Tree Memory Networks for Modelling Long-term Temporal Dependencies

Tharindu Fernando[a,*], Simon Denman[a], Aaron McFadyen[b], Sridha Sridharan[a], Clinton Fookes[a]

[a]*Image and Video Research Laboratory, SAIVT, Queensland University of Technology, Australia.*
[b]*Robotics and Autonomous Systems, Queensland University of Technology, Australia.*

Abstract

In the domain of sequence modelling, Recurrent Neural Networks (RNN) have been capable of achieving impressive results in a variety of application areas including visual question answering, part-of-speech tagging and machine translation. However this success in modelling short term dependencies has not successfully transitioned to application areas such as trajectory prediction, which require capturing both short term and long term relationships. In this paper, we propose a Tree Memory Network (TMN) for jointly modelling both long term relationships between multiple sequences and short term relationships within a sequence, in sequence-to-sequence mapping problems. The proposed network architecture is composed of an input module, controller and a memory module. In contrast to related literature which models the memory as a sequence of historical states, we model the memory as a recursive tree structure. This structure more effectively captures temporal dependencies across both short and long term time periods through its hierarchical structure. We demonstrate the effectiveness and flexibility of the proposed TMN in two practical problems: aircraft trajectory modelling and pedestrian trajectory modelling in a surveillance setting. In both cases the proposed approach outperforms the current state-of-the-art. Furthermore, we perform an in depth analysis on the evolution of the memory module content over

*Corresponding author at: Image and Video Research Laboratory, SAIVT, Queensland University of Technology, Australia.

Email addresses: t.warnakulasuirya@qut.edu.au (Tharindu Fernando), s.denman@qut.edu.au (Simon Denman), aaron.mcfadyen@qut.edu.au (Aaron McFadyen), s.sridharan@qut.edu.au (Sridha Sridharan), c.fookes@qut.edu.au (Clinton Fookes)

time and provide visual evidence on how the proposed TMN is able to map both short and long term relationships efficiently via a hierarchical structure.

Keywords: Memory Networks, Trajectory Prediction, Recurrent Networks

1. Introduction

Sequence-to-sequence modelling is a vital element in machine learning and knowledge representation, with multiple application areas including machine translation [1], trajectory prediction [2], and part-of-speech tagging [3]. This problem can be represented as predicting an output sequence, $Y = [\mathbf{y_1}, \ldots, \mathbf{y_T}]$, given an input sequence $X = [\mathbf{x_1}, \ldots, \mathbf{x_T}]$. Predicting a future element, $\mathbf{y_t}$, of the sequence at time instance t, utilising the current input to the model, $\mathbf{x_t}$, and the content of the memory from the previous time step, $\mathrm{M_{t-1}}$, can be represented as,

$$\mathbf{y_t} = \mathbf{f}(\mathbf{x_t}, \mathrm{M_{t-1}}). \tag{1}$$

Modelling long term relationships in between sequences can be considered one of the most challenging problems within the machine learning community [4, 5]. Although many memory architectures proposed for sequence-to-sequence modelling are capable of mapping short term relationships, they are less successful when handling long term dependencies [6].

Long term relationships within the data are extremely useful and can significantly influence the accuracy of the predictions when considering the repetitive nature of many processes. For instance consider an air traffic modelling problem. Even though short term dependencies such as current weather and neighbouring traffic are the most influential factors, the repetitive nature of the aircraft trajectories and the flight schedules suggests that one can easily deduce a coherence among trajectories over a period of days and/or seasons. For example, at a point in time a certain runway may be in use, so similar trajectories should be observed over the short term, but a sudden change of runway (due to weather) means "new" trajectories will be seen. Having a long term memory means that these are not actually "new" trajectories, but can instead be recalled from an earlier, similar weather event or operational configuration.

A similar logic can be applied when modelling pedestrian behaviour in a surveillance scenario, such as in the example shown in Fig. 1. While the current location of neighbouring pedestrians is most influential, one cannot discard the influence of historical behaviour under similar contexts and events. Pedestrians may be wandering in a free area, and as new trains arrive a new "flow" of

2

(a) High pedestrian flow in □ (b) Low pedestrian flow in □ (c) Similar flow to (a) in □

Figure 1: Pedestrian flow at different times in the Grand central dataset [7]. A high pedestrian flow in the highlighted area is observed at the 07:08 time stamp with the arrival of a train before the flow decreases at the 10:32 time stamp. A similar flow to subfigure (a) is observed at the 21:29 time stamp confirming our hypothesis that future pedestrian flow can be anticipated with the aid of historical data.

pedestrian movement appears in response to the congestion. But as we posses historical data from similar contexts, one should be able to accurately anticipate such pedestrian motion.

