Robust training of recurrent neural networks to handle missing data for disease progression modeling

Mostafa Mehdipour Ghazi[1,2,3], **Mads Nielsen**[1,2], **Akshay Pai**[1,2], **M. Jorge Cardoso**[3], **Marc Modat**[3], **Sebastien Ourselin**[3], **Lauge Sørensen**[1,2]
[1]Biomediq A/S, Copenhagen, DK
[2]Department of Computer Science, Univeristy of Copenhagen, DK
[3]Centre for Medical Image Computing, University College Londen, UK
mehdipour@biomediq.com

Abstract

Disease progression modeling (DPM) using longitudinal data is a challenging task in machine learning for healthcare that can provide clinicians with better tools for diagnosis and monitoring of disease. Existing DPM algorithms neglect temporal dependencies among measurements and make parametric assumptions about biomarker trajectories. In addition, they do not model multiple biomarkers jointly and need to align subjects' trajectories. In this paper, recurrent neural networks (RNNs) are utilized to address these issues. However, in many cases, longitudinal cohorts contain incomplete data, which hinders the application of standard RNNs and requires a pre-processing step such as imputation of the missing values. We, therefore, propose a generalized training regime for the most widely used RNN architecture, long short-term memory (LSTM) networks, that can handle missing values in both target and predictor variables in the training set. The proposed LSTM algorithm is applied for modeling of Alzheimer's disease (AD) progression using magnetic resonance imaging (MRI) biomarkers, and it achieves a lower mean absolute error for prediction of values across all considered MRI biomarkers compared to using a standard LSTM network with data imputation or to a prevalent parametric DPM algorithm. In addition, linear discriminant analysis-based classification of AD using predicted biomarker values of the proposed method results in a larger area under the receiver operating characteristic curve (AUC) compared to the same alternatives, and the AUC is comparable to state-of-the-art AUCs from a recent cross-sectional medical image classification challenge. Built-in handling of missing values in LSTM training paves the way for application of RNNs in disease progression modeling.

1 Introduction

Alzheimer's disease (AD) is a chronic neurodegenerative disorder that begins with short-term memory loss and develops over time causing issues in conversation, orientation, and control of bodily functions [1]. Early diagnosis of the disease is challenging and is usually made once cognitive impairment has already compromised daily living. Hence, developing robust data-driven methods for disease progression modeling (DPM) utilizing longitudinal data is necessary to yield a complete perspective of the disease for better diagnosis, monitoring, and prognosis [2].

Existing DPM techniques attempt to describe biomarker measurements as a function of disease progression through continuous curve fitting. In the AD progression literature, a variety of regression-based methods have been applied to fit logistic or polynomial functions to the longitudinal dynamic of each biomarker [3–8]. However, parametric assumptions on the biomarker trajectories limits the

1st Conference on Medical Imaging with Deep Learning (MIDL 2018), Amsterdam, The Netherlands.

applicability of such methods; in addition, none of the existing approaches consider the temporal dependencies among measurements, and most rely on independent biomarker modeling and require alignment of subjects' trajectories – either as a pre-processing step or as part of the algorithm.

Recurrent neural networks (RNNs) are sequence learning methods that can offer continuous, non-parametric, joint modeling of longitudinal data while taking temporal dependencies among measurements into account [9]. However, since longitudinal cohort data often contain missing values due to, for instance, dropped out patients, unsuccessful measurements, or varied trial design, standard RNNs require pre-processing steps for data imputation which often results in suboptimal analyses and predictions [10]. Therefore, the lack of methods to handle missing values in RNNs is evident [11].

Long short-term memory (LSTM) networks are widely used types of RNNs developed to effectively capture long-term temporal dependencies by dealing with the exploding and vanishing gradient problem during backpropagation through time [12–14]. Specifically, they utilize a memory cell with nonlinear (reset) gating units – so called constant error carousel (CECs) – to learn storing history for either long or short time periods. Since its introduction, a variety of LSTM networks have been developed for different time-series applications [15]. The vanilla LSTM, among others, is the most commonly used architecture that employs three reset gates, applies full gate recurrence (passing recurrent inputs at the previous and current time-steps), and backpropagates full gradients. Nevertheless, the complete topology can include biases and peephole (cell-to-gates) connections.

