Modelling Progression of Alzheimer's Disease with RNN

Pedro Miguel Pinto dos Santos
pedro.m.p.santos@tecnico.ulisboa.pt

Instituto Superior Técnico, Lisboa, Portugal

July 2021

Abstract

Alzheimer's Disease is a form of dementia which will become more present as life expectancy increases. Modelling the progression of this pathology would be beneficial for the medical community. The goal of this thesis is to correctly predict the diagnoses of patients, namely if they developed the disease or any stage that might lead to it.

Longitudinal datasets such as ADNI provided the cognitive tests, diagnostics, patient age and MRI data which were used as features for this work. RNN based models are indicated for this task as they are best suited for longitudinal data. To achieve this goal, in addition to a new RNN architecture termed Time-Aware Long Short Term Memory Network (T-LSTM), two other strategies were tested: time intervals (TI) between consultations, and generative models for the imputation of missing data. Varying TI are always present in some datasets due to acquisition protocols. Generative models achieve imputations based on the statistical distribution data.

The generative models achieved better results than the baseline imputation methods and part of the state of the art, despite not using as many features as desired. The T-LSTM allows for more data to be used than the LSTM network, creating bigger datasets which are advantageous for this task. However, its classification results are not better.

Concluding, the generative methods are competitive with the state of the art and might yield the best results in this task. The T-LSTM allows for more data to be used but its decay mechanism might be hindering its results.

Keywords: Alzheimer's Disease, RNN, T-LSTM, GAN, VAE.

1. Introduction

AD is a neurodegenerative disease which slowly and progressively destroys brain cells and represents 60-65% of cases of dementia. This disease affects cognitive function, presenting itself mainly with the loss of memory and, for example, through confusion and slower thinking. It happens through abnormal deposits of proteins forming amyloid plaques and tau tangles throughout the brain. With no currently known cure and as life expectancy increases, the number of people affected with this pathology and other forms of dementia will most probably increase [1].

According to the EFPIA [2], Europe has 10.5 million patients with some form of dementia, of which up to 80% suffer from AD. In the USA, AD is the sixth leading cause of death, with recent estimates placing it in third place [3]. Modelling the progression of this disease would be of great help for diagnosis, treatment purposes, caretakers and the national health services. The typical first symptom of AD is more memory loss than what would be normal for the patient's age but to an extent that does not interfere with normal day-to-day life, which is designated as Mild Cognitive Impairment (MCI). While not all patients who present MCI progress to AD, this stage precedes AD.

Machine Learning methods such as Recurrent Neural Networks (RNN), software like Python and ML libraries are increasingly popular to tackle problems such as the one to model the progression of AD. The RNN come at the cost of requiring large datasets and complete longitudinal data with equal time intervals between samples [4].

In this work, the models to be applied took into account the handling of missing data with simple state of the art methods compared with generative model based methods, namely a model called Generative Adversarial Imputation Network (GAIN) and another one called Variational Autoencoder (VAE). It was also analysed how to exploit time dependencies between observations by using an appropriate RNN architecture which was compared to the state of the art LSTM, all with the goal of im-

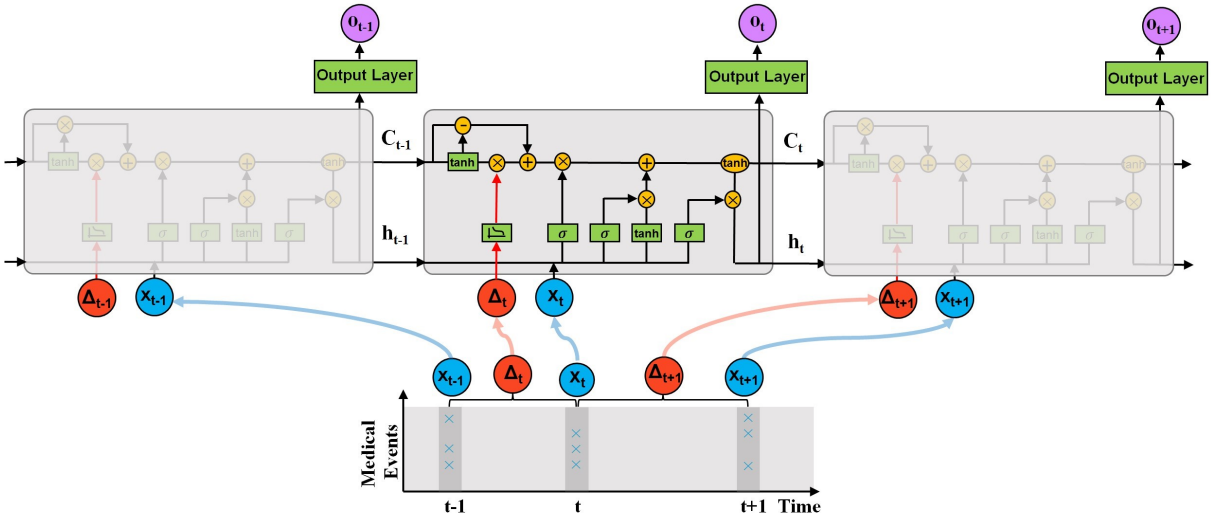

Figure 1: T-LSTM architecture from I. M. Baytas et al. [5]

proving the current models for modelling progression of AD with RNN.