In this paper we are interested in efficiently aggregating such long term dependencies among input data. In the sample scenario from the Grand Central dataset [7] presented in Fig. 1, a high pedestrian flow in the highlighted area is observed at the 07:08 time stamp with the arrival of a train before the flow decreases at the 10:32 time stamp. A similar flow to Fig. 1 (a) is observed at the 21:29 time stamp in Fig. 1 (c), confirming our hypothesis that future pedestrian flow can be anticipated with the aid of historical data.

As such, we propose an augmented memory architecture which can be generalised to any sequence to sequence modelling problem. The contributions of this work can be summarised as follows:

1. A new recursive memory network architecture capable of modelling long term temporal dependencies, using an efficient tree structure.
2. Application of the proposed memory architecture to two practical problems: aircraft trajectory modelling and pedestrian trajectory modelling in a surveillance setting, where in both cases we are able to achieve state-of-the-art results.
3. An in depth analysis on the evolution of the memory module content, where we study the changes in hidden state representations over time and discuss interpretable patterns.

The two applications we demonstrate the proposed approach on, aircraft tra-

3

jectory prediction and pedestrian trajectory prediction, are both related sequence-to-sequence modelling tasks, however they have distinct characteristics that illustrate the adaptability of the proposed approach. Aircraft trajectories are primarily a function of the flight schedule, which is typically fixed over a week, but still varies according to changes such as weather and off schedule arrivals/departures. Pedestrian trajectories however are less dependent on a schedule and are more influenced by the behaviour of other nearby pedestrians.

We would like to emphasise the fact that even though we are demonstrating our approach on two different application scenarios from the trajectory prediction domain, the varied nature of these problems demonstrates how the proposed model can be directly applied to any sequence-to-sequence prediction problem where modelling long term relationships is necessary. Possible application areas include diver behaviour modelling for autonomous driving [8, 9, 10], text and video synthesis [11], and context aware machine translation [12].

2. Related works

Related work within the scope of this paper can be categorised into memory architectures (Section 2.1), aircraft trajectory prediction approaches (Section 2.2) and pedestrian trajectory prediction approaches (Section 2.3).

2.1. Memory architectures

Deep learning models such as Recurrent Neural Networks (RNN) have been applied extensively for many sequence-to-sequence modelling problems and have been capable of producing state-of-the-art results. A number of approaches [13, 14, 15, 16, 17, 10, 18] have also utilised what are termed "memory modules", to aid prediction. The memory stores important facts from historical inputs and then generates the future predictions based on the stored knowledge. A sample architecture with an input module, controller and an external memory is shown in Fig. 2. Firstly the input module generates a vector representation, c_t, for the input, x_t, at time instance t. The controller then triggers a memory read operation. The memory module, with an attention process, searches the history and outputs relevant facts. The final output is generated by merging c_t with the memory output. Finally, the controller triggers a memory update operation where the memory, M_{t-1}, is updated with c_t.

