traditional regression methods, or required extensive problem-specific modeling.

While machine learning and especially sequential deep learning models such as RNNs [2, 3], LSTMs [4] and GRUs [5] led to the skyrocketing of prediction accuracies as far as time series forecasting was concerned, their black box approach was one of their most significant shortcomings. Especially in physics-related problems, such as weather forecasting [6], or seismic activity predictors [7], accurate results are not sufficient if there is no additional physical insight such as the importance of different features for each prediction, or temporal correlations between these features' values.

With this problem as a stepping stone, the present report aims to a perform a review of the so called Interpretable Multi-Variable LSTM (IMV-LSTM) Neural Network [8], which managed to bridge the gap between model explainability and accurate model predictions. This model was the state-of-the-art in 2019 as far as predictions on the dataset studied herein are concerned. In addition to this, it provided rich insight with respect to the average and temporal importance of each feature present in the dataset. Apart from reviewing the original paper, the IMV-LSTM model is re-developed in PyTorch [9] and extended with regards to specific features, such as its prediction window, which was limited to 1-hour predictions in the original paper.

The remaining sections of this paper are structured as follows. In Section II, a brief review of studies related to explainability in deep learning networks is presented, mainly focusing on post-analyzing approaches [10]–[13] and attention mechanisms [14, 15]. Section III gives some basic information on the used dataset and provides a detailed analysis of the data preprocessing stage. After a short summary of the relevant theoretical concepts, the model's architecture as well as some information about its PyTorch development is analyzed in Section IV. Emphasis is given on the two different implementations of the proposed model, the so called IMV-Tensor and IMV-Full versions. Subsequently, in Section V, the main results of the model's deployment are discussed and some important conclusions are drawn with regards to its performance, as well as the explainability of its results. These are summarized in Section VI and an outlook on future work is presented. Finally, Section VII lists each author's contributions to the total corpus of the present project.

II. Related Work

The interpretability of Artificial Neural Networks has been studied recently in the literature, following a number of interpretation methods. Research such as in [16] suggests that model interpretability and robustness are closely connected

and there has been a number of literature review studies [17] of relevant methods. Research on RNNs more specifically has mainly focused on attention methods and post-analyzing on trained models.

RNN attention-based methods have been thoroughly examined lately, with attention mechanisms mostly implemented across time steps and on the hidden states of the models [14, 15, 18]–[24]. In their work, [14] and [15] experimented with encoder networks and attention mechanisms on their models' hidden states. However, the hidden states combine information from all input features resulting in a biased attention when the corresponding variables' importance is measured. In [15] additional bias is added by the contribution coefficients on attention values. Additionally, [14, 18] and [15] weight input data by attentions, however, prediction performance could be impaired as the direction of correlation with the target variable is not taken into account. Moreover, it is uncommon for current attention based methods to provide variable-wise temporal interpretability.

Post-analyzing interpretation has also been studied in the literature [25, 26]. The authors in [11] introduced a family of rule-based, model-agnostic explanations that highlight part of the input sufficient for the classifier to make a prediction. Another approach is detailed in [27], where a test instance is picked for interpretation, then new perturbed samples are generated locally and corresponding predictions are made by the trained model on those. Subsequently, an interpretable model is trained, which is weighted by the proximity of the perturbed instances to the instance of interest. However, such perturbation-based approaches might generate samples that differ from the original distribution of the data. A part of the investigations reported in [13] and [10] corresponds to the exploration of the sensitivity of prediction with respect to the input features, which are analyzed by employing gradient-based methods. Moreover, the authors of [28] claim to have achieved better results than gradient-based related methods by extracting temporal importance scores over words or phrases of individual sequences by decomposing the memory cells of trained RNNs. The aforementioned methods focused in most cases on only one importance type and are computationally heavy, failing to boost the predictive performance.

In [29], a wavelet-based neural network structure is proposed and the authors introduce an importance analysis method that can quantify the importance of each middle layer to the final output of the model. In [30], it is found that piecewise linear neural networks (PLNNs) that adopt a piecewise linear activation function such as ReLU can be trained to select meaningful features that significantly improve interpretability. The authors of [31] introduce an RNN architecture composed of input-switched affine transformations with input-dependent recurrent weights in order to characterize the linear contribution of each input to the model predictions. However, this method could result in significant losses in prediction performance.The work presented in this report is focused on the exploration of the internal structure of LSTM towards learning both accurate forecasting and importance measures.

It is worth noting that tensorization and decomposition of RNNs' hidden states has also been explored in literature. In [32], matrices are introduced as crucial to neural network architecture and those matrices compose the input, hidden and output layers. The authors of [33] proposed Tensorized LSTM, in which the hidden states are represented by tensor, as an alternative to network widening and adding layers, where the former introduces additional parameters and the latter increases the total runtime. Finally, [34]–[36] proposed to partition the hidden layer into separated modules with different updates. However, the resulting hidden state tensors and update processes did not provide the desired interpretability due to the fact that the variable-wise correspondence is not maintained.

III. Dataset & Preprocessing

The dataset used in this work is the Beijing PM2.5, which is part of the Monash, UEA & UCR time series regression repository. This dataset contains a time series of 43,824 hourly PM2.5 data entries and the associated meteorological data in Beijing, China. PM2.5 describes the concentration of fine inhalable particles in the air, with diameters that are generally 2.5 micrometers or smaller. There is a total of 8 features in the dataset: hourly air pollutants concentration measurements (PM2.5), temperature, pressure, dew point, accumulated rainfall, accumulated snowfall, wind direction and wind speed measurements taken from 12 nationally controlled air quality monitoring sites. The air-quality data were collected by the Beijing Municipal Environmental Monitoring Center. The meteorological data in each air-quality site were matched with the nearest weather station from the China Meteorological Administration. The time period the data refers to spans over a period of four years, from January 1st, 2010 to December 31st, 2014.

The authors of the original paper have excluded the wind direction feature from their PM2.5 air pollutants prediction input feature space. However, we include it in the subsequent analysis in order to utilize all the information provided by the dataset and presumably boost the proposed models' prediction performance. As we will show in the results section (Section V), the wind direction seems to have a positive predictive contribution while also increasing interpretability of the model's variables' importance.

Research [39] suggests that short-term exposure to air pollution is associated with a significant increase in mortality. This is why the Clean Air Act requires the US Environmental Protection Agency (EPA) to review national air quality standards every five years to determine whether they need to be maintained or revised in order to protect public health. In Table I, we present a taxonomy of different air quality categories based on the Air Quality Index (AQI) values as revised by the EPA in 2012. The fine particle pollution PM2.5 concentrations are scaled from 0 to 500 to form the index values. This table can help understand subsequent prediction results and interpret

how prediction errors influence possible air quality category deviations.

AQI Category	Index Values	PM2.5 Breakpoints ($\mu g/m^3$)
Good	0 - 50	0.0 - 12.00
Moderate	51 - 100	12.1 - 35.4
Unhealthy for Sensitive Groups	101 - 150	35.5 - 55.4
Unhealthy	151 - 200	55.5 - 150.4
Very Unhealthy	201 - 300	150.5 - 250.4
Hazardous	301 - 400	250.5 - 350.4
	401 - 500	350.5 - 500

Table I. Air Quality Index breakpoints per the U.S. Environmental Protection Agency, 2012.

The dataset contains the 8 aforementioned features, all of which correspond to numerical data, except for the wind direction feature, which is categorical with 4 values: North-West (NW), North-East (NE), South-East (SE) and Sout-West (SW), all of which denote the wind's direction. Therefore, the first preprocessing step includes performing one-hot-encoding of this variable, thus generating 4 new binary variables. The dataset also contains some unassigned/null values for the target variable PM2.5, which are filled by using the last observed non-null value of the variable. Any values that remained not assigned after the operation, are then filled using a backward filling, which propagates the first observed non-null value backward until another non-null value is met. At the end of this process, there are no missing values for PM2.5, while preserving all observations unaltered.