2. Proposed Methods

2.1. Proposed RNN Architecture

The Time-Aware Long Short Term Memory (T-LSTM) [5], was designed to handle varying Time Intervals (TI) between sequences. In this architecture, which is depicted in fig. 1, the TI between measurements are used as an input to create a Discounted Short-Term Memory which has the goal of creating an Adjusted Previous Memory.

To define this Adjusted Previous Memory (C^*_{t-1}), there are several steps, all defined by I. M. Baytas et al. [5]. Firstly, the definition of the Short-Term Memory is as follows

$$C^S_{t-1} = tanh(W_d \cdot C_{t-1} + b_d). \tag{1}$$

Followed by the Discounted Short-Term Memory

$$\hat{C}^S_{t-1} = C^S_{t-1} \cdot g(\Delta_t), \tag{2}$$

which results of the application of the nonlinear function g, that has the goal to introduce a penalty which is larger to large TI. An example of such, and the implemented function in the T-LSTM architecture, is the following

$$g(\Delta_t) = \frac{1}{ln(e + \Delta_t)}. \tag{3}$$

The Long-Term Memory is defined as the difference between the Current Memory and the Short-Term Memory

$$C^T_{t-1} = C_{t-1} - C^S_{t-1}. \tag{4}$$

Finally, the Adjusted Previous Memory is given by

$$C^*_{t-1} = C^T_{t-1} + \hat{C}_{t-1}. \tag{5}$$

This last expression can be rewritten taking into consideration the definitions of Long-Term Memory (4) and of Discounted Short-Term Memory (2)

$$\begin{aligned} C^*_{t-1} &= C_{t-1} - C^s_{t-1} + C^S_{t-1} \cdot g(\Delta_t) = \\ &= C_{t-1} - C^S_{t-1}(1 - g(\Delta_t). \end{aligned} \tag{6}$$

With this formulation, it is highlighted that the contribution of the Short-Term Memory depends of the value of $g(\Delta_t)$. Moreover, if this function g takes value 1 (which would happen with $\Delta_t = 0$ in the formulation above), the Short-Term Memory does not contribute to the Adjusted Previous Memory and thus the latter would be defined as solely the Current Memory (like in a standard LSTM cell). For large values of Δ_t, the function g tends to 0 and the Short-Term Memory discounts more as it is attributed a higher weight in the expression above. In the limit situation where $g = 0$, the Adjusted Previous Memory would have the exact same formulation that of the Long-Term Memory defined previously in Equation (4).

2.2. Generative Imputation Methods

2.2.1 GAIN

The GAIN, much like any Generative Adversial Network (GAN), is made up of a generator (G) and a discriminator (D). The G is trained to maximise the D's misclassification rate while the D is trained to classify which data is real or generated. The method implemented by J. Yoon et al. [6] aims to learn the distribution of the data and to impute missing data.

The goal with the GAIN adaptation is to make the generator learn how to impute longitudinal data such that the discriminator cannot tell which data is real and which one is imputed, even when given a hint vector that suggests some points in the data

are imputed. To implement this variation of the GAN, first, it will be required to compute mask arrays (M), where each entry indicates if a value exists (taking value of 1) or not (taking value of 0) for each variable.

In a following step, the GAIN model will have to learn the distribution of the original dataset. To achieve this, the G will have to learn a mapping which tries to map this random noise vector into a realistic time series, where Z is a random noise vector, $\tilde{X}$ the original data at the input of the G, M the masking vector and $\bar{X}$ the imputed data.

$$\bar{X} = G(\tilde{X}, M, (1 - M) \odot Z), \quad (7)$$

$$\hat{X} = M \odot \tilde{X} + (1 - M) \odot \bar{X}. \quad (8)$$

The D will contain both the real, but incomplete, samples of data and the fake, but complete, data generated by G as well as a hint vector as the following figure depicts

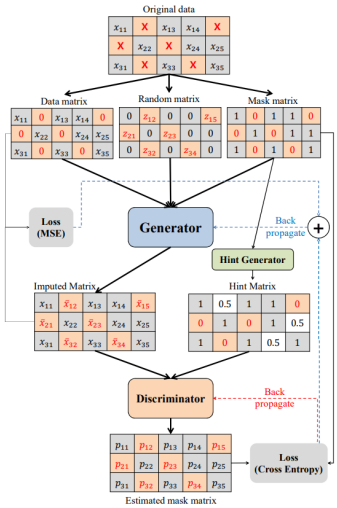

Figure 2: GAIN architecture, taken from [6]. In the Original data, X marks missing data, which is replaced with 0 when it is input in the generator.