The authors in [13] have utilised a memory module to improve performance for natural language processing tasks. Their proposed memory architecture is not fully extendible given the use of an offline feature engineering process using a
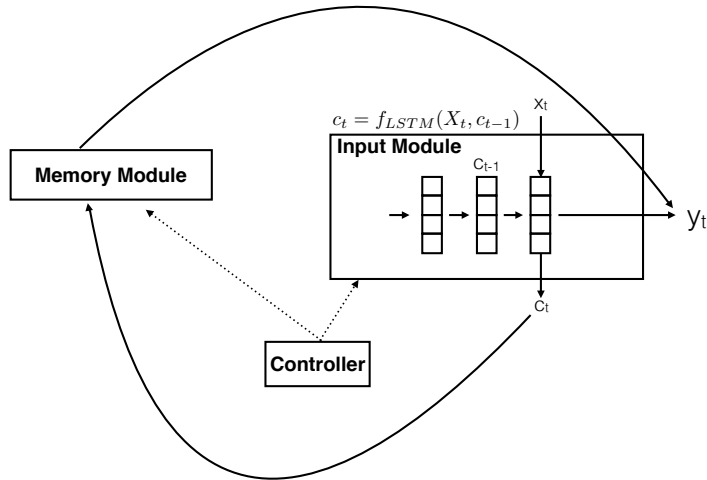
Figure 2: Neural Network architecture with an external memory. The memory is used to store important historical facts which can be utilised for future predictions. The controller is responsible for issuing read and write commands in order to take out and write back to the memory. An input module is used to encode and generate a vector representation from the input
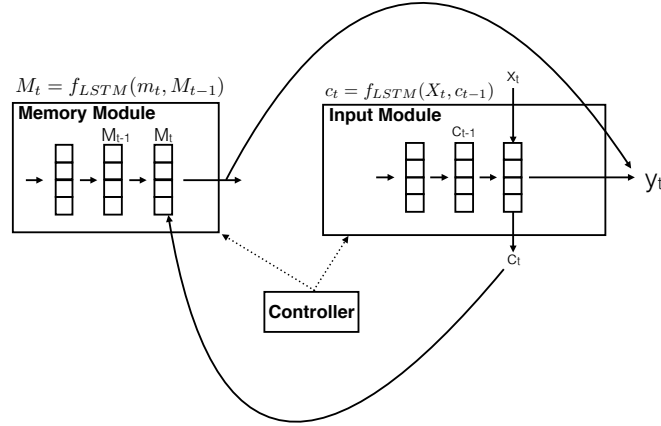
bag-of-words approach. In similar works, such as [16] and [19] for image caption generation, and [15] and [20] for visual question answering, the authors have extensively applied the notion of external memory. The memory architecture, "episodic memory", proposed in [14] has been shown to be capable of outperforming the other external memory architectures noted above [13, 15, 16, 19, 20] in terms of accuracy.

Fig. 3 (a) depicts the episodic memory model proposed in [14]. The authors model the "episodic memory" as a hierarchical recurrent sequence model utilising the sequential nature of the memory. The authors propose a generalised neural sequential module with recurrent LSTM memory cells for sequence encoding, memory mechanism and response generation. The above work is further extended in [21] through incorporating a shared memory architecture. Even with the exemplary results for short term dependency modelling problems, none of the above stated architectures are capable of handling sequences with long term relationships.
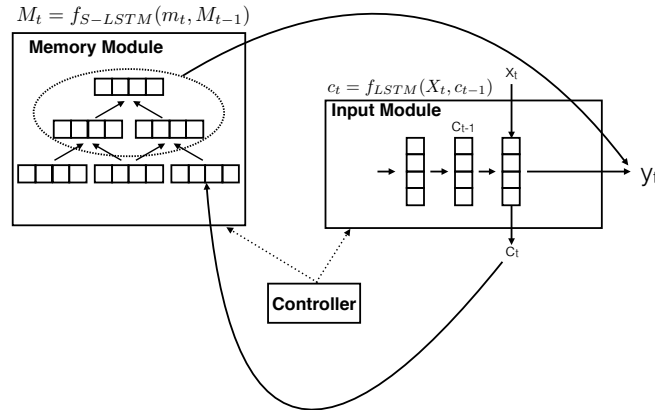
In approaches such as [14, 21, 22] the memory is a composed of a single layer of memory units. The memory update mechanism in [14, 21] can be written as,

$$\mathrm{M_t} = f_{LSTM}(\mathbf{m_t}, \mathrm{M_{t-1}}), \tag{2}$$

where $\mathbf{m_t}$ is a score value that quantifies the relevance of the content of the mem-

5

(a) Episodic memory model proposed in [14] with LSTM memory cells

(b) Proposed TMN model with S-LSTM memory cells.

Figure 3: Comparison of the memory model proposed in [14] (a) with the proposed memory model (b). In both approaches at time instance t, the input module generates representation c_t for the input x_t. Then the controller triggers a memory read operation. The memory module, with the attention process, outputs relevant facts. The final output is given by merging c_t with the memory output. Finally the memory update operation updates the memory, M_{t-1}, with c_t.

ory module (M_{t-1}) at time $t-1$ to the current context, c_t (see 3 (a)); where as in [22] the authors completely update the content of the memory locations based on m_t.

The main drawback of using approaches such as [14, 21, 22] with long term

dependency modelling and big data sets is that, in most cases, the attention mechanism will generate small scores for all the examples as they are all dissimilar to the input. Therefore with those approaches we are required to maintain an extremely large memory sequence as similar inputs only occur after long time intervals. Furthermore, in recurrent models such as in LSTMs, when the sequence becomes too long the output becomes biased towards recent observations [23], rather than considering the entire set of prior observations equally.