In this paper, we propose a method for training of LSTM networks that can handle missing values in both target and predictor variables. This is achieved via applying the batch gradient descent algorithm together with normalizing the loss function and its gradients with respect to the number of missing target and predictor variables passing the batch-sequence, to ensure a proportional contribution of each weight per epoch. Unlike previous methods that attempt to utilize any bias inherent in patterns of missing values to improve prediction [10, 11], our goal is to make the LSTM network training robust to missing values. The proposed LSTM algorithm is applied for modeling of AD progression in the Alzheimer's Disease Neuroimaging Initiative (ADNI) cohort [16] based on imaging biomarkers, and the estimated biomarker trajectories are used to predict the clinical status of subjects per visits.

Our main contribution is three-fold. Firstly, we formulate a generalization of backpropagation through time for LSTM networks that can handle incomplete data and show that such built-in handling of missing values provides better modeling compared to using data imputation with standard LSTM networks. Secondly, we model temporal dependencies within the ADNI data using the proposed LSTM network via sequence-to-sequence learning. To the best of our knowledge, this is the first time such multi-dimensional sequence learning methods are studied for neurodegenerative disease progression. Lastly, we introduce an end-to-end approach for modeling the longitudinal dynamics of imaging biomarkers – without need for trajectory alignment – and clinical status prediction. This is a practical way to implement a robust DPM for both research and clinical applications.

2 Proposed LSTM algorithm

The main goal of this work is to minimize the influence of missing values on the learned LSTM network parameters. This is achieved by using the batch gradient descend approach together with the backpropagation through time algorithm modified to contribute the effects of missing data in the input and target vectors. More specifically, the algorithm accumulates the subject-specific input weight gradients weighted according to the number of available time points per input node (biomarker) using the subject-biomarker specific normalization factor of $\boldsymbol{\beta}_3^j$. In addition, it uses an L2-norm loss function with residuals weighted according to the number of available time points per output node (biomarker) using the subject-biomarker specific normalization factor of $\boldsymbol{\beta}_2^j$, and normalized with respect to the total number of available input values (visits-biomarkers) – propagated in the forward pass – using the subject-specific normalization factor of β_1^j. Such modification of the loss function also ensures that all gradients of the network weights are indirectly normalized. Finally, the use of batch gradient descend guarantees that there are enough subjects-visits in the batch to be connected at least once to each input node to proportionally contribute in the weight updates.

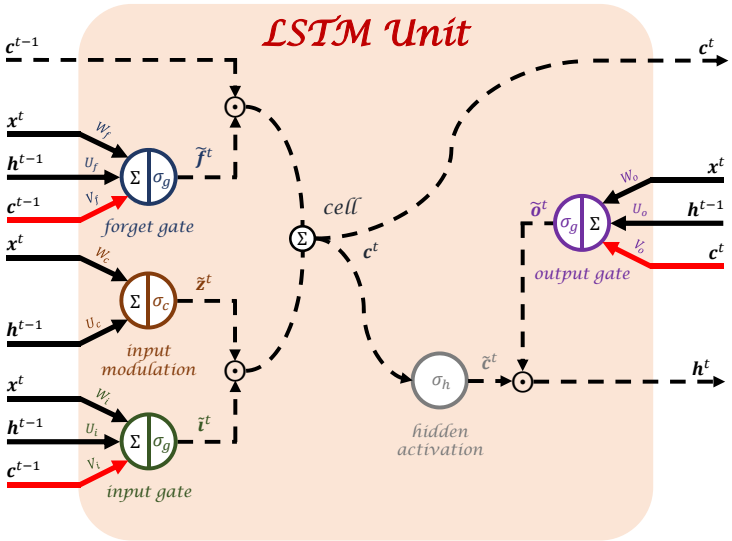

Figure 1: An illustration of vanilla LSTM unit with peephole connections in red color. The solid and dashed lines show weighted and unweighted connections, respectively.

2.1 The basic LSTM architecture

Figure 1 shows a typical schematic of a vanilla LSTM architecture. As can be seen, the topology includes a memory cell, an input modulation, a hidden activation, and three nonlinear reset gates, namely input gate, forget gate, and output gate, each of which accepting recurrent inputs at the previous and current time-steps. The memory cell learns to maintain its state over time while the multiplicative gates learn to open and close access to the constant error/information flow to prevent exploding or vanishing gradients. In fact, the input gate protects the memory contents from perturbation by irrelevant inputs. This is while the output gate protects other units from perturbation by currently irrelevant memory contents. The forget gate, on the other hand, deals with continual or very long input sequences. Finally, peephole connections allow the gates to access the CEC of the same cell state.