As with any typical GAN, this variation is also trained adversarially. The training of the standard GAN would, for example, have D maximise the probability of distinguishing between a fake and a real data adversarial to the G minimising the probability of D distinguishing with success. The twist for the GAIN is that it is trained adversarially in respect to the mask, that is, D is trained to maximise the probability of predicting M while G is trained to minimise the probability of D successfully predicting M [6]. Thus, the formulation of the generator and the objective of the GAIN, taken from [6] are as follows

$$\min_G \max_D V(D,G) = \mathbb{E}_{\hat{X},M,H}[M^T log D(\hat{X}, H) + \\ + (1 - M)^T log(1 - D(\hat{X}, h))] \quad (9)$$

To highlight that this function is indeed in respect to the masking vector M, the following definition of the loss function (based on the cross-entropy) is defined by Y. Yoon et. al.

$$\mathbb{L}(a, b) = \sum_{i=1}^{d} [a_i \, log(b_i) + (1 - a_i) log(1 - b_i)]. \quad (10)$$

Considering that the discriminator outputs an estimated masking vector $\hat{M}$ as seen in figure 2

$$\hat{M} = D(\hat{X}, H). \quad (11)$$

When combining these last two expressions, the outcome of rewriting (9) highlights that the training is a function of the masking vector

$$\min_G \max_D V(D,G) = \mathbb{E}[\mathbb{L}(M, \hat{M})]. \quad (12)$$

Both the G and D in this work were composed of an LSTM layer of 128 units followed by a fully connected layer. The G had a learning rate of 0.01 while the D had a learning rate of 0.001.

The next step is to find the best vector z for any incomplete time series x such that $G(z)$ is similar to x. To achieve this, the discriminator outputs a mask matrix which will be compared with the real mask matrix by computing its cross-entropy and used to train the D. This loss, alongside with the Mean Squared Error (MSE) of the non masked points of the imputed matrix and of the data matrix, will be summed and used to train the G.

A new feature in this GAIN network is the hint mechanism. The hints are a subspace of the masked values. The discriminator is allowed to try and tell apart values that otherwise it would not be able to access due to the mask. This mechanism exists, according to the authors [6], as a way to prevent cases where there are several distributions that G could reproduce that would be optimal in respect to D but that would not be optimal for the imputation problem. This parameter was set to 0.9 in the course of this work.

Training these networks can be quite hard due to the mode collapse problem which happens due to the fact that the optimal generator for a fixed discriminator is a sum of deltas on the points the discriminator assigns the highest values [7].

2.2.2 β-VAE

VAE [8] [9] are a generative model constituted of an Autoencoder network and a random sampling layer. Autoencoders are networks with two parts: an encoder and a decoder. The encoder uses the data as input and, as output, it has a representation of the input data in reduced dimensionality, represented in what is called of latent space. The decoder takes this representation in the latent space

as input and decodes it as output, thus creating a reconstruction, as close as possible, of the original data. Autoencoders and VAE work for any kind of data and longitudinal data is no exception. They are thus designed to encode a representation of the input in a lower dimension space and then do the closest reconstruction possible in respect to the original data. Typically, an Autoencoder is trained in respect to reconstruction loss, using the MSE or cross-entropy as loss function [10].

In this work, the encoder had two layers and 128 and 64 units, respectively. The decoder also had two layers, with 64 and 128 units respectively.

The random sampling layer works by taking the output of the encoder and transforming it in two vectors, each one with the dimension of the latent space: σ for the standard deviation and μ for the mean of each variable. With these two vectors, the latent space becomes continuous and will follow a probability distribution due to some changes in how the loss will be calculated. Each encoding is now a point with an "area" given by its standard deviation, centred at the mean, in the latent space. Thus, not just the point centred at the mean is of its class but also the cluster in the surrounding "area".

The typical VAE can have its architecture represented as follows

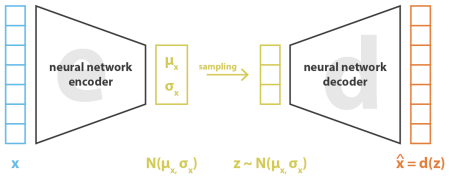

Figure 3: VAE architecture, taken from [11].

This random sampling changes the loss function into two components: one part for the reconstruction loss and another one for the Kullback-Leibler divergence (or KL loss, in the VAE context) of the distribution in the latent space. The Kullback-Leibler Divergence is, intuitively, how much two probability distributions diverge from each other. A value of 0 means that the two distributions are the same (no divergence) and the higher this value the more divergent are the two distributions. Its formal definition [12] is as follows

$$D_{KL}(p_1||p_2) = \sum_{x \in X} p_1(x) log \frac{p_1(x)}{p_2(x)}. \quad (13)$$

In the VAE, these loss functions are defined as follows for inputs of dimension p, latent space of dimension n and original data X_i and decoded data $\tilde{X}_i$:

$$\text{Reconstruction Loss} = \frac{1}{n} \sum_{i=1}^{p} (X_i - \tilde{X}_i), \quad (14)$$

$$\text{KL Loss} = \sum_{i=1}^{n} (\sigma_i^2 + \mu_i^2 - log(\sigma_i) - 1), \quad (15)$$

$$\begin{aligned} \text{Loss} = \text{Reconstruction Loss} + \text{KL Loss} = \\ = \frac{1}{n} \sum_{i=1}^{p} (X_i - \tilde{X}_i) + \\ + \sum_{i=1}^{n} (\sigma_i^2 + \mu_i^2 - log(\sigma_i) - 1). \end{aligned} \quad (16)$$

Notice that this KL Loss formulation is specific of the VAE. In order to assign a different weight to the KL Loss, as to control how much this term affects the overall loss and the distribution of the latent space, the following formulation of the loss is used

$$\begin{aligned} \text{Loss} = \frac{1}{n} \sum_{i=1}^{p} (X_i - \tilde{X}_i) + \\ + \beta \sum_{i=1}^{n} (\sigma_i^2 + \mu_i^2 - log(\sigma_i) - 1) \end{aligned} \quad (17)$$

and this changes, in literature, the name of this generative model from VAE to $\beta-$VAE [13]. In practice, changing β to 0 transforms the VAE in a regular Autoencoder. A very large value of β forces the latent space of the VAE to be centered at the origin and with little variance (i.e. the "area" around the origin is very small), whereas a very small value of β gives freedom to the latent space to take whichever shape the encoding process will produce. In the work, the value of β was set to 0.1, as it yielded the best results.