The work of Liang et. al [24, 25] on semantic object parsing has also provided an in-depth analysis on the limitations of sequentially structured LSTMs and how it affects the information flow when modelling data with complex, multilevel correlations. In [24] they propose a graph structured LSTM network where they model the contextual dependencies within an image at the super pixel level at the lowest level of the graph. This idea of hierarchical modelling is extended in [25] in order to have a dynamically evolving multi-level graph structure instead of a static hierarchy as in [24]. They achieve state-of-the-art results for semantic segmentation via this hierarchical representation learning approach. However, in contrast to their work, which focuses on learning semantic correspondences within a particular example, we are interested in learning long range temporal dependencies in between examples.

Hypothetically, if the memory module has enough non-linearity and if we have a sufficiently large database, any output should be capable of being produced from the contents of the memory. Furthermore the model should be capable of learning and modelling both short and long term contextual effects from the memory contents. This can be seen as a dictionary learning process where the memory is the dictionary being learnt. The memory module should learn to produce an accurate prediction for different combinations of inputs and historical trajectories.

Recently the recursive LSTM (S-LSTM) [26] was proposed where the authors extend the sequential LSTM to tree structures, in which a memory cell can reflect the historical memories of multiple child cells or multiple descendant cells in a recursive process. The authors in [23] have performed an in depth analysis of the strengths and weaknesses of sequential and recursive structures for a neural language modelling task; and found syntactical relationships, such as structure and logic in the input data are best captured via a recursive LSTM (i.e. S-LSTM). In contrast when encoding the semantics of sentences, sequential LSTM models provide state-of-the-art results.

The proposed model is illustrated in Fig 3 (b). Motivated by the positive characteristics that the S-LSTM architecture exhibits such as feature compression power and the preservation of semantic relationships among data, our approach

eschews a sequence representation in favour of a tree-based approach. A detailed analysis of the architecture in comparison to state-of-the-art methods is presented in Section 3.3.

2.2. Aircraft trajectory prediction approaches

Approaches such as [27, 28] utilise probability models for aircraft dynamics to generate predictions of future aircraft motion. They rely solely on the assumptions made regarding the dynamics of the aircraft. Importantly, they ignore all historic information, constituting a major drawback.

In [29, 30, 31] researchers treated aircraft trajectory prediction as a machine learning problem, in which they train the model using historical trajectory data together with weather observations. Most recently the authors in [32] proposed an approach that considered trajectories as a set of 4 dimensional data cubes, together with weather parameters. Initially they performed time series clustering on data for segmentation and then learnt a HMM for each cluster. However due to the uncertainty with weather observations, these trajectory prediction approaches become inefficient.

Several efforts have been made to improve the trajectory prediction by better wind estimation [33, 34, 35, 36, 37], yet these approaches have failed to achieve significant improvement in the task of trajectory prediction. Furthermore we would like to emphasise the fact that all of the above stated approaches consider aircrafts individually, without considering the air traffic within the neighbourhood, completely discarding important factors such as the volume and the proximity of nearby air traffic. Even though weather is a vital factor for future predictions it is implicit in the behaviour of neighbouring traffic. Therefore, inferring a notion of weather through neighbouring traffic is computationally inexpensive compared to tedious interpolations that ground based weather observations require [32].

2.3. Pedestrian trajectory prediction approaches

When considering the literature for human behaviour prediction the social force model [38, 39, 40, 41, 42] and its variants can be considered to be well-established. Such approaches generate attractive and repulsive forces between pedestrians, and thus define the optimal path under different contexts with respect to the neighbourhood.

Despite the prevalence of social force models, a number of probabilistic approaches have also been proposed. Zhou et al. [43] proposed a a mixture model approach, but this technique ignored the interactions among pedestrians. Wang

8

et al. [44] proposed a "topic model" which was extended to incorporate spatio-temporal dependencies in [45] and [46]. All of the above stated approaches use hand-engineered features as the input to the prediction module, which can be considered their main drawback as they fail to account for the semantics of the scene. Depending on the domain knowledge of the feature engineer, hand-crafted features may only capture abstract semantics of the environment.

Alahi et al. [47] removed the need for hand-crafted features via an unsupervised feature learning approach. The authors encode the trajectory of each pedestrian in the scene at that particular time using LSTMs. The hidden states of the neighbouring pedestrians at the immediately preceding time step are used in generating their position at the current time step. As pointed out in [48], this approach is only able to generate reactive behaviours such as collision avoidance, and fails to generate smooth trajectories for long term trajectory planning. Fernando et al. [48] have extended the idea of [47] to incorporate the entire trajectory of the pedestrian of interest as well as the neighbouring pedestrians. To the best of our knowledge none of the literature addressing human behaviour prediction has considered the long-term relationships among human behavioural patterns. Motivated by this limitation, we intend to explore the utility of temporal data for trajectory prediction via a tree memory network.