We then proceed to split the dataset into training, validation and test sets. 60% of the dataset is maintained for training, 20% for validation and the remaining 20% for testing the final models. Then, using the training set, we perform a min-max scaling transformation on the training, validation and test set data. Each value, X, of a feature is transformed to X' based on the following:

$$X' = \frac{X - X_{\text{min}}^{\text{train}}}{X_{\text{max}}^{\text{train}} - X_{\text{min}}^{\text{train}}},$$

where $X_{\text{min}}^{\text{train}}$ and $X_{\text{max}}^{\text{train}}$ are the minimum and maximum values of each feature in the training set, respectively. This operation maps the training data exclusively in the $[0, 1]$ range. Validation and test data will be scaled as well, however, they are not bound to lie in the same range. Lastly, the data is cast to PyTorch Tensor format in order to be compatible with the PyTorch implementation used. A batch size of 64 observations is selected in order to construct appropriate data loaders that will feed the data batches to the implemented models.

IV. Model Description

Both IMV-LSTM (Full and Tensor) versions capture the main structure of the classic LSTM [4]. The key innovation which allows us to extract variable-wise interpretability is the use of multiple hidden states, one for each variable, instead of only one as opposed to the classic architecture of the LSTM. The main paradigm is the following: Assume we have $N - 1 (N = 11$ in our case) exogenous time series and a target series $\mathbf{y}$ of length T, where $\mathbf{y} = (y_1, \ldots, y_T)$. Moreover, assume that our input is a multi-variable series $\mathbf{X}_T = \{\mathbf{x}_1, \ldots, \mathbf{x}_T\}$, where $\mathbf{x}_t = (\mathbf{x}_t^1, \ldots, \mathbf{x}_t^{N-1}, y_t)$ with $\mathbf{x}_t^1, \ldots, \mathbf{x}_t^{N-1}$ being the independent variables and y_t the target variable at time step $1 \leq t \leq T$. As described in Section III, both the independent variables $\mathbf{x}_t^1, \ldots, \mathbf{x}_t^{N-1}$ as well as the target variable y_t are real numbers. However, since the generalization to higher dimensions has no extra difficulties, we will assume throughout this section that $\mathbf{x}_t^1, \ldots, \mathbf{x}_t^{N-1}, y_t$ are all multi-dimensional vectors defined on the same Euclidean space $\mathbb{R}^{d_0 \times 1}$. Now, given $\mathbf{X}_T$, our aim is to learn a non-linear mapping $\mathcal{F} : \mathbb{R}^{(N \times d_0) \times T} \rightarrow \mathbb{R}^{d_0}$ to predict the upcoming values of the target series, i.e. $\hat{y}_{T+1} = \mathcal{F}(\mathbf{X}_T)$.

Extending the ideas in [8] we define a non-negative integer k ($k = 0$ in the original paper) corresponding to the length of future predictions made by the model. In other words, we extend the original implementation in order to determine a non-linear mapping $\mathcal{F} : \mathbb{R}^{(N \times d_0) \times T} \rightarrow \mathbb{R}^{d_0 \times k}$ to predict the next $\hat{y}_{T+1}, \ldots \hat{y}_{T+1+k}$ values of the target sequence, i.e. $\mathcal{F}(\mathbf{X}_T) = (\hat{y}_{T+1}, \ldots, \hat{y}_{T+1+k})$, for $k \geq 0$. This is really important especially in cases where the target values correspond to measurements of vital importance. For example, in our case the time step t corresponds to hours and the target y_{t+1} is the value of $PM_{2.5}$ which refers to the atmospheric particulate matter (PM) that have a diameter of less than 2.5 micrometers. As discussed in Section III, exposure to air pollution might have severe impacts in human's health. Therefore, knowing the values of the $PM_{2.5}$ for longer time periods might be crucial.

Concerning the interpretability aspect of the model, we are interested in the variable importance and the variable-wise temporal importance of each variable separately. The former captures the importance of a variable when compared to other variables while the latter measures the importance of each variable separately with respect to the time steps t, $1 \leq t \leq T$. Formally, we represent the variable importance as a vector $\mathbf{I} \in \mathbb{R}_{\geq 0}^N$ for which $\sum_{n=1}^N \mathbf{I}_n = 1$ and the variable-wise temporal importance as a vector $\mathbf{T}^n \in \mathbb{R}_{\geq 0}^{T-1}$ for which $\sum_{k=1}^{T-1} T_k^n = 1$, for every variable $1 \leq n \leq N$. Elements of this vector are normalized (i.e. sum to one) and reflect the relative importance of the corresponding variable or time instant w.r.t. the prediction.

As stated above, the idea of IMV-LSTM is to make use of the hidden state matrix and develop an associated update scheme, such that each element (e.g. row) of the hidden matrix encapsulates information exclusively from a certain variable of the input. To this end, we denote by $\widetilde{\mathbf{h}}_t = (\mathbf{h}_t^1, \ldots, \mathbf{h}_t^N)^\top$ the hidden state matrix at time step t, where $\widetilde{\mathbf{h}}_t \in \mathbb{R}^{N \times d}$, $\mathbf{h}_t^n \in \mathbb{R}^d$. Now, the vector $\mathbf{h}_t^n$ captures the total contribution of the n-nth variable for the time steps $1, \ldots, t$. The input-to-hidden transition tensor is defined as $\mathbf{U}_j = (\mathbf{U}_j^1, \ldots, \mathbf{U}_j^N)$, where $\mathbf{U}_j \in \mathbb{R}^{N \times d \times d_0}$, $\mathbf{U}_j^n \in \mathbb{R}^{d \times d_0}$, d_0 is the dimension of individual variables at each time step and d the dimension

of the hidden states. In our case, since both the independent variables and the target variables are real numbers we have $d_0 = 1$. Moreover, the dimension of the hidden states in our implementation is equal to $d = 128$. The hidden-to-hidden transition is defined as $\mathbf{W}_j = (\mathbf{W}_j^1, \dots, \mathbf{W}_j^N)$, where $\mathbf{W}_j \in \mathbb{R}^{N\times d\times d}$ and $\mathbf{W}_j^n \in \mathbb{R}^{d\times d}$. As in the case of standard LSTM neural networks, IMV-LSTM has the input $\mathbf{i}_t$, forget $\mathbf{f}_t$, output $\mathbf{o}_t$ gates and the memory cell $\mathbf{c}_t$ in the update process. The operations handled in a hidden unit are the following: Given the newly incoming input $\mathbf{x}_t$ at time t and the hidden state matrix $\tilde{\mathbf{h}}_{t-1}$, the hidden state update is given by:

$$\tilde{\mathbf{j}}_t = \tanh\Big(\mathbf{W}_j \otimes \tilde{\mathbf{h}}_{t-1} + \mathbf{U}_j \otimes \mathbf{x}_t + \mathbf{b}_j\Big), \tag{1}$$

where $\tilde{\mathbf{j}}_t = (\mathbf{j}_t^1, \dots, \mathbf{j}_t^N)^\top \in \mathbb{R}^{N\times d}$ has the same shape as the hidden state matrix. Each element $\mathbf{j}_t^n$ corresponds to the update of the hidden state w.r.t. input variable n. The terms $\mathbf{W}_j \otimes \tilde{\mathbf{h}}_{t-1}$ and $\mathbf{U}_j \otimes \mathbf{x}_t$ capture the information from the previous hidden states and the information from the new input $\mathbf{x}_t$, respectively. By $\otimes$ we denote the tensor-dot operation; for example by $\mathbf{W}_j \otimes \tilde{\mathbf{h}}_{t-1}$ we mean the product of the tensors along the N axis, i.e. $\mathbf{W}_j \otimes \tilde{\mathbf{h}}_{t-1} = (\mathbf{W}_j^1 \mathbf{h}_{t-1}^1, \dots, \mathbf{W}_j^N \mathbf{h}_{t-1}^N)^T$, where $\mathbf{W}_j^n \mathbf{h}_{t-1}^n \in \mathbb{R}^d$. Depending on the update scheme of the gates and memory cells, we implement two different IMV-LSTM architectures. The IMV-Full and IMV-Tensor models.