2.3. Datasets and Proposed Method for Classification

The TI are not constant between the ADNI clinical data, so the procedure was to acquire two sets of data: one with constant TI which were always 12 months apart between consultations, and another one that had no restrictions regarding the elapsed time between consultations. Both datasets have five consultations per patient, with the condition that every single consultation has a diagnosis. Thus, the first set of data was evenly spaced in time while the second was not.

This way, the LSTM network was used to constitute models for the generative imputation methods and for classification, which were trained and tested with the first set. The T-LSTM had high computational requirements and a complex TensorFlow 2.0 implementation. The latter caused incompatibilities between with the Keras based implementations of the generative models. Consequently, the second dataset only underwent the simple baseline imputation techniques and was only used to train and test the T-LSTM classification model.

Each one of the two datasets discussed in this chapter was firstly imputed with each technique and then saved. The only exception were the full datasets, which were obtained by dropping all patients with missing data. In a following step, the previously obtained datasets are used as input to train and then test their respective classification RNN model, which produces one classification per each consultation (thus producing five classifications in total per each patient).

3. Implementation Details

3.1. Forward

Filled in missing data by carrying forward the last known instance of data. In the implementation used in this work, in an instance where the missing data had no previously known value, the first (posterior) known value was carried backward. Patients were removed from the imputed dataset in case there existed no values of at least one feature across all of the five consultations.

3.2. Mean

The missing values are replaced by the mean of all known values of that feature. Notice that, similarly to the Forward Imputation, when there were no values for a feature across all consultations, the patient was removed from the imputed dataset.

3.3. Random

The implementation used in this work replaced the missing values with random values between the minimum and maximum of each feature across all patients. This means that the interval of values that can be randomly chosen for imputation depends on the minimum and maximum across all patients for that given feature.

3.4. Baseline Imputation Methods

With the purpose of constituting a baseline, Masking, Mean and Forward Imputations are the three proposed methods as they achieve state of the art results, are simpler than the GAIN or the β-VAE, are widely used [14], [15], [5], [16], [17] and can be applied to both the unevenly and evenly spaced datasets.

Masking required to fill in the missing values with a mask value. The Mean Imputation replaced the missing values with the mean of the known values and the Forward Imputation worked by replacing the missing values with the previously known value.

3.5. RNN models and their Parameters

Two models were designed in scope of this work with one main difference between each: one of the models used as RNN the LSTM while the other one used the T-LSTM. Each model was composed of a Masking layer (which only had any impact when the masked datasets are the input), followed by two RNN layers of the same dimension that use the hyperbolic tangent as activation function, as it is often found to converge faster than other activation functions. They are then followed by a fully connected layer with softmax as its activation function, which produced the outputs, i.e., the diagnosis of each patient at every single consultation.

The first LSTM and the first T-LSTM layers of each model had 20% and 30% dropout at input, respectively. Both models have 10 % recurrent dropout on each layer, as these values yielded the best results, and both models used as kernel and bias regularisers the L2 norm. All RNN layers had 256 units, as this amount of units was suggested to have yielded the best results through the state of the art [14], [15], [18], [17]. With these considerations, the LSTM based model had a total of 826115 parameters and the T-LSTM based model had 957699 parameters. The 15.93% more trainable parameters in the latter model are related to the differences that the T-LSTM model as seen in Figure 1.

The Adam optimizer [19] was used for both models but with different learning rates due to the Keras based implementation converging much quicker than the TensorFlow based one. Thus the LSTM model's learning rate was $1 \cdot 10^{-3}$ and the T-LSTM model's learning rate was $5 \cdot 10^{-6}$.

3.6. Training Process

Each dataset was divided into five folds which are then normalised, as four of the folds made the training set and the remaining fold made the test set in each of the five iterations. The training set was further split such that 10% of the total data in each iteration was used to form a validation set.

The model was trained using as cost function the categorical cross-entropy and the weights of the network that yielded the best results in the validation set were saved. This metric was being measured by obtaining the lowest validation loss during the training process.

Attending to the lack of AD patients throughout the consultations causing some class imbalance, a known strategy of class weighting present in the scikit-learn library was used with the following formulation for each class

$$\text{class weight} = \frac{\text{samples}}{\#\text{classes} \cdot \text{instances of class}}. \tag{18}$$

This strategy assigned weights which were larger the less present a certain class was, and smaller the more present a class was.