2.2 Feedforward in LSTM networks

Assume $\boldsymbol{x}_j^t \in \mathbb{R}^{N\times 1}$ is the j-th observation of an N-dimensional input vector at current time t. If M is the number of output units, feedforward calculations of the LSTM network under study can be summarized as

$$
\begin{aligned}
\boldsymbol{f}_j^t &= W_f\boldsymbol{x}_j^t + U_f\boldsymbol{h}_j^{t-1} + \boldsymbol{V}_f \odot \boldsymbol{c}_j^{t-1} + \boldsymbol{b}_f \longrightarrow \tilde{\boldsymbol{f}}_j^t = \sigma_g(\boldsymbol{f}_j^t)\\
\boldsymbol{i}_j^t &= W_i\boldsymbol{x}_j^t + U_i\boldsymbol{h}_j^{t-1} + \boldsymbol{V}_i \odot \boldsymbol{c}_j^{t-1} + \boldsymbol{b}_i \longrightarrow \tilde{\boldsymbol{i}}_j^t = \sigma_g(\boldsymbol{i}_j^t)\\
\boldsymbol{z}_j^t &= W_c\boldsymbol{x}_j^t + U_c\boldsymbol{h}_j^{t-1} + \boldsymbol{b}_c \longrightarrow \tilde{\boldsymbol{z}}_j^t = \sigma_c(\boldsymbol{z}_j^t)\\
\boldsymbol{c}_j^t &= \tilde{\boldsymbol{f}}_j^t \odot \boldsymbol{c}_j^{t-1} + \tilde{\boldsymbol{i}}_j^t \odot \tilde{\boldsymbol{z}}_j^t \longrightarrow \tilde{\boldsymbol{c}}_j^t = \sigma_h(\boldsymbol{c}_j^t)\\
\boldsymbol{o}_j^t &= W_o\boldsymbol{x}_j^t + U_o\boldsymbol{h}_j^{t-1} + \boldsymbol{V_o} \odot \boldsymbol{c}_j^t + \boldsymbol{b}_o \longrightarrow \tilde{\boldsymbol{o}}_j^t = \sigma_g(\boldsymbol{o}_j^t)\\
\boldsymbol{h}_j^t &= \tilde{\boldsymbol{o}}_j^t \odot \tilde{\boldsymbol{c}}_j^t
\end{aligned}
$$

where $\{\boldsymbol{f}_j^t, \boldsymbol{i}_j^t, \boldsymbol{z}_j^t, \boldsymbol{c}_j^t, \boldsymbol{o}_j^t, \boldsymbol{h}_j^t\} \in \mathbb{R}^{M\times 1}$ and $\{\tilde{\boldsymbol{f}}_j^t, \tilde{\boldsymbol{i}}_j^t, \tilde{\boldsymbol{z}}_j^t, \tilde{\boldsymbol{c}}_j^t, \tilde{\boldsymbol{o}}_j^t, \tilde{\boldsymbol{h}}_j^t\} \in \mathbb{R}^{M\times 1}$ are j-th observation of forget gate, input gate, modulation gate, candidate or cell state, output gate, and hidden output at time t before and after activation, respectively. Moreover, $\{W_f, W_i, W_o, W_c\} \in \mathbb{R}^{M\times N}$ and $\{U_f, U_i, U_o, U_c\} \in \mathbb{R}^{M\times M}$ are respectively sets of connecting weights from input and recurrent to the gates and cell; $\{\boldsymbol{V}_f, \boldsymbol{V}_i, \boldsymbol{V}_o\} \in \mathbb{R}^{M\times 1}$ is the set of peephole connections, $\{\boldsymbol{b}_f, \boldsymbol{b}_i, \boldsymbol{b}_o, \boldsymbol{b}_c\} \in$

$\mathbb{R}^{M\times 1}$ represents corresponding biases of neurons, and the product $\odot$ denotes the element-wise multiplication. Finally, σ_g, σ_c, and σ_h imply nonlinear activation functions assigned for the gates, input modulation, and hidden output, respectively. Logistic sigmoid is always applied for the gates with range $[0, 1]$ while hyperbolic tangent is typically used for both cell input and cell output with range $[-1, 1]$.