3. Tree Memory Network (TMN) Model

In this work, we are motivated by the exemplary results that were achieved from a tree structure for the task of discriminative dictionary learning from trajectories in [49]. In contrast to mapping all the historic data with a shallow layer of recurrent memory cells, we hierarchically map the memory with a bottom up tree structure where all historic states are represented in the bottom layer of the tree, and as we progress up the hierarchy we concatenate the most significant features in order to generate the output at a particular time step.

Furthermore, rather than stacking individual recurrent layers such as in [50, 51, 52], we utilise a Tree-LSTM structure as it focusses only on the historical information from its two neighbours. Therefore it can be seen as propagating significant features from two temporally adjacent neighbours to the upper layer.

3.1. Input Module

Let $X^i = [\mathbf{x_1^i}, \mathbf{x_2^i}, \ldots, \mathbf{x_T^i}]$ be the ith input sequence where T represents the number of time steps. The input module computes a vector representation $\mathbf{c_t}$ for

9

the input sequence via a LSTM layer,

$$\mathbf{c_t} = f_{LSTM}(\mathbf{x_t}, \mathbf{c_{t-1}}), \tag{3}$$

where $\mathbf{c_t} \in \mathbb{R}^k$, and k is the embedding dimension of the LSTM.

3.2. Memory Module

Consider $N \in \mathbb{R}^{p \times k}$ as the sequence of historical LSTM embeddings, with length p and embedding dimension k, that we would like to model as the memory. It can be seen as a queue structure with length p and each data element within the queue has a dimension of k. We adapt the S-LSTM model to represent the memory module of the proposed framework. It extends the general sequential structure in LSTMs to a bottom up tree structure composed of a compressed representation of children nodes to a parent node. This provides us with a principled way of considering long-distance interactions between memory inputs, and avoids the current drawbacks of LSTM models when handling lengthy sequences [23]. For simplicity, we represent the recursive-LSTM structure as a binary tree, where each parent node has two child nodes; however the extension of this model to any tree structure is straightforward.

3.2.1. Memory Read

When computing an output at time instance t we extract out the tree configuration at time instance $t-1$. Let $M_{t-1} \in \mathbb{R}^{k \times (2^l-1)}$ be the memory matrix resultant from concatenating nodes from the tree from the top to $l = [1, \ldots]$ depth. This allows us to capture different levels of abstraction that exist in our memory network. Let f^{score} be an attention scoring function which can be implemented as a multi-layer perceptron [53],

$$\mathbf{m_t} = f^{score}(\mathrm{M_{t-1}}, \mathbf{c_t}), \tag{4}$$

$$\alpha = softmax(\mathbf{m_t}). \tag{5}$$

Eq. 4 and Eq. 5 provide an attention mechanism that finds the most relevant memory items given the current input,

$$\mathbf{z_t} = \mathrm{M_{t-1}}\alpha^T. \tag{6}$$

Then the final output can be represented as,

$$y_t = ReLU(W_{out}\mathbf{z_t} + (1 - W_{out})\mathbf{c_t}), \tag{7}$$

where W_{out} are the output weights.

10

3.2.2. Memory update

In the proposed memory architecture each memory cell contains one input gate, i_t, one output gate, o_t, and two forget gates, f_t^L and f_t^R. At time instance t each node in the memory network is updated in the following manner,

$$i_t = \sigma(W_{hi}^L h_{t-1}^L + W_{hi}^R h_{t-1}^R + W_{ci}^L c_{t-1}^L + W_{ci}^R c_{t-1}^R), \tag{8}$$

$$f_t^L = \sigma(W_{hf_l}^L h_{t-1}^L + W_{hf_l}^R h_{t-1}^R + W_{cf_l}^L c_{t-1}^L + W_{cf_l}^R c_{t-1}^R), \tag{9}$$

$$f_t^R = \sigma(W_{hf_r}^L h_{t-1}^L + W_{hf_r}^R h_{t-1}^R + W_{cf_r}^L c_{t-1}^L + W_{cf_r}^R c_{t-1}^R), \tag{10}$$

$$\beta = W_{hc}^L h_{t-1}^L + W_{hc}^R h_{t-1}^R, \tag{11}$$

$$c_t^P = f_t^L \times c_{t-1}^L + f_t^R \times c_{t-1}^R + i_t \times tanh(\beta), \tag{12}$$

$$o_t = \sigma(W_{ho}^L h_{t-1}^L + W_{ho}^R h_{t-1}^R + W_{co}^P c_t^P), \tag{13}$$

$$h_t^P = o_t \times tanh(c_t^P), \tag{14}$$

where h_{t-1}^L, h_{t-1}^R, c_{t-1}^L and c_{t-1}^R are the hidden vector representations and cell states of the left and right children respectively. The relevant weight vectors, W, are represented with appropriate super and subscripts where the superscript represents the relevant child node, and the subscript represents the relevant gate and the vector the weight is attached to. The process is illustrated in Fig 4.