4. Results & Discussion

This task was performed for each one of the models and divided by dataset, which produced clas-

sification results by consultation. The results have three main metrics in which they will be evaluated across the consultations:

- Accuracy.
- F1 Score for each class in a one vs all setting.
- Receiver Operating Characteristic (ROC) for each class in a one vs all setting and their Area Under the ROC Curve (AUC).

The F1 Score was calculated for each model at each one of the k folds, while the ROC was calculated posterior to the concatenation of outputs of every model, meaning that this curve was singular for each imputed dataset. The F1 Score is calculated a total of k times and in the end the average F1 Score for each class was reported.

4.1. Results

The obtained results were split between both models had emphasis on the fifth (and last) consultation. The main results will be presented with more detail on the last consultation but results will be shown for all consultations.

The LSTM model had less patients in its datasets but the β-VAE and GAIN imputed datasets available to use. Thus, the results obtained with the previously discussed metrics at last consultation are reported in the two tables in the next page.

Notice that in parenthesis are the total number of patients with each diagnostic at the last consultation in each imputed dataset. To exemplify, 57, 39 and 36 are the number of patients with CN, MCI and AD diagnosis, respectively, for the dataset containing only full data in Table 1. The best results for either model are highlighted in bold.

The LSTM model's dataset was smaller than the T-LSTM model's dataset and, despite this, the numbers of patients with each labels varied within each table. This is due to how each imputation method works. The Full dataset had the least patients, as it excluded every patient with any instance of missing data. The Mean and Forward imputed datasets had the same amount of patients, which was lower than the amount of patients present in the β-VAE and GAIN imputed datasets and in the Masked dataset. This occurred due to situations where all the values for at least one feature are missing and thus neither of the two former methods can produce a value to fill in the missing value.

Each table has a relatively similar range of results throughout methods and models. In the first table, the LSTM model's results can be placed in two groups: the first one with the Full dataset, Forward and Mean imputed datasets and the second one with the Masked, β-VAE, GAIN and Random imputed datasets. The former group had a generally lower standard deviation for the Test F1 Score while having lower values of both Test F1 Score and Test AUC for the MCI and AD classes. The latter group had higher standard deviation in the Test F1 Score in theImputation, especially in terms of F1 Score CN and MCI classes and lower standard deviation in the AD class. Its test AUC values for the MCI and AD classes were also higher than for the first group.

In terms of Test Accuracy, the second group performs best due to having lower standard deviation and higher average values than the first group, which is especially true for both the β-VAE and GAIN imputed datasets. The condition that RNN have of requiring large datasets to output better results might also explain the performance difference of the two groups, as the second group of results performed best because the imputations at stake allowed for much bigger sets of data. The separation of the results in two was also a consequence of the size of the datasets of each group. Bigger datasets have generally much better performance for the MCI and AD classes. Across most metrics, the VAE Imputation yielded the best results, closely followed or even surpassed by the GAIN Imputation. The results from the β-VAE Imputation were considered best as they had lower standard deviation than the GAIN Imputation and the highest overall F1 Score values.

The second table contains the T-LSTM model's results. The Mean imputed dataset had the lowest standard deviation across the metrics while the Masked dataset had the highest standard deviations for almost every metric. Across all metrics, the Forward Imputation yielded the best results and was closely followed by the Random Imputation, especially in terms of F1 Score. This was due to how data in the dataset starts missing in later consultations rather than in earlier ones and thus, with the progression of AD, which is irreversible, an imputation technique that uses the last known value is more prone to yield better results than one that averages all known consultations for the same patient, like the Mean imputation. This reasoning on the differences in performance between the Forward and Mean imputed datasets applies for both models, as in either dataset they also had higher missing rates in later consultations. It is also worth to emphasise that the number of patients in this model was much higher than in the LSTM model. This was due to the possibility of using non-equally temporally spaced patients that this model has.

In both tables, the Masked datasets had the highest overall standard deviation rates in the Test F1 Score, despite not performing poorly in any

Dataset (CN-MCI-AD)	Test Accuracy (%)	Test F1 Score (% per class)			Test AUC (per class)		
		CN	MCI	AD	CN	MCI	AD
Full (57-39-36)	80.26±4.64	**90.81±5.95**	59.42±9.99	79.17±4.20	0.961	0.847	0.956
Forward (86-65-64)	79.07±3.89	89.39±4.16	64.14±4.53	78.18±2.14	0.970	0.857	0.956
Mean (86-65-64)	76.28±7.11	87.86±5.36	56.85±11.68	77.18±6.21	**0.978**	0.865	0.904
Mask (105-223-101)	79.49±3.55	66.13±20.53	62.72±29.12	**83.42±2.54**	0.960	0.910	**0.972**
β-VAE (105-223-101)	81.11±3.47	73.13±19.01	**75.60±10.22**	79.95±3.85	0.967	0.913	0.969
GAIN (105-223-101)	**81.12±4.13**	70.66±19.92	72.55±17.29	82.23±3.08	0.969	**0.918**	0.971
Random (105-223-101)	77.16±2.80	69.68±18.80	64.01±22.39	79.14±2.02	0.960	0.894	0.963

Table 1: Average across all folds of the LSTM results of Accuracy and Test F1 Score and their respective Standard Deviations, as well as with the Test AUC which was calculated off the ROC.