The update scheme of gates and memory cells for the IMV-Full model can be summarized as follows:

$$\begin{bmatrix}\mathbf{i}_t\\ \mathbf{f}_t\\ \mathbf{o}_t\end{bmatrix} = \sigma\Big(\mathbf{W}\cdot\big(\mathbf{x}_t \oplus \mathrm{vec}(\tilde{\mathbf{h}}_{t-1})\big) + \mathbf{b}\Big) \tag{2}$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \mathrm{vec}(\tilde{\mathbf{j}}_t) \tag{3}$$

$$\tilde{\mathbf{h}}_t = \mathrm{matricization}(\mathbf{o}_t \odot \tanh(\mathbf{c}_t)), \tag{4}$$

where by $\mathrm{vec}(\tilde{\mathbf{h}}_{t-1})$ we denote the resulting vector after the row-wise concatenation of the columns of $\tilde{\mathbf{h}}_{t-1}$. Therefore, $\mathrm{vec}(\tilde{\mathbf{h}}_{t-1})$ is a vector in $\mathbb{R}^{D\times 1}$, where $D = Nd$. By $\oplus$ we denote the concatenation operation; e.g. $\mathbf{x}_t \oplus \mathrm{vec}(\tilde{\mathbf{h}}_{t-1})$ is the resulting vector after concatenating the columns of the matrix $\mathbf{x}_t$ and the vector $\mathrm{vec}(\tilde{\mathbf{h}}_{t-1})$ in a single vector in $\mathbb{R}^{D'\times 1}$, where $D' = N^2 d d_0$. The operator $\oplus$ is the element-wise multiplication and $\mathrm{matricization}(\mathbf{o}_t \oplus \tanh(\mathbf{c}_t))$ reshapes the vector $\mathbf{o}_t \oplus \tanh(\mathbf{c}_t) \in \mathbb{R}^{Nd}$ to a matrix with dimensions $N \times d$. As we can see from the previous equations; the Full model operates in similar fashion as the standard LSTM. The only difference lies at the construction of the hidden states, where in IMV-Full model we have one hidden state for every variable. Having calculated the hidden state update $\tilde{\mathbf{j}}_t$ in (1), the hidden states are concatenated into a single vector and the remaining process is essentially the same with the standard LSTM.

On the other hand, the IMV-Tensor model contains gates and memory cells for each variable separately. In other words, as opposed to the Full model, the Tensor model generates $\mathbf{i}_t, \mathbf{f}_t, \mathbf{o}_t$ and $\mathbf{c}_t$ for every variable. More formally, the operations in the hidden unit can be summarized by the following set of equations:

$$\begin{bmatrix}\mathbf{i}_t\\ \mathbf{f}_t\\ \mathbf{o}_t\end{bmatrix} = \sigma\Big(\mathcal{W} \otimes \tilde{\mathbf{h}}_{t-1} + \mathcal{U} \otimes \mathbf{x}_t + \mathbf{b}\Big)$$
$$\tilde{\mathbf{c}}_t = \tilde{\mathbf{f}}_t \odot \tilde{\mathbf{c}}_{t-1} + \tilde{\mathbf{i}}_t \odot \tilde{\mathbf{j}}_t$$
$$\tilde{\mathbf{h}}_t = \tilde{\mathbf{o}}_t \odot \tanh(\tilde{\mathbf{c}}_t).$$

Now, in order to determine the variable importance vector $\mathbf{I} \in \mathbb{R}^N_{\geq 0}$ and the variable-wise temporal vectors $\mathbf{T}^n \in \mathbb{R}^{T-1}_{\geq 0}$, we first define the variable attention and the variable-wise temporal attention by using a mixture attention mechanism. The mixture attention mechanism can be summarized as follows: temporal attention is first applied to the sequence of hidden states corresponding to each variable, so as to obtain the summarized history and importance of each variable separately. Then by using the history enriched hidden state of each variable, variable attention is derived to merge variable-wise states. More precisely, by introducing a discrete latent variable z_{T+1} over the set of values $\{1, \dots, N\}$ corresponding to N input variables, the mixture attention can be formulated as follows:

$$\begin{aligned} p(y_{T+1} \,|\, \mathbf{X}_T) &= \sum_{n=1}^N p(y_{T+1} \,|\, z_{T+1} = n, \mathbf{X}_T) \\ &\quad \cdot \mathbb{P}(z_{T+1} = n \,|\, \mathbf{X}_T) \\ &= \sum_{n=1}^N p(y_{T+1} \,|\, z_{T+1} = n, \mathbf{h}_1^n, \dots, \mathbf{h}_T^n) \\ &\quad \cdot \mathbb{P}(z_{T+1} = n \,|\, \tilde{\mathbf{h}}_1, \dots \tilde{\mathbf{h}}_T) \\ &= \sum_{n=1}^N p(y_{T+1} \,|\, z_{T+1} = n, \underbrace{\mathbf{h}_T^n \oplus \mathbf{g}^n)}_{\substack{\text{variable-wise}\\ \text{temporal attention}}} \\ &\quad \cdot \underbrace{\mathbb{P}(z_{T+1} = n \,|\, \mathbf{h}_T^1 \oplus \mathbf{g}^1, \dots, \mathbf{h}_T^N \oplus \mathbf{g}^N)}_{\text{Variable attention}}. \end{aligned} \tag{5}$$

The vector $\mathbf{g}^n$ is the temporal attention weighted sum of hidden states of variable n. It is given by

$$\mathbf{g}^n = \sum_{t=1}^N \alpha_t^n \mathbf{h}_t^n,$$

where the weights α_t^n are calculated by

$$\alpha_t^n = \frac{\exp(f_n(\mathbf{h}_t^n))}{\sum_{k=1}^T \exp(f_n(\mathbf{h}_\mathbf{k}^\mathbf{n}))}, \tag{6}$$

for every $1 \leq t \leq T$. The function f_n can be any function $f_n : \mathbb{R}^d \to \mathbb{R}$ that maps the vectors $\mathbf{h}_t^n$ to real numbers. In our implementation $f_n(\mathbf{h}_t^n)$ is given by

$$f_n(\mathbf{h}_t^n) = \tanh(\mathbf{h}_t^n \cdot \alpha_n^T + \alpha_n^b), \tag{7}$$

where α_n is a vector in $\mathbb{R}^{128\times 1}$ with each coordinate drawn from the normal distribution $\mathcal{N}(0,1)$ independently and α_n^b is a real number drawn also from $\mathcal{N}(0,1)$. The weights α_t^n in (6)

capture the overall temporal contribution of the n-nth variable for a window time period T. Obviously, the time step for which $t^* = \arg\max_{1\leq t\leq T} \alpha_t^n$ is the step that the n-nth variable contributed the most to the prediction of the target value y_{T+1}. To estimate the probability $p(y_{T+1} \,|\, z_{T+1} = n, \mathbf{h}_T^n \oplus \mathbf{g}^n)$ and preserve at the same time the nature of the problem at hand, i.e. the output must be in $\mathbb{R}$ and not restricted to $(0, 1)$, we calculate $p(y_{T+1} \,|\, z_{T+1} = n, \mathbf{h}_T^n \oplus \mathbf{g}^n)$ as $\phi_n(\mathbf{h}_T^n \oplus \mathbf{g}^n)$, where ϕ_n is a linear mapping from $\mathbb{R}^{256\times 1}$ to $\mathbb{R}$. Now, since the probability $\mathbb{P}(z_{T+1} = n \,|\, \mathbf{h}_T^1 \oplus \mathbf{g}^1, \ldots, \mathbf{h}_T^N \oplus \mathbf{g}^N)$ represents the overall contribution (normalized in $(0, 1)$) of the n-th variable with respect to other variables we use the softmax function to estimate it. More precisely, for every variable n we first calculate $f(\mathbf{h}_T^n \oplus \mathbf{g}^n)$, where $f : \mathbb{R}^{256\times 1} \to \mathbb{R}$ is a linear mapping and finally we apply the softmax function over the sequence of points $\{(f(\mathbf{h}_T^n \oplus \mathbf{g}^n)\}_{n=1}^N$ to get

$$\begin{aligned}&\mathbb{P}(z_{T+1} = n \,|\, \mathbf{h}_T^1 \oplus \mathbf{g}^1, \ldots, \mathbf{h}_T^N \oplus \mathbf{g}^N)\\ &= \frac{e^{f(\mathbf{h}_T^n \oplus \mathbf{g})}}{\sum_{k=1}^N e^{f(\mathbf{h}_T^k \oplus \mathbf{g})}}.\end{aligned} \tag{8}$$