3.3. Relation to current state of the art

The major difference between the proposed approach and current state of the art methods such as [14, 21, 22] is the representation of the memory. In those approaches the memory is a composed of a single layer of memory units. The memory update mechanism in [14] and [21] can be written as,

$$\mathrm{M_t} = LSTM(\mathbf{m_t}, \mathrm{M_{t-1}}), \tag{15}$$

where m_t can be obtained using Eq. 4. In [22] the authors completely update the contents of the memory locations with respect to the output of Eq. 6. This process can be written as,

$$\mathrm{M_t} = \mathrm{M_{t-1}}(1 - (\mathbf{z_t} \otimes \mathbf{e_k})^T) + (h_t \otimes \mathbf{e_p})(\mathbf{z_t} \otimes \mathbf{e_k})^T, \tag{16}$$

where 1 is a matrix of ones and e_k and e_p are vectors of ones. $\otimes$ denotes the outer product which duplicates its left vector p or k times to form a matrix. In contrast we model our memory using a binary tree structure where it is updated in a bottom up fashion utilising Eq. 8 to 14.
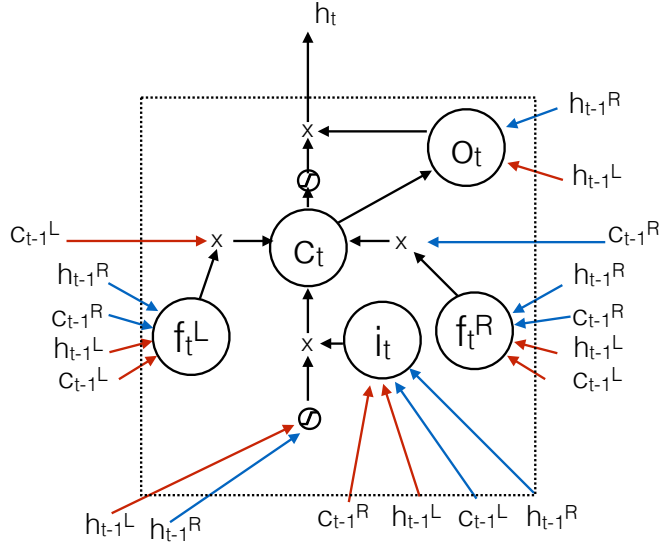
Figure 4: Tree memory cell architecture. f_t^L, f_t^R, o_t, i_t represents the left forget gate, right forget gate, output gate and input gate respectively. $\times$ represents multiplication

The main drawback of using approaches such as [14, 21, 22] with long term dependency modelling and big data sets is that, in most cases, the attention mechanism will generate small score values for all the examples as they are dissimilar to the input. Therefore with those approaches we are required to maintain an extremely large memory sequence as similar inputs only occur after long time intervals. However in recurrent models such as in LSTMs, when the sequence becomes too long the output becomes biased towards recent historic states [23], rather than considering the entire set of historic states equally. In contrast, we represent memory with a tree structure where we learn the logical coherence among neighbouring memory cells with different levels of abstraction.

4. Experimental results

We present the experimental results on two trajectory datasets: an aircraft trajectory database from the south east Queensland (SEQ) region of Australia; and a widely utilised pedestrian trajectory database. These datasets are specifically chosen to demonstrate the capability of the proposed model to handle varying dimensionalities and temporal relationships and present its real world applicability.