2.3 Robust backpropagation through time

Let $\boldsymbol{\mathcal{L}} \in \mathbb{R}^{M\times 1}$ be the loss function defined based on the actual target $\boldsymbol{s}$ and network output $\boldsymbol{y}$. Here, we simply assume that the network output is the same hidden output. The idea is to calculate the partial derivatives of the loss function (δ) with respect to the weights using the chain rule. Hence, the backpropagation calculations through time using full gradients can be obtained as

$$\boldsymbol{\mathcal{L}} = \frac{1}{2}\sum_{j,t}\frac{1}{\beta_1^j\boldsymbol{\beta}_2^j}\odot(\boldsymbol{y}_j^t-\boldsymbol{s}_j^t)^2 \longrightarrow \delta\boldsymbol{y}_j^t = \frac{\partial\boldsymbol{\mathcal{L}}}{\partial\boldsymbol{y}_j^t} = \frac{1}{\beta_1\boldsymbol{\beta}_2}\odot(\boldsymbol{y}_j^t-\boldsymbol{s}_j^t)$$
$$\delta\boldsymbol{h}_j^t = \delta\boldsymbol{y}_j^t + U_f^T\delta\boldsymbol{f}_j^{t+1} + U_i^T\delta\boldsymbol{i}_j^{t+1} + U_c^T\delta\boldsymbol{z}_j^{t+1} + U_o^T\delta\boldsymbol{o}_j^{t+1}$$
$$\delta\tilde{\boldsymbol{o}}_j^t = \delta\boldsymbol{h}_j^t\odot\tilde{\boldsymbol{c}}_j^t \longrightarrow \delta\boldsymbol{o}_j^t = \delta\tilde{\boldsymbol{o}}_j^t\odot\sigma_g'(\boldsymbol{o}_j^t)$$
$$\delta\tilde{\boldsymbol{c}}_j^t = \delta\boldsymbol{h}_j^t\odot\tilde{\boldsymbol{o}}_j^t \longrightarrow \delta\boldsymbol{c}_j^t = \delta\tilde{\boldsymbol{c}}_j^t\odot\sigma_h'(\boldsymbol{c}_j^t) + \delta\boldsymbol{c}_j^{t+1}\odot\tilde{\boldsymbol{f}}_j^{t+1} + \boldsymbol{V}_f\odot\delta\boldsymbol{f}_j^{t+1} + \boldsymbol{V}_i\odot\delta\boldsymbol{i}_j^{t+1} + \boldsymbol{V}_o\odot\delta\boldsymbol{o}_j^t$$
$$\delta\tilde{\boldsymbol{z}}_j^t = \delta\boldsymbol{c}_j^t\odot\tilde{\boldsymbol{i}}_j^t \longrightarrow \delta\boldsymbol{z}_j^t = \delta\tilde{\boldsymbol{z}}_j^t\odot\sigma_c'(\boldsymbol{z}_j^t)$$
$$\delta\tilde{\boldsymbol{i}}_j^t = \delta\boldsymbol{c}_j^t\odot\tilde{\boldsymbol{z}}_j^t \longrightarrow \delta\boldsymbol{i}_j^t = \delta\tilde{\boldsymbol{i}}_j^t\odot\sigma_g'(\boldsymbol{i}_j^t)$$
$$\delta\tilde{\boldsymbol{f}}_j^t = \delta\boldsymbol{c}_j^t\odot\boldsymbol{c}_j^{t-1} \longrightarrow \delta\boldsymbol{f}_j^t = \delta\tilde{\boldsymbol{f}}_j^t\odot\sigma_g'(\boldsymbol{f}_j^t)$$
$$\delta\boldsymbol{x}_j^t = W_f^T\delta\boldsymbol{f}_j^t + W_i^T\delta\boldsymbol{i}_j^t + W_c^T\delta\boldsymbol{z}_j^t + W_o^T\delta\boldsymbol{o}_j^t$$

where $\beta_1^j = J\frac{||\boldsymbol{x}_j||}{TN}$ and $\boldsymbol{\beta}_2^j = |\boldsymbol{y}_j| \in \mathbb{Z}^{M\times 1}$ are normalization factors to handle missing values with batch size J and sequence length T. Also, $||\boldsymbol{x}_j||$ and $|\boldsymbol{y}_j|$ denote the total number of available input values and the number of available target time points per biomarker, respectively. Finally, if $\theta \in \{f, i, z, o\}$ and $\phi \in \{f, i\}$, the gradients of the loss function with respect to the weights are calculated as

$$\delta W_\theta = \sum_{j=1}^{J}\delta\boldsymbol{\theta}_j^{\{0\to T\}}\boldsymbol{x}_j^{\{0\to T\}}\odot\frac{1}{\boldsymbol{\beta}_3^j}$$
$$\delta U_\theta = \sum_{j=1}^{J}\delta\boldsymbol{\theta}_j^{\{1\to T\}}\boldsymbol{h}_j^{\{0\to T-1\}}$$
$$\delta\boldsymbol{V}_\phi = \sum_{j=1}^{J}\sum_{t=0}^{T-1}\delta\boldsymbol{\phi}_j^{t+1}\odot\boldsymbol{c}_j^t$$
$$\delta\boldsymbol{V}_o = \sum_{j=1}^{J}\sum_{t=0}^{T}\delta\boldsymbol{o}_j^t\odot\boldsymbol{c}_j^t$$
$$\delta\boldsymbol{b}_\theta = \sum_{j=1}^{J}\sum_{t=0}^{T}\delta\boldsymbol{\theta}_j^t$$

where $\boldsymbol{\beta}_3^j = \frac{|\boldsymbol{x}_j|}{T} \in \mathbb{Z}^{1\times N}$ is the input normalization factor handling missing values and $|\boldsymbol{x}_j|$ is the number of available input time points per biomarker.