As far as the learning phase is concerned, we denote by Θ the set of parameters of the whole network as well as the probabilities in (5) which we wish to estimate. Then, the learning paradigm can be summarized as follows: Given a set of M training sequences $\{\mathbf{X}_T\}$ and $\{y_{T+1}\}_M$, we aim to learn both Θ and importance vector $\mathbf{I}$ and $\{\mathbf{T}^n\}_N$ for prediction and insights into the data. In the theoretical framework, the authors in [8], using the Expectation-Maximization scheme, minimize the function

$$\begin{aligned}&\mathcal{L}(\Theta, \mathbf{I}) =\\ &\quad -\sum_{m=1}^M \Bigg[\mathbb{E}_{q_m^n}\Big(\log p(y_{T+1,m} \,|\, z_{T+1,m} = n, \mathbf{h}_T^n \oplus \mathbf{g}^n)\Big)\\ &\quad -\mathbb{E}_{q_m^n}\Big(\log \mathbb{P}(Z_{T+1,m} = n \,|\, \mathbf{h}_T^1 \oplus \mathbf{g}^1, \ldots, \mathbf{h}_T^N \oplus \mathbf{g}^N)\Big)\\ &\quad -\mathbb{E}_{q_m^n}\Big(\log \mathbb{P}(z_{T+1,m} = n \,|\, \mathbf{I})\Big)\Bigg],\end{aligned} \tag{9}$$

where by q_m^n we denote

$$\begin{aligned}q_m^n &= \mathbb{P}(z_{T+1,m} = n \,|\, \mathbf{X}_{T,m}, y_{T+1,m}; \Theta)\\ &\propto p(y_{T+1,m} \,|\, z_{T+1,m} = n, \mathbf{X}_{T,m}) \cdot \mathbb{P}(z_{T+1,m} = n \,|\, \mathbf{X}_{T,m})\\ &\approx p(y_{T+1,m} \,|\, z_{T+1,m} = n, \mathbf{h}_T^n \oplus \mathbf{g}^n)\\ &\quad \cdot \mathbb{P}(z_{T+1,m} = n, \,|\, \mathbf{h}_T^1 \oplus \mathbf{g}^1, \ldots, \mathbf{h}_T^N \oplus \mathbf{g}^N).\end{aligned} \tag{10}$$

Now, minimizing (9) leads to a closed form for the importance vectors $\mathbf{I}$ and $\{\mathbf{T}^n\}_N$; i.e. $\mathbf{I}$ can be calculated as

$$\mathbf{I} = \frac{1}{M}\sum_m \mathbf{q}_m,\ \mathbf{q}_m = (q_m^1, \ldots, q_m^n)^T, \tag{11}$$

and the sequence of vectors $\{\mathbf{T}^n\}_N$ by

$$\mathbf{T}^n = \frac{1}{M}\sum_m \boldsymbol{\alpha}_m^n,\ \boldsymbol{\alpha}_m = (\alpha_{1,m}^n, \ldots, \alpha_{T,m}^n)^T. \tag{12}$$

The closed forms in (11) and (12) allows us to dispense from the the complexity of $\mathcal{L}(\Theta, \mathbf{I})$ and use a more suitable loss function for the regression task at hand. To this end, we use as loss function the mean squared error which is calculated as

$$\mathcal{L}_{\text{MSE}}(\hat{y}_1, \ldots, \hat{y}_M; \Theta) = \frac{1}{2}\sum_{j=1}^M (\hat{y}_j - y_k)^2, \tag{13}$$

where the prediction $\hat{y}_{T+1}$ is obtained by the weighted sum of means as:

$$\hat{y}_{T+1} = \sum_{n=1}^N \mu_n \mathbb{P}(z_{T+1} = n \,|\, \mathbf{h}_T^1 \oplus \mathbf{g}^1, \ldots, \mathbf{h}_T^N \oplus \mathbf{g}^N), \tag{14}$$

with $\mu_n = p(y_{T+1} \,|\, z_{T+1} = n, \mathbf{h}_T^n \oplus \mathbf{g}^n)$ corresponding to the proportion of the contribution of the n-th variable to the target value.

The extension to include larger prediction steps comes without any particular difficulties. In the case where we wish to predict the next $y_{T+1}, \ldots, y_{T+1+k}$ target values, where $k > 0$, we simply run the same forward pass as described earlier for $k + 1$ steps and obtain a sequence of prediction points $\hat{y}_{T+1}, \ldots, \hat{y}_{T+1+k}$. Now, the MSE loss function in (13) has the form

$$\mathcal{L}_{\text{MSE}}(\hat{y}_{k,1}, \ldots, \hat{y}_{k,M}; \Theta) = \frac{1}{2}\sum_{j=1}^M \sum_{i=0}^k (\hat{y}_{k,j}^{(i)} - y_{k,j}^{(i)})^2, \tag{14}$$

where $\hat{y}_k = (\hat{y}_k^{(0)}, \ldots, \hat{y}_k^{(k)}) = (\hat{y}_{T+1}, \ldots, \hat{y}_{T+1+k})$ are the $k + 1$ predictions and $\hat{y}_k = (y_k^{(0)}, \ldots, y_k^{(k)}) = (y_{T+1}, \ldots, y_{T+1+k})$ are the $k + 1$ true targets.

V. Experimental Results & Discussion

The first part of the experimental procedure consists of the training setup for both the Tensor and Full versions of the IMV-LSTM. The optimizer used is Adam, with Mean Square Error as the loss function[1], since the problem at hand is a regression task. L2-regularization is also activated for the optimizer, in order to reduce the rate of the network's overfitting.

Another feature implemented for regularization purposes, which also reduces the network's training time, is the so-called early stopping mechanism[2]. At the end of each training epoch, t, for a model trained using early stopping, the average loss, L_t, is calculated by evaluating the network on the validation data and compared to the average validation loss of the previous epoch, L_{t-1}. If $L_t < L_{t-1}$, a copy of the model is saved locally on the disk (checkpoint) and the training continues. However, if an increase in average validation loss is spotted, i.e. $L_t > L_{t-1}$, a countdown process is initiated. Given a patience threshold, p, if the average validation loss during any of the next p epochs is calculated to be lower than L_{t-1}, the countdown is reset, a new checkpoint is created and the

[1]Recall the discussion in Section IV concerning the choice of the loss function in the algorithm's utilization.

[2]One of the reasons why Dropout layers were not included in the network's architecture, unlike in the original paper, is because extensive testing proved that the network does not tend to overfit due to the inclusion of this mechanism.

training continues. If, on the other hand, $L_{t+k} > L_{t-1}$ for all $k = 1, 2, \ldots, p$, the training process is halted and the trained model returns to its last checkpoint. This process allows the choice of an arbitrarily large number of training epochs, since there is a guarantee that training will be halted once the minimum validation loss - given a threshold - is calculated. In addition, it ensures that overfitting will not significantly impact the model's generalizability. The patience threshold p is chosen equal to 15 for the following experiments.

As far as the network's learning rate is concerned, a scheduler module is utilized (more specifically, PyTorch's `StepLR`) in order to decrease the learning rate by multiplying its current value with a factor of $\gamma = 0.9$ every 20 epochs. This is done so that the network can assimilate the more intricate details of the studied dataset after having learned most of the training data's volume. Still, the initial value for the learning rate remains one of the problem's hyperparameters.

Along with the L2-regularization's weight decay, the model's initial learning rate is tuned using a triple-stage grid search. During the first stage, 10 randomly chosen equidistant values (in logarithmic scales) are chosen for each hyperparameter, thus forming a 10×10 grid. The values chosen belong in the range $\left[10^{-6}, 10^{0}\right]$ for the weight decay and $\left[10^{-4}, 10^{-1}\right]$ for the initial learning rate. For each hyperparameter pair, each LSTM network is trained on the training set until early stopping is activated and is subsequently evaluated on the validation set using the Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) as metrics. The hyperparameter pair for which the lowest validation error is achieved becomes the center of a new 3x3 grid which is the initial state of the second stage, as depicted in Fig. 1.