12

4.1. Experiment 1: Terminal Area Air Traffic Prediction

We obtain air traffic data from the south east Queensland (SEQ) region in Australia from 30-11-2014 to 30-11-2015. We use the real position reports recorded by the Australian Advanced Air Traffic System (TAAATS) used for air traffic management in Australia [54]. As a pre-processing step, the data is transformed into trajectories utilising the aircraft identification tags and the reported timing. Trajectories with less than 3 position reports are removed as they are too short for the trajectory modelling task. Finally, each trajectory is re-sampled to a length of 50 data points. When resampling, we interpolate the data points such that they have equal distance in the time domain. This is done to ensure sufficient data for training by up-sampling small trajectories. This gives us 260,735 trajectories. The aircraft trajectories are represented as 3 dimensional data streams where each point $\mathbf{x_t^i}$ of the input sequence $X^i = [\mathbf{x_1^i}, \mathbf{x_2^i}, \ldots, \mathbf{x_T^i}]$ can be represented as,

$$\mathbf{x_t^i} = \begin{pmatrix} x_t^i \\ y_t^i \\ z_t^i \end{pmatrix}, \tag{17}$$

where T is the number of time steps in the ith input sequence. In this experiment we observed the first 25 frames of the aircraft trajectory, and predicted the next 25 frames. For training we selected the first 182,515 (i.e 70%) trajectories chronologically. The remaining 78,220 trajectories were used for testing.

Based on the recommendations provided in [32, 55, 56], we measure the following three error metrics for the aircraft trajectory prediction experiment. The trajectory prediction errors are calculated for each observed radar track point in each input trajectory segment. Let n be the number of trajectories in the testing set, and we seek to predict the trajectory for the time period $t = T^{obs} + 1$ to T^{pred}, having observed the trajectory of the same aircraft from $t = 1$ to T^{obs}. Let the predicted course from North for the trajectory i at t^{th} time instance be denoted by $\hat{\theta}_t^i$. Then,

$$\Delta x_t^i = \hat{x_t^i} - x_t^i, \tag{18}$$

and,

$$\Delta y_t^i = \hat{y_t^i} - y_t^i. \tag{19}$$

Now we can define,

(a) Average altitude error vs length of memory module, p

(b) Average altitude error vs embedding dimension, k

(c) Average altitude error vs number of levels, from the top of the memory tree, in the memory read, l
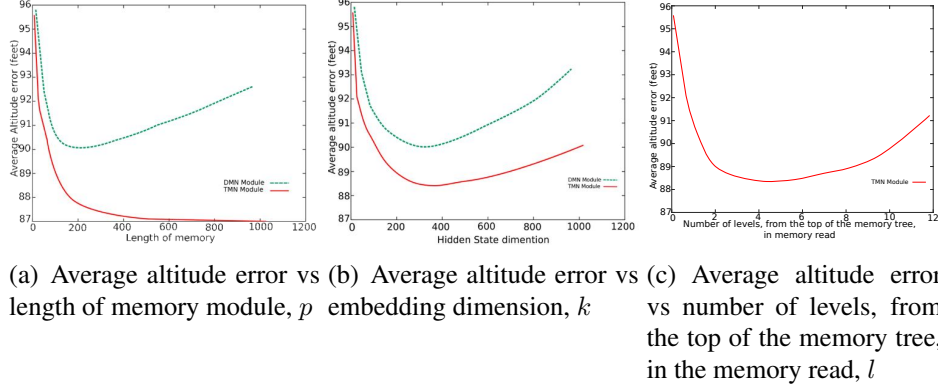
Figure 5: Parameter evaluation for length of memory module, p, embedding dimension, k, and the number of levels, from the top of the memory tree, in the memory read, l

1. *Average along track error (AE):*

$$AE = \frac{\sum_{i=1}^{n} \sum_{t=T^{obs}+1}^{T^{pred}} (\Delta x_t^i sin(\hat{\theta}_t^i) + \Delta y_t^i cos(\hat{\theta}_t^i))}{n(T^{pred} - (T^{obs} + 1))} \tag{20}$$

2. *Average cross track error (CE):*

$$CE = \frac{\sum_{i=1}^{n} \sum_{t=T^{obs}+1}^{T^{pred}} (\Delta x_t^i cos(\hat{\theta}_t^i) - \Delta y_t^i sin(\hat{\theta}_t^i))}{n(T^{pred} - (T^{obs} + 1))} \tag{21}$$

3. *Average altitude error (ALE):*

$$ALE = \frac{\sum_{i=1}^{n} \sqrt{\sum_{t=T^{obs}+1}^{T^{pred}} (\hat{z}_t^i - z_t^i)^2}}{n(T^{pred} - (T^{obs} + 1))} \tag{22}$$

As baseline models we implement the HMM approach (**HMM**) proposed in [32] and the Dynamic Memory Networks approach (**DMN**) given in [14]. For the **HMM** the required weather data is obtained from the Australian Bureau of Meteorology [57]. We observed wind speed and direction and temperature at one minute frequency. The data interpolation and parameter quantisation is performed in the same way as in [32].