Table 1: Demographics statistics of the TADPOLE dataset

	Visits (male / female)	Age, year (mean±SD male / female)	Education, year (mean±SD all)
CN	1,356 / 1,389	76.67±6.44 / 75.85±6.28	16.38±2.70
MCI	2,454 / 1,604	75.59±7.47 / 73.87±8.09	15.91±2.84
AD	1,208 / 900	77.22±7.11 / 75.45±7.92	15.18±2.99
All (incl. unlabeled)	12,741	76.00±7.38	15.91±2.86

2.4 Momentum batch gradient descent

As an iterative algorithm, momentum batch gradient descent method is applied to find the local minimum of the loss function calculated over a batch while speeding up the convergence. The update rule can be written as

$$\vartheta^{new} = \mu\vartheta^{old} - \alpha(\delta\omega + \gamma\omega^{old})$$
$$\omega^{new} = \omega^{old} + \vartheta^{new}$$

where ϑ is the weight update initialized at zero, ω is a typical to be updated weight array, $\delta\omega$ is the gradient of loss function with respect to the weight ω, and α, γ, and μ are the learning rate, weight decay or regularization factor, and momentum weight, respectively.

3 Experiments

3.1 Data preparation

We utilize the dataset of The Alzheimer's Disease Prediction Of Longitudinal Evolution [1] (TADPOLE) challenge for time-series analysis using the LSTM network. The dataset is composed of data from three ADNI phases of ADNI 1, ADNI GO, and ADNI 2. This includes biomarkers acquired from 1,737 subjects (957 males and 780 females) during 12,741 visits at 22 distinct time points between 2003 and 2017. The data is captured by measures of brain structural integrity using magnetic resonance imaging (MRI), positron emission tomography (PET), and diffusion tensor imaging (DTI), or obtained from blood tests of cerebrospinal fluid (CSF), neuropsychological (cognitive) tests, as well as demographics and genetics information. Table 1 summarizes statistics of the demographics in the TADPOLE dataset. Note that the subjects include missing measurements during their visits and not all of them are clinically labeled.

In this paper, we have merged existing groups labeled as cognitively normal (CN), significant memory concern (SMC), and normal (NL) under cognitively normal, mild cognitive impairment (MCI), early MCI (EMCI), and late MCI (LMCI) under mild cognitive impairment, and Alzheimer's disease (AD) and Dementia under Alzheimer's disease. Moreover, groups with labels converting from one status to another, e.g. "MCI-to-AD", are assumed to belong the next status ("AD" in this example).

MRI biomarkers are used for AD progression modeling. This includes T1–weighted brain MRI volumes of ventricles, hippocampus, whole brain, entorhinal cortex, fusiform, and middle temporal gyrus. We normalize MRI volumes with respect to the corresponding intracranial volume (ICV). Out of 22 visits, we select 11 visits – including baselines – with a fix interval of one year to span the majority of measurements and subjects. Next, we filter the data outliers based on the potential range of each biomarker and normalize the measurements to be in the range $[-1, 1]$. Finally, subjects with less than three distinct visits per biomarker are removed to better adapt with the LSTM network.

For the evaluation purpose, we partition the entire dataset to three non-overlapping subsets of training, validation, and test. To achieve this, we randomly select 10% of the within-class subjects for validation and the same for testing. To be more specific, based on the baseline labels of subjects, we randomly choose within-class samples ensuring to have enough subjects with few and large number of visits in each subset.

[1] https://tadpole.grand-challenge.org

3.2 Evaluation metrics

Mean absolute error (MAE) and multi-class area under the receiver operating characteristic (ROC) curve (AUC) are used to assess the modeling and classification performances, respectively. MAE measures accuracy of continuous prediction per biomarker by computing the difference between actual and estimated variables as follows

$$\mathrm{MAE} = \frac{1}{\mathcal{I}} \sum_{j,t} |\boldsymbol{y}_j^t - \boldsymbol{s}_j^t|$$

where $\boldsymbol{s}_j^t$ and $\boldsymbol{y}_j^t$ are the ground-truth and estimated values of the specific biomarker for the j-th subject at the t-th visiting point, respectively, and $\mathcal{I}$ is the number of existing points in the target array $\boldsymbol{s}$.