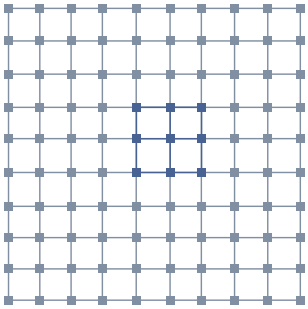

Fig. 1. A depiction of the grid at the end of the grid search's first-stage, during the process of hyperparameter tuning.

The resulting 3x3 grid is then transformed into a 5x5 grid, by considering the midpoints between the grid's nodes and the process of training and validation is repeated. The third stage is identical to the second stage and the final pair of the hyperparameters is utilized for the experiments using the two types of IMV-LSTM networks. The optimal values for the initial learning rate and the weight decay turn out to be $8{\cdot}10^{-4}$ and $3{\cdot}10^{-5}$, respectively, for both types of IMV-LSTM networks (Tensor and Full).

All experiments are performed using the optimal values for the two hyperparameters that are related to the networks' optimizer, as well as batches of data containing 64 items each. The first experiment corresponds to a series of 10 training-evaluation cycles (in order to be able to draw safer conclusions in a more robust manner) for the IMV-Tensor model with a single-prediction time window and the model's performance is measured as the average of the ten cycles. Obviously, the model's final performance (calculated as MAE and RMSE losses on the test set) is not the only focus of the experiment. The importance of each feature is also a crucial aspect of the present analysis, during both the training and the evaluation process. Fig. 2 depicts the experimental viewpoint for the two types of feature importance metrics that were presented in Eqs. (11), (12) of Section IV.

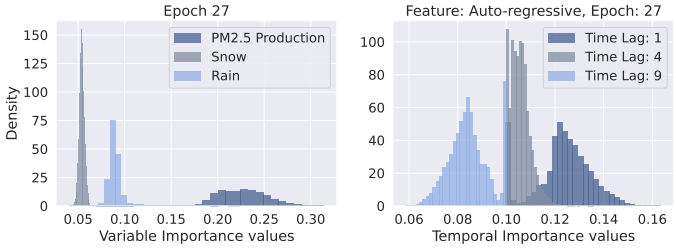

Fig. 2. Histograms of the importance for different features (left) and the temporal importance for a single feature (right) at the end of a randomly chosen training epoch.

Each of the two graphs concerns the training process of the model during one of the ten cycles and is acquired at the end of a randomly chosen epoch (epoch No. 27 here), using the network's predictions for the validation set's data. The reason why both graphs depict histograms is because both importance metrics are acquired for each prediction (or, equivalently, for each sample point's feature vector) separately. Therefore, the calculated importance of each feature may vary between different feature vectors, which are used for different predictions.

As far as the histograms of the left graph of Fig. 2 are concerned, they correspond to the calculated variable importance per feature, i.e. the degree to which each feature influences the model's predictions, for three selected features. The values of this metric are normalized such that the sum of all features' importance values per prediction is equal to 1 (100%) - see Eqs. (11) and (12). For the three features chosen, it is clear that PM2.5 measurements are the most important feature by a large margin, however its importance per prediction varies from $\sim$ 18% to $\sim$ 28%, which is why its histogram is somewhat spread over the horizontal axis. On the other hand, the accumulated snowfall and rainfall measurements appear to be in general less important, since they are centered around 4% and 8%, respectively. Nonetheless, their histograms are very localized with sharp peaks, an indication that their importance does not significantly vary between different predictions.

The histograms of the right graph of Fig. 2 correspond to the calculated variable-wise temporal importance per feature: the LSTM model's input is a series of 10 measurements acquired over a window of 10 hours for each feature. Each feature's

temporal importance corresponds to a measure of each of these 10 measurements' influence on the final prediction. This importance metric is also normalized such that a single feature's temporal measurements' influences sum to 1. The feature selected in this graph is PM2.5 measurements[3] and it becomes evident that its most recent values have the highest influence on the prediction, as expected.

What is common in both of these graphs is that information cannot be easily extracted from them. For this reason, aggregated versions of the two importance metrics are considered in what follows. An example of such aggregations is shown in Fig. 3, which depicts the time evolution per training epoch of a randomly chosen cycle, for each feature's importance on the model's predictions on the validation set's data.

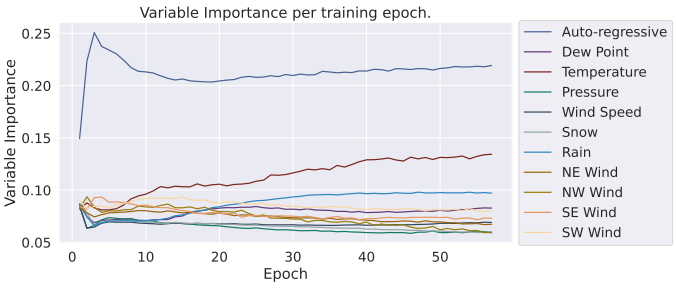

Fig. 3. Variable importance per training epoch of the IMV-Tensor model.

These importance values are obtained for each feature by summing over all the importance values per prediction and dividing by the total number of predictions - a simple mean value extraction. Note that, due to the normalization of the variable importance per feature per prediction, the corresponding mean values are also normalized such that they sum to 1 (100%). It appears that, while each feature's mean importance varies from epoch to epoch, from one point onwards the three most important features are the same: the PM2.5 measurements (auto-regressive), the temperature and the accumulated rainfall, in this order of importance. Additionally, it is worth noting that the PM2.5 measurements are the feature with the highest variance when it comes to its mean importance, which begins from $\sim 15\%$, it reaches a maximum of $\sim 25\%$ and converges at a value of $\sim 21\%$.

Model	RMSE Loss	MAE Loss
IMV-Tensor (present report)	**20.797 ± 0.037**	**11.502 ± 0.016**
IMV-Tensor (original paper)	24.290 ± 0.450	14.870 ± 0.440

Table II. Average prediction scores (measured as RMSE and MAE losses) for the IMV-Tensor model.

During the ten training-evaluation cycles, the IMV-Tensor model requires an average of 61 epochs of training before the early stopping mechanism is activated and the training process is terminated. As far as the model's evaluation on the test set is concerned, the average MAE and RMSE scores (including the corresponding error margins) are recorded in Table II. The scores from the original paper are also provided in the same table. Fig. 4 shows the graph of predicted (blue color) and target (grey color) values for the test set's data, as obtained by the best performing model out of the ten cycles.

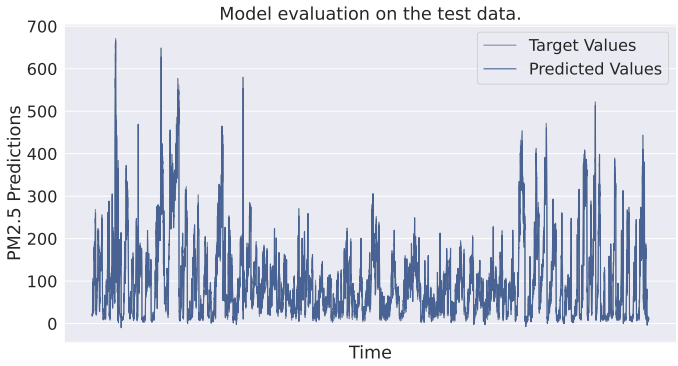

Fig. 4. Target values (grey) and predictions (blue) from the IMV-Tensor model on the test set data.

It becomes evident that the implementation developed for the purposes of the present report significantly outperforms the IMV-Tensor model of the original paper, which was in fact the better among the two IMV-LSTM variants (Tensor and Full). The most prevalent interpretation of this outcome is that the inclusion of the wind's direction as a feature provides the network with additional, nontrivial information. The model's accuracy is astonishing as far as the physics of the studied problem are concerned as well: the network's main task for the studied dataset is the correct prediction of PM2.5 levels in the atmosphere, so that the general population can be alerted on time in cases of elevated pollution levels. Given the fact that the smallest window of PM2.5 values which corresponds to a distinct AQI category (see Section III) has a width of 12 $\mu g/m^3$, a mean absolute error of $\sim$ 11.5 $\mu g/m^3$ on PM2.5 measurements indicates that the model will practically never fail to alert the population about an unexpected negative change of the AQI.