Dataset (CN-MCI-AD)	Test Accuracy (%)	Test F1 Score (% per class)			Test AUC (per class)		
		CN	MCI	AD	CN	MCI	AD
Full (123-163-60)	74.28±1.48	89.08±1.76	**67.22±8.36**	60.68±3.94	0.968	0.864	0.905
Forward (180-229-96)	73.47±6.15	89.65±4.15	64.14±9.12	**63.47±6.18**	**0.978**	0.877	0.919
Mean (180-229-96)	72.67±5.62	**90.59±2.91**	63.17±2.60	61.15±6.90	**0.978**	0.865	0.904
Mask (296-509-139)	72.46±6.32	87.58±4.79	66.09±13.91	56.54±8.05	0.966	0.870	0.922
Random (296-509-139)	**73.94±5.90**	85.15±6.50	71.65±6.24	57.77±8.88	0.970	**0.882**	**0.924**

Table 2: Average across all folds of the T-LSTM results of Accuracy and Test F1 Score and their respective Standard Deviations, as well as with the Test AUC which was calculated off the ROC.

other metric. This is explained by the amount of data that this method masks. The more instances of data that were masked, the less data the network used in its training process. As RNN perform better the more data they have available to be trained on, having less data available worsens their performance and thus the performance of Masking.

Comparing the two tables, the Test Accuracy was lower in the T-LSTM model and this model had a generally higher standard deviation. The Test F1 score was more often higher with a lower standard deviation for the CN and MCI classes for the T-LSTM model. On the other hand it was overall lower with a higher standard deviation for the AD class. The Test AUC was similar for the CN class and slightly lower on both the MCI and AD classes for the T-LSTM model. This means that, even though the T-LSTM model had more patients in its datasets and used the TI between consultations, it did not yield better results for the AD class, despite performing better for the other two classes. This effect happens as almost no patient began with an AD diagnosis and, as the TI value was only zero at the first consultation, all the subsequent consultations had their weights discounted due to how the Adjusted Previous Memory was calculated, as seen in Equation 5. This way, the learned generalisation for this class was of lower quality than in the LSTM model.

Another way to present these results which has a better graphical visualisation is through the usage of boxplots, side by side, of each kind of metric for each dataset, in the figures present in the next page.

The first boxplot highlights that the Test Accuracy, in median, was higher for the LSTM model and had less standard deviation. This representation also gives a visual intuition to the negative impact that higher standard deviations and outliers have to the Test Accuracy. The second boxplot reinforces what previously was discussed, namely that, for the CN and MCI classes, the T-LSTM model produced better results while for the AD class the LSTM model was best.

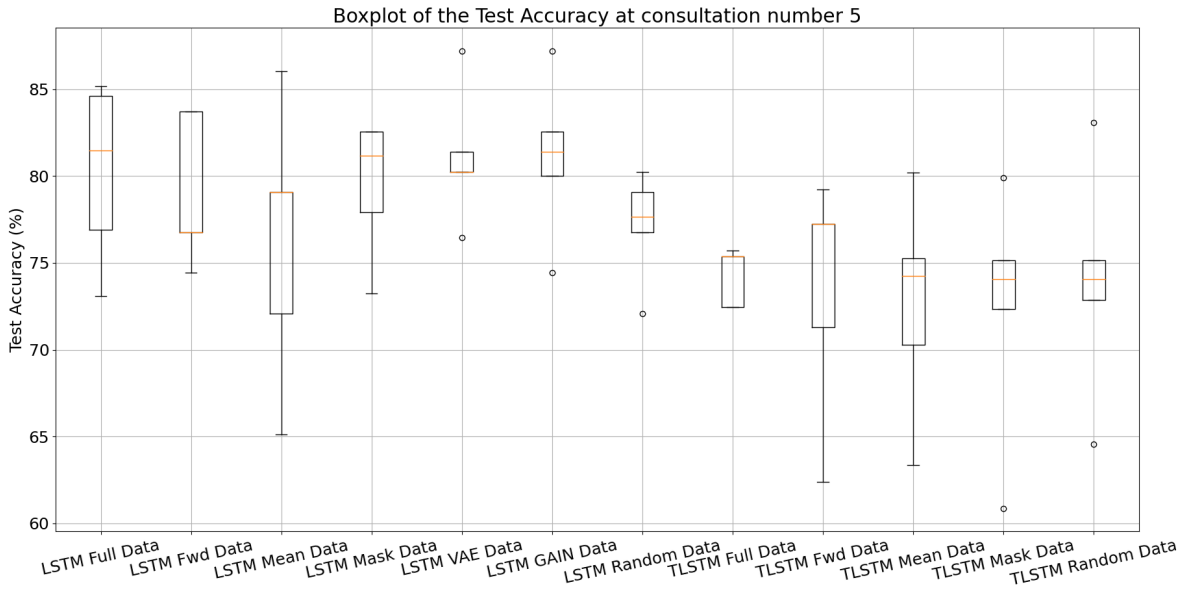

Figure 4: Boxplots of the Test Accuracy each method side by side. The yellow lines mark each median.