Multi-class AUC [17], on the other hand, is a measure to examine the diagnostic performance of a multi-class test using ROC analysis. AUC can be calculated using the posterior probabilities as follows

$$\mathrm{AUC} = \frac{1}{(n_c(n_c-1))} \sum_{i=1}^{n_c-1} \sum_{k=i+1}^{n_c} \frac{1}{n_i n_k} \Big[\mathrm{SR}_i - \frac{n_i(n_i+1)}{2} + \mathrm{SR}_k - \frac{n_k(n_k+1)}{2} \Big]$$

where n_c stands for the number of distinct classes, n_i denotes the number of available points belonging to the i-th class, and SR_i is the sum of the ranks of posteriors $p(c_i|\boldsymbol{s}_i)$ after sorting all concatenated posteriors $\{p(c_i|\boldsymbol{s}_i), p(c_i|\boldsymbol{s}_k)\}$ in an increasing order, where $\boldsymbol{s}_i$ and $\boldsymbol{s}_k$ are vectors of scores belonging to the true classes c_i and c_k, respectively.

3.3 Experimental setup

All the proposed and utilized methods in this work are developed using the in-house implementation in MATLAB R2017b. We initialize the LSTM network weights by generating uniformly distributed random values in the range $[-0.05, 0.05]$ and set the weights' updates and weights' gradients to zero. We set the batch size to the number of available training subjects. Furthermore, for simplicity, we use the first ten visits to estimate the second to eleventh visits per subject and use the estimated values for evaluation. Finally, we train the network using feedforward and the proposed method of backpropagation through time where the network replace the input missing values and corresponding error of the output missing values with zero.

We utilize the validation set to adjust the network parameters each time by adjusting one of the parameters while keeping the rest at fixed values to achieve the lowest average MAE in the validation set. Using this strategy, the optimal network parameters are adjusted as $\alpha = 0.1$, $\mu = 0.9$, and $\gamma = 0.0001$ with 1,000 epochs. Also, the corresponding MAEs for the validation set are obtained as $2.9590 \times 10^{-3}, 2.4603 \times 10^{-4}, 1.4943 \times 10^{-2}, 2.4161 \times 10^{-4}, 7.5522 \times 10^{-4}, 9.6592 \times 10^{-4}$, for ventricles, hippocampus, whole brain, entorhinal cortex, fusiform, and middle temporal gyrus, respectively. It is worthwhile mentioning that all the normalized estimated measurements are transformed back to their actual ranges to calculate MAEs. We also experience that using LSTM network with peephole connections improves the performance.

3.4 Results

After training the network with the best achieved models, we test our network using the obtained test subset. Next, we train the network using mean imputation (LSTM-Mean) [11] and forward imputation (LSTM-Forward) [10]. Moreover, we apply the regression method [3] to computationally model the AD progression. Table 2 compares the test modeling performance (MAE) of MRI biomarkers using aforementioned approaches.

As it can be deduced from Table 2, our proposed method outperforms all other modeling techniques in all categories. It should be noticed that when we apply data imputation, the backpropagation formulas simply generalize to the standard LSTM network.

Table 2: Test modeling performance (MAE) for MRI biomarkers using different DPM methods.

	Proposed	LSTM-Mean	LSTM-Forward	Jedynak et al. [3]
Ventricles	3.0674×10^{-3}	6.2010×10^{-3}	4.7204×10^{-3}	8.0718×10^{-3}
Hippocampus	2.3267×10^{-4}	5.0916×10^{-4}	3.3977×10^{-4}	5.1455×10^{-4}
Whole brain	1.3298×10^{-2}	2.3746×10^{-2}	1.6389×10^{-2}	5.5125×10^{-3}
Entorhinal cortex	2.1138×10^{-4}	3.0324×10^{-4}	2.5489×10^{-4}	3.4660×10^{-4}
Fusiform	6.7932×10^{-4}	1.2964×10^{-3}	1.0044×10^{-3}	9.0342×10^{-4}
Middle temporal gyrus	8.6750×10^{-4}	1.2606×10^{-3}	1.1759×10^{-3}	1.1092×10^{-3}

Table 3: Test prediction performance (AUC) for MRI biomarkers using different DPM methods.