Moving on to the importance analysis, the final feature importance values as calculated based on the model's predictions on the test set's data can be seen in Table III along

Feature	Normalized Importance (%)
PM2.5	21.65 ± 1.02
Dew Point	5.95 ± 0.33
Temperature	12.97 ± 1.42
Pressure	6.37 ± 0.54
Wind Speed	8.26 ± 0.75
Snow	7.90 ± 0.53
Rain	9.25 ± 0.86
NE Wind	6.52 ± 0.24
NW Wind	6.56 ± 0.26
SE Wind	6.94 ± 0.37
SW Wind	7.13 ± 0.39

Table III. Final average variable importance for the IMV-Tensor model.

[3]This is why the "auto-regressive" label is used: past PM2.5 measurements are used in order to determine future values of the same feature.

with their corresponding error margins. Note that two types of aggregations have been performed here: first, the mean values of each feature's importance have been extracted for each evaluation cycle, similarly to what was done in order to obtain the graph of Fig. 4. Then, the average importance of the ten cycles was extracted and finally each feature's importance was normalized in order for the sum of their values to be equal to 1 (100%). It becomes clear that the final importance values follow the trend that was observed during the final epochs of the graph of Fig. 4, i.e. the fact that the PM2.5 measurements, the temperature and the accumulated rainfall, in this order, are the most influential features for the predictions.

As far as the average variable-wise temporal importance values are concerned, they are obtained via similar aggregations and are presented in the heatmap of Fig. 5.

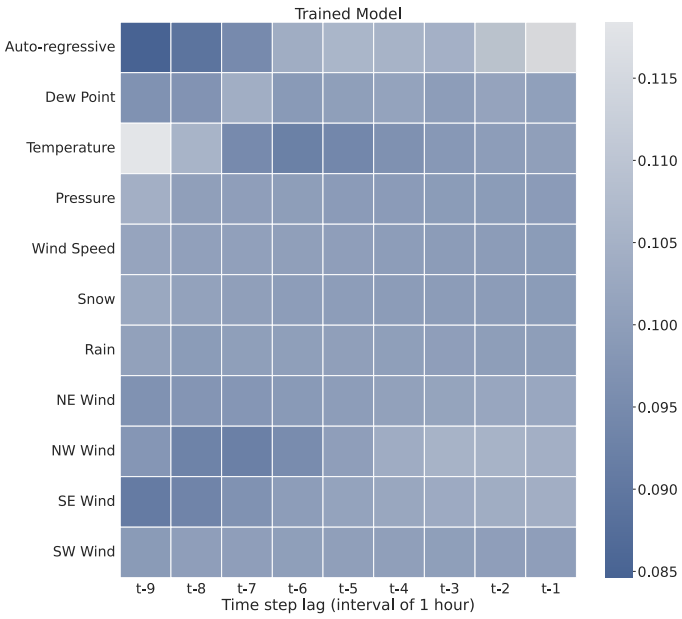

Fig. 5. Variable temportal importance heatmap for the IMV-Tensor model.

Note that emphasis here is not given on inter-feature correlations (i.e. vertical lines) but on intra-feature correlations (i.e. horizontal lines). Each line is normalized to sum to 1 (100%) and each cell's intensity corresponds to the influence of a specific feature's value at a specific time step $t - l$, where l is a time lag of 1 to 9 hours. It appears that PM2.5 measurements (auto-regressive) have a long auto-correlation of approximately 6 hours (in agreement with the original paper's results) and the smaller the time lag, the higher the feature's importance (more recent measurements have a higher impact on the prediction compared to less recent ones). Furthermore, it becomes apparent that the temperature's long term history is more important for the predictions compared to its short term history. Finally, the inverse is true for the wind's direction, as it seems that its short term history tends to affect the prediction more than its long term history.

These observations conclude the first experiment, so another 10-cycle training-evaluation experiment commences, this time for the IMV-Full model, again using a single-prediction time window. This model requires an average of 29 epochs of training before the early stopping mechanism is activated, thus terminating the training process. The corresponding average MAE and RMSE scores (including their errors) are shown in Table IV, alongside the scores reported in the original paper. Fig. 6 depicts the best-performing model's evaluation graph, where the predictions are shown in blue and the target values are shown in gray color.

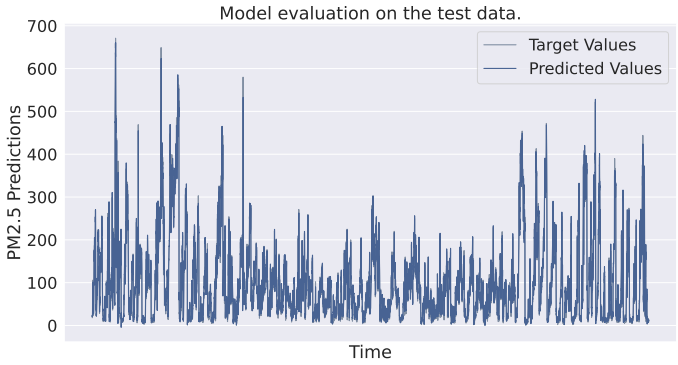

Fig. 6. Actual values (grey) and predictions (blue) from the IMV-Full model on the test set data.

Model	RMSE Loss	MAE Loss
IMV-Full (present report)	**20.613 ± 0.049**	**11.374 ± 0.024**
IMV-Full (original paper)	24.470 ± 0.340	15.230 ± 0.610

Table IV. Average prediction scores (measured as RMSE and MAE losses) for the IMV-Full model.

Unlike what was observed in the original paper, the IMV-Full version of the LSTM model developed for the purposes of the present analysis achieves a better prediction performance on the test set's data, even though the MAE loss is only 1.11% lower compared to the IMV-Tensor version. Nonetheless, this report's models once again outperform the original paper's ones, by a margin of 25.32% for the MAE loss and 15.76% for the RMSE loss.

As far as the importance metrics are concerned, Table V shows the final feature importance values, as calculated

Feature	Normalized Importance (%)
PM2.5	20.82 ± 1.82
Dew Point	5.80 ± 0.27
Temperature	12.08 ± 1.11
Pressure	7.93 ± 0.53
Wind Speed	8.31 ± 0.73
Snow	8.75 ± 0.46
Rain	9.53 ± 0.87
NE Wind	6.48 ± 0.19
NW Wind	6.03 ± 0.22
SE Wind	6.90 ± 0.31
SW Wind	7.37 ± 0.39

Table V. Final variable importance for the IMV-Full model.

based on the model's predictions on the test set's data. While the three most important features are the same as the ones identified for the IMV-Tensor model, the overall scores of all features are somewhat altered. For example, the Pressure's importance has increased by 1.56%, an increase which corresponds to approximately 3 times its error margin in both cases. However, such differences are to be expected and are in full agreement with what was observed in the original paper, since the gate and memory update schemes evolve independently in the IMV-Tensor model, thereby leading to different hidden states compared to the ones extracted from the IMV-Full approach.

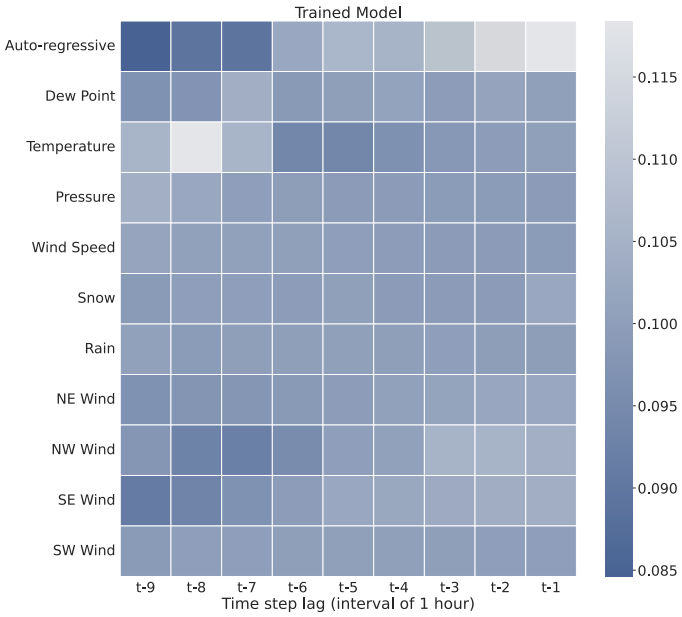

Fig. 7. Variable temportal importance heatmap for the IMV-Full model.