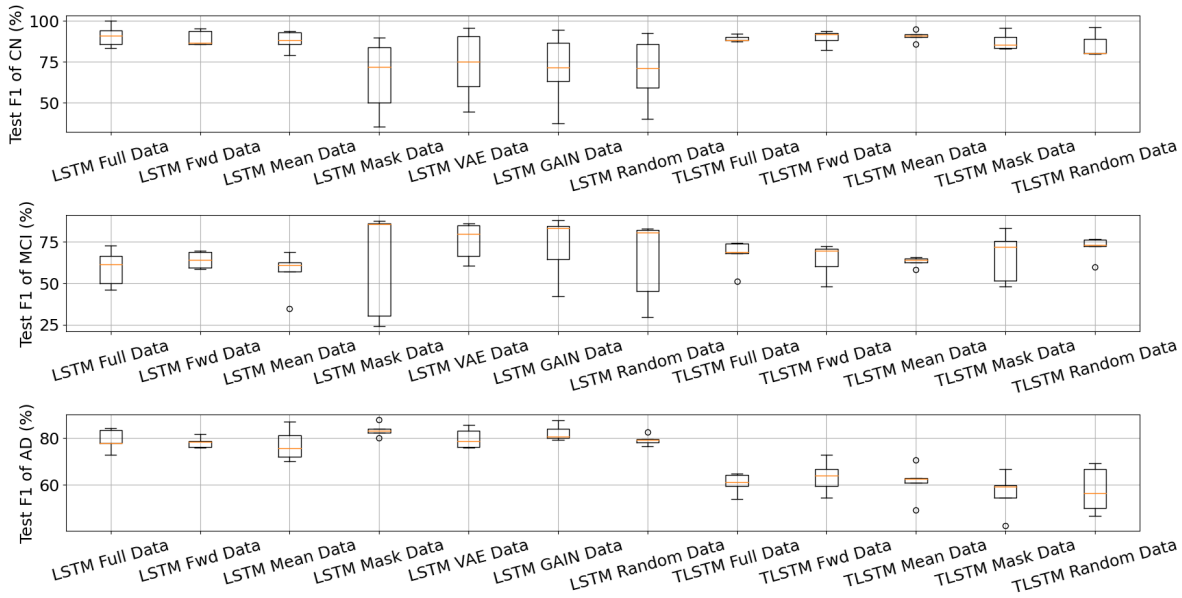

Figure 5: Boxplots of the F1 Score of each class and each method side by side. The yellow lines mark each median.

Finally, in either model, through the previous tables and figures, it is concluded that the Random Imputation is always competitive with both the baseline methods and the proposed methods. While this doesn't invalidate the comparisons between the different imputations, it leads to a conclusion that imputation was not necessary for the datasets used in this work.

5. Conclusions

In the present setup, the generative methods produced results which were competitive, in terms of imputation quality, with the standard widely used methods, especially for lower rates of missing data. These results could have been more competitive with the state of the art but they show potential, especially the β-VAE and GAIN.

In the course of this work, the obtained results were comparable to the state of the art, surpassing some results in terms of Test Accuracy while employing less patients than in the work of M. M. Ghazi et al., 2019 [17]. On the other hand, not using all relevant features for this disease hinders

the results of this work and brings its Test Accuracy down. The CSF and PET measurements, which had too high missing rates, as previously discussed, could not be used. In part, this lack of features was a reason why the obtained results were behind some other state of the art results such those of Z .C Lipton et al. 2016 [15]. It was also the main reason why the Random Imputation performs competitively with the other imputations.

One of the advantages of a T-LSTM based model was that, through the usage of the TI, it enabled more patients to be used in its dataset. Despite the much larger amount of patients, the results were not better across all metrics than those of an LSTM based model. In fact, this novel model had lower Test Accuracy than the LSTM model. Despite having had better F1 Scores for the CN and MCI patients, the T-LSTM model performed worse for AD patients in this metric. Some of the performance differences between the results of the two models are explained with the dataset size differences, namely the differences in standard deviation between the two models. However, the mechanism of the Adjusted Previous Memory, with its current formulation, could potentially be reducing the Test Accuracy of this model and the F1 Score of the AD class. This said, considering how RNN need large datasets, models of networks such as the T-LSTM allow for usages of more data than the LSTM, which is a big advantage.

As the remaining consultations are introduced, a pattern emerged. The results were relatively good in the first two consultations due to the problem being between a binary classification task, as there are almost no AD patients in the datasets of either model so early. At the third consultation onward, the classification task became more difficult as the MCI patients developed AD and, from that point on, there are more AD patients. Datasets where all three classes would be present throughout every stage without major class imbalances would make mitigate this situation.

6. Further Work

There are, however, some changes which could be made to this work and potentially could be relevant not just for these methods but also for handling longitudinal data with RNN in general.

The same way that each class can be attributed weights with formulations such as the one in Equation 18, the same school of thought could be applied to weight each sequence (or, in this case, for each consultation). At the moment there were not any examples of this concept in the literature and thus different ways to weight each sequence could be theorised, for example, in the following ways:

- Each sequence's weight could depend on relevant features of the data such as, to list some examples, the age of a patient, the TI between consultations, the label(s) of that sequence, or some (non) linear decay function.
- Each individual sequence's weight depending on characteristics typical of the task. To exemplify, in AD related works, it would make sense that some consultations are more key to diagnose someone with this pathology, and thus these said key consultations would have larger weights than the others.
- The sequences of patients that, for example, develop AD despite prior MCI or CN diagnoses, could have a larger weight than the sequences of patients that have a constant diagnosis for every consultation. That is, sequences where there is an evolution of a patient's status could be given more importance in the model's learning.