	Proposed	LSTM-Mean	LSTM-Forward	Jedynak et al. [3]
CN vs. MCI	0.5914	0.5838	0.5800	0.5468
CN vs. AD	0.9029	0.8404	0.8150	0.7826
MCI vs. AD	0.7844	0.6936	0.6890	0.7330
CN vs. MCI vs. AD	0.7596	0.7059	0.6947	0.6875

To evaluate the ability of estimated measurements in predicting the clinical labels, we apply linear discriminant analysis (LDA) to the multi-dimensional training biomarkers' estimations corresponding to the available labels to compute the posterior probability scores in the test data. The obtained scores are then used to calculate the AUCs. The clinical status prediction results are summarized in Table 3 for different methods.

As can be seen, here also the proposed method generally outperforms all other schemes. This, in turn, reveals the effect of modeling on the classification performance. One could of course use different classifiers to improve the results. But, our focus in this paper is on disease progression modeling or sequence-to-sequence learning. On the other hand, it is possible to train the LSTM network for classification (sequence-to-label) problem. However, in practice, even much less data is labeled to be used for training the network. Therefore, training the LSTM network with the aim of classification not only would sacrifice the modeling, but also could result in a pour prediction performance because of the lack of sufficient data.

4 Discussion

The diagnostic LDA classification results using predicted MRI imaging biomarker values from the DPM models were comparable to state-of-the-art cross-sectional MRI-based classification results in the recent Computer-Aided Diagnosis of Dementia (CADDementia) challenge [18]. LDA classification using predicted features from the method proposed in this paper achieved a multi-class AUC of 0.76 which was within the top-five multi-class AUCs in the challenge that ranged from 0.79 to 0.75. It should, however, be noted that there are important differences between this study and the CADDementia challenge. This study had the advantage of training and test set from the same cohort whereas CADDementia algorithms were applied to classify data from independent cohorts. On the other hand, the top performing CADDementia algorithms incorporated other types of MRI features in addition to volumetry. Another important difference is that in CADDementia, test features were available whereas in this study, features were predicted for the test subject based on longitudinal historical data and then fed to the classifier.

The most common approach to missing data handling in LSTMs is a data imputation pre-processing step prior to application of a vanilla LSTM, usually using mean or forward imputation. This two-step procedure decouples missing data handling and network training and is heavily influenced by the choice of data imputation scheme, resulting in sub-optimal performance. Other approaches update the architecture to utilize possible correlations between missing value patterns and the target to improve prediction [10, 11]. Our goal was different; we wanted to make the training of the LSTM network robust to missing values, to more faithfully capture the true underlying signal and make the learned model generalizable across cohorts and not rely on specific cohort or demographic-specific circumstances correlated with the target.

In general, standard LSTM networks are designed to handle sequences with a fixed temporal or spatial sampling rate within longitudinal data. We used the same approach in the AD progression modeling application by disregarding, for example, months 3, 6 and 18 follow-up visits, and confined the experiments to yearly follow-up in the ADNI data. However, one could utilize modified LSTM architectures such as time-aware LSTM [19] to address irregular time steps in longitudinal patient records.

In summary, a training algorithm for LSTM networks was proposed aiming to improve robustness to missing data, and applied to AD progression modeling using longitudinal measurements of imaging biomarkers with missing values. To the best of our knowledge this is the first time RNNs have been applied for this purpose. The proposed training method demonstrated better performance than using imputation prior to standard LSTM network, both in terms of joint modeling and prediction of the biomarkers, and in terms of subsequent diagnostic classification using the biomarker predictions. In addition, it outperformed an established parametric, regression-based DPM method, and provided classification performance comparable with state-of-the-art in the context of MRI imaging biomarker classification of AD. This study highlights the potential of RNNs for modeling of disease progression using longitudinal measurements of biomarkers, provided that proper care is taken to handle problems with this type of data, including missing values.

Acknowledgments

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 721820. This work uses the TADPOLE data sets https://tadpole.grand-challenge.org constructed by the EuroPOND consortium http://europond.eu funded by the European Union's Horizon 2020 research and innovation programme under grant agreement No 666992.

References

[1] Guy McKhann, David Drachman, Marshall Folstein, Robert Katzman, Donald Price, and Emanuel M. Stadlan. Clinical diagnosis of Alzheimer's disease. *Neurology*, 34(7):939–939, 1984.

[2] Neil P. Oxtoby and Daniel C. Alexander. Imaging plus X: multimodal models of neurodegenerative disease. *Current opinion in neurology*, 30(4):371, 2017.