The same observations also hold for the average variable-wise temporal importance values, which are shown in the heatmap of Fig. 7. In general, the results are practically the same with the ones acquired from the IMV-Full mdoel, with the exception of the Temperature's importance, which reaches its peak at a time lag of 8 hours instead of 9.

Since both the IMV-Tensor and IMV-Full versions showed similar results as far as the importance metrics are concerned, the main characteristics that differentiate between the two models is the total time required for their training and, of course, their accuracy. The IMV-Full LSTM is superior to the IMV-Tensor version with respect to both of these aspects, which is why it is the model used for the final experiment of the present analysis.

The final experiment consists of another 10-cycle training-evaluation procedure for the IMV-Full model, however this time the model is trained to predict in a window of 2 hours, instead of 1. After all, this addition was one of the main improvements performed on the original paper's idea. Despite this change, the total training time required does not significantly change, since the model requires an average of 31 epochs of training before the early stopping mechanism is activated. Its score on the test set's data is **25.112** $\pm$ **0.067** for the RMSE loss and **14.923** $\pm$ **0.029** for the MAE loss. Obviously, these results are worse than the ones obtained for the predictions in a 1-hour window, although this is to be expected. Still, the loss values are comparable to the ones reported in the original paper for a 1-hour prediction window, which is a rather remarkable feat. More specifically, the original paper's RMSE loss is lower than the one reported here by 2.62%, however the corresponding MAE loss is higher than the one reported here by 2.02%.

Moving on to the importance metrics, the final feature importance values for the IMV-Full model with double prediction window are shown in Table VI. Again, the PM2.5 measurements, the temperature and the accumulated rainfall are the most impactful features when it comes to the model's predictions. However, the PM2.5 measurements' importance appears somewhat reduced, compared to its scores for the

Feature	Normalized Importance (%)
PM2.5	19.08 $\pm$ 0.54
Dew Point	6.74 $\pm$ 1.03
Temperature	14.57 $\pm$ 1.29
Pressure	7.45 $\pm$ 0.42
Wind Speed	7.11 $\pm$ 0.51
Snow	7.22 $\pm$ 0.52
Rain	8.61 $\pm$ 0.65
NE Wind	7.05 $\pm$ 0.27
NW Wind	7.57 $\pm$ 0.21
SE Wind	7.27 $\pm$ 0.31
SW Wind	7.33 $\pm$ 0.29

Table VI. Final variable importance for the IMV-Full model with double prediction window.

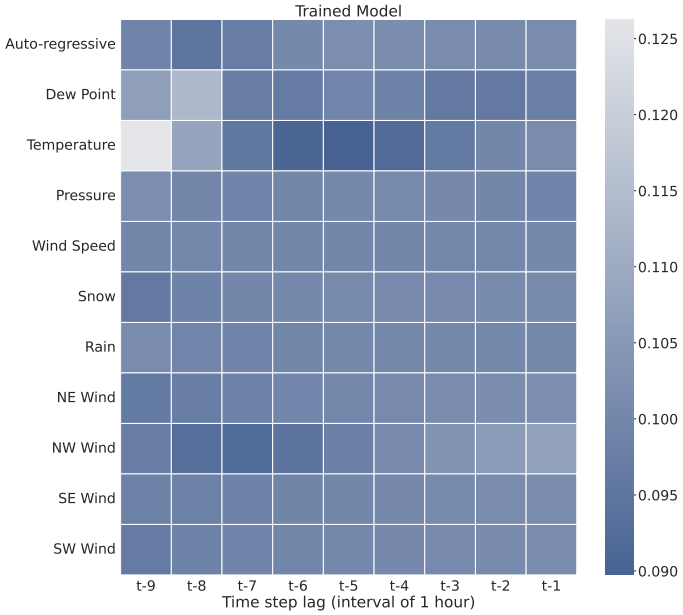

Fig. 8. Variable temportal importance heatmap for the IMV-Full model with double prediction window.

models that predict within a 1-hour window. This indicates that, while the past PM2.5 measurements are significant for the prediction of their next value, more features are needed in order to make predictions in broader time windows. This is also highlighted by the elevated importance of other features, with the temperature being the most prominent one, showing an increase in importance of more than 2%.

Finally, Fig. 8 depicts a heatmap of the average variable-wise temporal importance values. The results are not significantly different compared to the ones obtained for the models that predict in a 1-hour window. The only exception is that of the PM2.5 measurements (auto-regressive), where the long-term correlation that was previously present is no longer evident. This comes in agreement with the observation that as the model's prediction window is broadened, the importance of features other than the PM2.5 measurements increases.

VI. Summary & Outlook

To reiterate, after re-developing both IMV-LSTM versions (Tensor and Full) in PyTorch, the PM2.5 Beijing Dataset was used in order to perform benchmarks on the two models, by training and evaluating each of them 10 times, using a 1-hour prediction window. The IMV-Tensor version required an average of 61 epochs of training and scored an RMSE Loss of 20.797 ± 0.037 and a MAE Loss of 11.502 ± 0.016, surpassing the scores reported from the Keras implementation of the original paper. As far as explainability was concerned, previous PM2.5 measurements proved to be the most important feature with a long autocorrelation, followed by the Temperature (the long term history of which seemed to have a greater impact on the predictions) and the accumulated rainfall. The average training time of the IMV-Full LSTM model was 29 epochs (i.e. less than half compared to the IMV-Tensor version) and its accuracy scores were 20.613 ± 0.049 and 11.374 ± 0.024 for the RMSE and MAE Loss, respectively. The results for the calculated importance metrics were consistent with the ones obtained using the IMV-Tensor model. The IMV-Full model not only surpassed the scores reported in the original paper, but also showed an increased accuracy when compared to the IMV-Tensor implementation. For this reason, it was utilized for a final series of experiments, this time using a 2-hour prediction window. In this case, the achieved scores were 25.112 ± 0.067 for the RMSE Loss and 14.923 ± 0.029 for the MAE Loss, which are comparable to the scores reported in the original paper, but for a 1-hour prediction window. Finally, feature importance-wise, the increase of the prediction window led to a decrease of the PM2.5 measurements' importance and an increase of other features' importance.

Undoubtedly, the IMV-LSTM implementations presented in the original paper and re-developed for the purposes of the present report, showed exceptional accuracy results, as well as consistency with respect to their importance-related results. Even though the versions shown here were already extended versions of the original IMV-LSTM, there is still room for improvement. For starters, further optimizations can be performed for the models trained to predict on broader time windows, possibly by exploring alternative weightings of the hidden state vectors used for the predictions. Furthermore, inter-feature temporal correlations can also be introduced, since the current model only deals with intra-feature temporal importance. Last but not least, even though the methods discussed in the original paper are applied in LSTM structures, they are generalizable and can therefore be implemented in other deep learning models as well. This was in fact one of the most highly stressed points during the reviewing process of the original paper[4].

VII. Contributions

All authors contributed equally in the development and consequent optimization of the PyTorch code for the IMV-LSTM models (Tensor and Full) presented in the original paper, which was the most challenging aspect of the assignment. V. Depastas was the main contributor with regards to the PM2.5 dataset investigation and data preprocessing, which is why they authored most of Sections II and III. C. Nikou was the main contributor as far as the mathematical description of the model was concerned, as well as the visualizations shown in the present report. They authored most of Section IV and also provided detailed documentation to the code. S. Rigas' main contributions were the extension of the model to be able to make predictions in broader time windows and the grid searches for the models' hyperparameters. For this reason, they authored most of Sections I and V. The writing of all remaining sections of the present report was undertaken by all authors equally. Finally, the proposed experiments and the interpretation of their results were also due to all authors equally.

[4]See the relevant discussion here - only for online version.

References

[1] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," Bulletin of Mathematical Biophysics, vol. 5, no. 4, pp. 115–133, 1943, DOI: 10.1007/BF02478259.