References

[1] Alzheimer's Disease and Alzheimer's Dementia. (accessed: 13.04.2020). [Online]. Available: https://www.alzheimer-europe.org/Dementia/Alzheimer-s-disease-and-Alzheimer-s-dementia

[2] European Federation of Pharmaceutical Industries and Associations. (accessed: 13.04.2020). [Online]. Available: https://www.efpia.eu/we-wont-rest/innovation/alzheimer-s-disease/

[3] National Institute on Aging - Alzheimer's Disease Fact Sheet. (accessed: 29.05.2020). [Online]. Available: https://www.nia.nih.gov/health/alzheimers-disease-fact-sheet

[4] F. Chollet, *Deep Learning with Python*. Manning Publications Co., 2018.

[5] I. M. Baytas, C. Xiao, X. Zhang, F. Wang, A. K. Jain, and J. Zhou, "Patient Subtyping via Time-Aware LSTM Networks," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 65–74. [Online]. Available: https://doi.org/10.1145/3097983.3097997

[6] J. Yoon, J. Jordon, and M. van der Schaar, "GAIN: Missing Data Imputation using Generative Adversarial Nets," in *Proceedings of the 35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, J. Dy and

A. Krause, Eds., vol. 80. PMLR, 10–15 Jul 2018, pp. 5689–5698. [Online]. Available: http://proceedings.mlr.press/v80/yoon18a.html

[7] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative Adversarial Nets," in *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, ser. NIPS'14. Cambridge, MA, USA: MIT Press, 2014, p. 2672–2680. [Online]. Available: https://dl.acm.org/doi/10.5555/2969033.2969125

[8] D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes," in *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014. [Online]. Available: https://arxiv.org/pdf/1312.6114.pdf

[9] C. Doersch, "Tutorial on Variational Autoencoders," 2016. [Online]. Available: https://arxiv.org/pdf/1606.05908.pdf

[10] I. Shafkat. Intuitively Understanding Variational Autoencoders - Towards Data Science. (accessed: 02.01.2021). [Online]. Available: https://towardsdatascience.com/intuitively-understanding-variational-autoencoders-1bfe67eb5daf

[11] J. Rocca and B. Rocca. Understanding Variational Autoencoders (VAEs) - Towards Data Science. (accessed: 05.02.2021). [Online]. Available: https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73

[12] S. Kullback, *Information Theory and Statistics*. John Wiley & Sons Inc., 1959.

[13] C. P. Burgess, I. Higgins, A. Pal, L. Matthey, N. Watters, G. Desjardins, and A. Lerchner, "Understanding disentangling in β-VAE," *CoRR*, vol. abs/1804.03599, 2018. [Online]. Available: http://arxiv.org/pdf/1804.03599.pdf

[14] M. Nguyen, N. Sun, D. C. Alexander, J. Feng, and B. T. T. Yeo, "Modeling Alzheimer's Disease Progression using Deep Recurrent Neural Networks," in *2018 International Workshop on Pattern Recognition in Neuroimaging (PRNI)*, 2018, pp. 1–4. [Online]. Available: https://doi.org/10.1016/j.neuroimage.2020.117203

[15] Z. C. Lipton, D. Kale, and R. Wetzel, "Directly Modeling Missing Data in Sequences with RNNs: Improved Classification of Clinical Time Series," in *Proceedings of the 1st Machine Learning for Healthcare Conference*, ser. Proceedings of Machine Learning Research, F. Doshi-Velez, J. Fackler, D. Kale, B. Wallace, and J. Wiens, Eds., vol. 56. Northeastern University, Boston, MA, USA: PMLR, 18–19 Aug 2016, pp. 253–270. [Online]. Available: http://proceedings.mlr.press/v56/Lipton16.html

[16] Z. Che, S. Purushotham, K. Cho, D. Sontag, and Y. Liu, "Recurrent Neural Networks for Multivariate Time Series with Missing Values," *Scientific Reports*, vol. 8, no. 1, p. 6085, 2018. [Online]. Available: https://doi.org/10.1038/s41598-018-24271-9

[17] M. Mehdipour-Ghazi, M. Nielsen, A. Pai, M. J. Cardoso, M. Modat, S. Ourselin, and L. Sørensen, "Training Recurrent Neural Networks robust to incomplete data: Application to Alzheimer's Disease progression modeling," *Medical Image Analysis*, vol. 53, p. 39–46, 2019. [Online]. Available: https://doi.org/10.1016/j.media.2019.01.004

[18] T. Wang, R. G. Qiu, and M. Yu, "Predictive Modeling of the Progression of Alzheimer's Disease with Recurrent Neural Networks," *Scientific Reports*, vol. 8, no. 1, p. 9161, 2018. [Online]. Available: https://doi.org/10.1038/s41598-018-27337-w

[19] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015. [Online]. Available: http://arxiv.org/pdf/1412.6980.pdf