[3] Bruno M. Jedynak, Andrew Lang, Bo Liu, Elyse Katz, Yanwei Zhang, Bradley T. Wyman, David Raunig, C. Pierre Jedynak, Brian Caffo, and Jerry L Prince. A computational neurodegenerative disease progression score: method and results with the Alzheimer's disease neuroimaging initiative cohort. *Neuroimage*, 63(3):1478–1486, 2012.

[4] Anders M. Fjell, Lars T. Westlye, Håkon Grydeland, Inge Amlien, Thomas Espeseth, Ivar Reinvang, Naftali Raz, Dominic Holland, Anders M. Dale, and Kristine B. Walhovd. Critical ages in the life course of the adult brain: nonlinear subcortical aging. *Neurobiology of aging*, 34 (10):2239–2247, 2013.

[5] Neil P. Oxtoby, Alexandra L. Young, Nick C. Fox, Pankaj Daga, David M. Cash, Sebastien Ourselin, Jonathan M. Schott, and Daniel C. Alexander. Learning imaging biomarker trajectories from noisy Alzheimer's disease data using a bayesian multilevel model. In *Bayesian and grAphical Models for Biomedical Imaging*, pages 85–94. 2014.

[6] Michael C. Donohue, Helene Jacqmin-Gadda, Mélanie Le Goff, Ronald G. Thomas, Rema Raman, Anthony C. Gamst, Laurel A. Beckett, Clifford R. Jack, Michael W. Weiner, Jean-Francois Dartigues, and Paul S. Aisen. Estimating long-term multivariate progression from short-term data. *Alzheimer's & dementia: the journal of the Alzheimer's Association*, 10(5): S400–S410, 2014.

[7] Wai-Ying Wendy Yau, Dana L. Tudorascu, Eric M. McDade, Snezana Ikonomovic, Jeffrey A. James, Davneet Minhas, Wenzhu Mowrey, Lei K. Sheu, Beth E. Snitz, Lisa Weissfeld, et al. Longitudinal assessment of neuroimaging and clinical markers in autosomal dominant Alzheimer's disease: a prospective cohort study. *The Lancet Neurology*, 14(8):804–813, 2015.

[8] Ricardo Guerrero, Alexander Schmidt-Richberg, Christian Ledig, Tong Tong, Robin Wolz, and Daniel Rueckert. Instantiated mixed effects modeling of Alzheimer's disease markers. *NeuroImage*, 142:113–125, 2016.

[9] Barak A. Pearlmutter. Learning state space trajectories in recurrent neural networks. *Neural Computation*, 1(2):263–269, 1989.

[10] Zachary C. Lipton, David C. Kale, and Randall Wetzel. Modeling missing data in clinical time series with RNNs. *Machine Learning for Healthcare*, 2016.

[11] Zhengping Che, Sanjay Purushotham, Kyunghyun Cho, David Sontag, and Yan Liu. Recurrent neural networks for multivariate time series with missing values. *arXiv:1606.01865*, 2016.

[12] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

[13] Felix A. Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with LSTM. 1999.

[14] Felix A. Gers and Jürgen Schmidhuber. LSTM recurrent networks learn simple context-free and context-sensitive languages. *IEEE Transactions on Neural Networks*, 12(6):1333–1340, 2001.

[15] Klaus Greff, Rupesh K. Srivastava, Jan Koutník, Bas R. Steunebrink, and Jürgen Schmidhuber. LSTM: A search space odyssey. *IEEE transactions on neural networks and learning systems*, 28(10):2222–2232, 2017.

[16] Ronald Carl Petersen, P.S. Aisen, L.A. Beckett, M.C. Donohue, A.C. Gamst, D.J. Harvey, C.R. Jack, W.J. Jagust, L.M. Shaw, A.W. Toga, J.Q. Trojanowski, and M.W. Weiner. Alzheimer's disease neuroimaging initiative (ADNI) clinical characterization. *Neurology*, 74(3):201–209, 2010.

[17] David J. Hand and Robert J. Till. A simple generalisation of the area under the ROC curve for multiple class classification problems. *Machine learning*, 45(2):171–186, 2001.

[18] Esther E. Bron, Marion Smits, Wiesje M. Van Der Flier, Hugo Vrenken, Frederik Barkhof, Philip Scheltens, Janne M. Papma, Rebecca M.E. Steketee, Carolina Méndez Orellana, Rozanna Meijboom, et al. Standardized evaluation of algorithms for computer-aided diagnosis of dementia based on structural MRI: the CADDementia challenge. *NeuroImage*, 111:562–579, 2015.

[19] Inci M. Baytas, Cao Xiao, Xi Zhang, Fei Wang, Anil K. Jain, and Jiayu Zhou. Patient subtyping via time-aware LSTM networks. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 65–74, 2017.