[2] D. E. Rumelhart, G. E. Hinton and R. J. Williams, "Learning internal representations by error propagation," Tech. rep. ICS 8504, 1985, Online: Here.

[3] M. I. Jordan, "Serial order: a parallel distributed processing approach," Tech. rep. ICS 8604, 1986, Online: Here.

[4] S. Hochreiter, J. Schmidhuber, "Long Short-Term Memory," Neural Computation, vol. 9, no. 8, pp. 1735-1780, 1997, DOI: 10.1162/neco.1997.9.8.1735.

[5] K. Cho, B. van Merriënboer, D. Bahdanau and Y. Bengio, "On the Properties of Neural Machine Translation: Encoder-Decoder Approaches," Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation, pp. 103-111, 2014, DOI: 10.3115/v1/W14-4012.

[6] M. Du, "Improving LSTM Neural Networks for Better Short-Term Wind Power Predictions," IEEE 2nd International Conference on Renewable Energy and Power Engineering, pp. 105-109, 2019, DOI: 10.1109/REPE48501.2019.9025143.

[7] M. Titos, A. Bueno, L. García, M. C. Benítez and J. Ibañez, "Detection and Classification of Continuous Volcano-Seismic Signals With Recurrent Neural Networks," IEEE Transactions on Geoscience and Remote Sensing, vol. 57, no. 4, pp. 1936-1948, 2019, DOI: 10.1109/TGRS.2018.2870202.

[8] T. Guo, T. Lin and N. Antulov-Fantulin, "Exploring interpretable LSTM neural networks over multi-variable data," Proceedings of the 36th International Conference on Machine Learning, vol. 97, pp. 2494-2504, 2019, Online: Here.

[9] A. Paszke, et al., "PyTorch: An Imperative Style, High-Performance Deep Learning Library," Advances in Neural Information Processing Systems 32, 2019, pp. 8024–8035, Online: Here.

[10] M. Ancona, E. Ceolini, C. Oztireli and M. Gross, "Towards better understanding of gradient-based attribution methods for deep neural networks," 6th International Conference on Learning Representations (ICLR 2018), 2018, Online: Here.

[11] M. T. Ribeiro, S. Singh and C. Guestrin, "Anchors: High-precision model-agnostic explanations," AAAI Conference on Artificial Intelligence, 2018, Online: Here.

[12] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," Advances in Neural Information Processing Systems, pp. 4765–4774, 2017, DOI: 10.5555/3295222.3295230.

[13] A. Shrikumar, P. Greenside and A. Kundaje, "Learning important features through propagating activation differences," Proceedings of the 34th International Conference on Machine Learning, vol. 70, pp. 3145–3153, 2017, DOI: 10.5555/3305890.3306006.

[14] Y. Qin, D. Song, H. Cheng, W. Cheng, G. Jiang, and G. W. Cottrell, "A dual-stage attention-based recurrent neural network for time series prediction," Proceedings of the 26th International Joint Conference on Artificial Intelligence, pp. 2627–2633, 2017, Online: Here.

[15] E. Choi, M. T. Bahadori, J. Sun, J. Kulas, A. Schuetz, and W. Stewart, "Retain: An interpretable predictive model for healthcare using reverse time attention mechanism," Advances in Neural Information Processing Systems, pp. 3504–3512, 2016, DOI: 10.5555/3157382.3157490.

[16] A. Noack, I. Ahern, D. Dou, and B. Li, "Does Interpretability of Neural Networks Imply Adversarial Robustness?," arXiv preprint, arXiv:1912.03430, 2019, Online: Here

[17] F. Fan, J. Xiong, M. Li and G. Wang, "On Interpretability of Artificial Neural Networks: A Survey," IEEE Transactions on Radiation and Plasma Medical Sciences PP(99):1-1, Online: Here

[18] Y. Xu, S. Biswal, S. R. Deshpande, K. O. Maher and J. Sun, "Recurrent attentive and intensive model of multimodal patient monitoring data," Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 2565–2573, ACM, 2018, Online: Here

[19] H. Choi, K. Cho and Y. Bengio, "Fine-grained attention mechanism for neural machine translation," Neurocomputing, 284:171–176, 2018, Online: Here

[20] T. Guo, Z. Xu, X. Yao, H. Chen, K. Aberer and K. Funaya, "Robust online time series prediction with recurrent neural networks," 2016 IEEE DSAA, pp. 816–825. IEEE, 2016, Online: Here

[21] G. Lai, W. -C. Chang, Y. Yang and H. Liu, "Modeling long-and short-term temporal patterns with deep neural networks," arXiv preprint arXiv:1703.07015, 2017, Online: Here

[22] Y. G. Cinar, H. Mirisaee, P. Goswami, E. Gaussier, A. Ait-Bachir and V. Strijov, "Position-based content attention for time series forecasting with sequence-to-sequence rnns," In International Conference on Neural Information Processing, pp. 533–544. Springer, 2017, Online: Here

[23] O. Vinyals, M. Fortunato and N. Jaitly, "Pointer networks," In Advances in Neural Information Processing Systems, pp. 2692–2700, 2015, Online: Here

[24] D. Bahdanau, K. Cho and Y. Bengio, "Neural machine translation by jointly learning to align and translate," In International Conference on Learning Representations, 2014, Online: Here

[25] W. J. Murdoch, P. J. Liu, and B. Yu, "Beyond word importance: Contextual decomposition to extract interactions from lstms," arXiv preprint, arXiv:1801.05453, 2018, Online: Here

[26] W. J. Murdoch and A. Szlam, "Automatic rule extraction from long short term memory networks," International Conference on Learning Representations, 2017, Online: Here

[27] M. T. Ribeiro, S. Singh and C. Guestrin, ""Why Should I Trust You?": Explaining the Predictions of Any Classifier," Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics, 2016, Online: Here

[28] L. Arras, G. Montavon, K.-R. Müller and W. Samek, "Explaining recurrent neural network predictions in sentiment analysis," arXiv preprint, arXiv:1706.07206, 2017, Online: Here

[29] J. Wang, Z. Wang, J. Li and J. Wu, "Multilevel wavelet decomposition network for interpretable time series analysis," Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery, pp. 2437–2446, 2018, Online: Here

[30] L. Chu, X. Hu, J. Hu, L. Wang and J. Pei "Exact and consistent interpretation for piecewise linear neural networks: A closed form solution," Proceedings of the 24th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 1244–1253, 2018, Online: Here

[31] J. N. Foerster, J. Gilmer, J. Sohl-Dickstein, J. Chorowski and D. Sussillo, "Input switched affine networks: An rnn architecture designed for interpretability," International Conference on Machine Learning, pp. 1136–1145, 2017, Online: Here

[32] K. Do, T. Tran and S. Venkatesh, "Matrix-centric neural networks," arXiv preprint, arXiv:1703.01454, 2017, Online: Here

[33] Z. He, S. Gao, L. Xiao, D. Liu, H. He and D. Barber, "Wider and deeper, cheaper and faster: Tensorized lstms for sequence learning," In Advances in Neural Information Processing Systems, pp. 1–11, 2017, Online: Here

[34] O. Kuchaiev and B. Ginsburg, "Factorization tricks for LSTM networks," arXiv preprint, arXiv:1703.10722, 2017, Online: Here

[35] D. Neil, M. Pfeiffer and S.-C. Liu, "Phased lstm: Accelerating recurrent network training for long or event-based sequences," Advances in Neural Information Processing Systems, pp. 3882–3890, 2016, Online: Here

[36] J. Koutnik, K. Greff, F. Gomez and J. Schmidhuber, "A clockwork RNN," International Conference on Machine Learning, pp. 1863–1871, 2014, Online: Here

[37] Pope Iii, C. Arden, et al. "Lung cancer, cardiopulmonary mortality, and long-term exposure to fine particulate air pollution." Jama 287.9 (2002): 1132-1141.

[38] Brook, Robert D., et al. "Particulate matter air pollution and cardiovascular disease: an update to the scientific statement from the American Heart Association." Circulation 121.21 (2010): 2331-2378.

[39] Q. Di, L. Dai, Y. Wang, "Association of Short-term Exposure to Air Pollution With Mortality in Older Adults," The Journal of the American Medical Association 318(24):2446, 2017, Online: Here