4.1.1. Prediction under normal conditions

Hyper parameters, the length of the memory module, p, and the embedding dimension, k, of the proposed memory module (**TMN**) and **DMN** are evaluated experimentally. Fig. 5 (a) shows the variation in average altitude error against p for the **TMN** and for **DMN** modules in solid red and dashed green lines respectively. For the proposed **TMN**, as the error converges around $p = 500$, we set the value of p as 512. For **DMN** the plot shows that error decreases at first and it starts increasing again when the length of the memory exceeds 200 hidden units. This verifies our assertion that naive memory models with sequential LSTM architectures fail to model long term dependencies. As $p = 180$ gives the lowest altitude error, for the **DMN** model we p to 180. We evaluate the optimal embedding dimension, k, in a similar manner. Fig. 5 (b) shows the variation of average altitude error against k for the **TMN** and for **DMN** modules. For both modules $k = 300$ produces the smallest altitude error, and as such we set the embedding dimension to 300 units.

Finally, we evaluate the number of levels from the top of the memory tree in the memory read, l, of the proposed memory module (**TMN**). The evaluation results, shown in Fig. 5 (c), suggest that $l = 4$ produces optimal results. It is observed that the error is reduced until $l = 4$ before increasing again when the number of levels in the memory read operation exceeds 6 levels. This is due to the density of the extracted memory activation. Using the tree structure of the memory we capture the information in a hierarchical manner, where only vital information from the bottom layers is passed to the top layer. Therefore when the extracted matrix becomes too dense, the decoding function fails to extract pertinent information, and the performance degrades.

We train the TMN model using stochastic gradient descent (SGD) with momentum. Evaluation results are presented in Table 1.

Metric	HMM	DMN	TMN
AE	1.103	1.039	**1.020**
CE	1.042	1.056	**1.011**
ALE	147.801	92.039	**87.001**

Table 1: Quantitative results for aircraft trajectory prediction. In all the methods the forecast trajectories are of length 25 frames. The first row reports the along track error (AE), the second row shows cross track error (CE) and the final row shows the altitude error (ALE).

The results in Table 1 illustrate the ability of the proposed model to infer different modes of air traffic behaviour. We note that without explicitly mod-

elling the weather or neighbourhood, the proposed architecture is able to learn the salient aspects and long term dependencies which are necessary for modelling aircraft trajectories. The accuracy improvement from the **DMN** to the **TMN** model, demonstrates that flat memory architectures such as [14] fail to capture long term dependencies; where as our proposed hierarchical memory architecture is able to successfully learn those long term relationships. In Section 5.3 we perform an in depth analysis on the hidden state activations of the memory module which illustrates how the proposed multi-layer architecture generates future trajectories while encoding the necessary information from the history.

In Fig. 6 we show prediction results of the **HMM**, **DMN** and our approach (**TMN**) on the aircraft trajectory dataset. We show the trajectories on normalised scales for visual clarity as it better demonstrates the dispersion of the predictions from the ground truth. It should be noted that our model generates more accurate predictions across the highly varied scenarios depicted in the database (i.e take off, landing, cruising, etc). For instance in the 1st and 2nd rows we show how the same model adapts to takeoff and landing scenarios. In the last row of Fig. 6 we show some failure cases. The reason for such deviations from the ground truth were mostly due to sudden turns and movements. Even though these trajectories do not match the ground truth, the proposed method still outperforms the current state-of-the-art methods, and generates more realistic trajectories.

Considering the results presented in Tab. 1 and the visualisation in Fig. Fig. 6, we observe that the **HMM** approach performs poorly in the ALE metric compared to the CE and AE metrics. Our database contains variety of aircraft manoeuvres including take off, landing and cruising; and the landing and takeoff manoeuvres involve sudden changes in the aircraft motion which is hard to capture with the limited capacity of the **HMM**. This is reflected in higher error values for ALE metric to CE and AE, which only consider the dispersions along the latitude and longitude directions.

The **DMN** module improves upon the **HMM**'s performance using a sequential memory, which acts as a short term memory of the aircraft motion patterns. This captures the dynamics within a particular trajectory but cannot match dependencies across long time spans, such as how different flight schedules affect the trajectory patterns, and attend to them systematically to extract important information. This results in the **TMN** module achieving the best performance in all considered metrics.

(a) Latitude vs Longitude

(b) Latitude vs Altitude

(c) Latitude vs Longitude

(d) Latitude vs Altitude

(e) Latitude vs Longitude

(f) Latitude vs Altitude

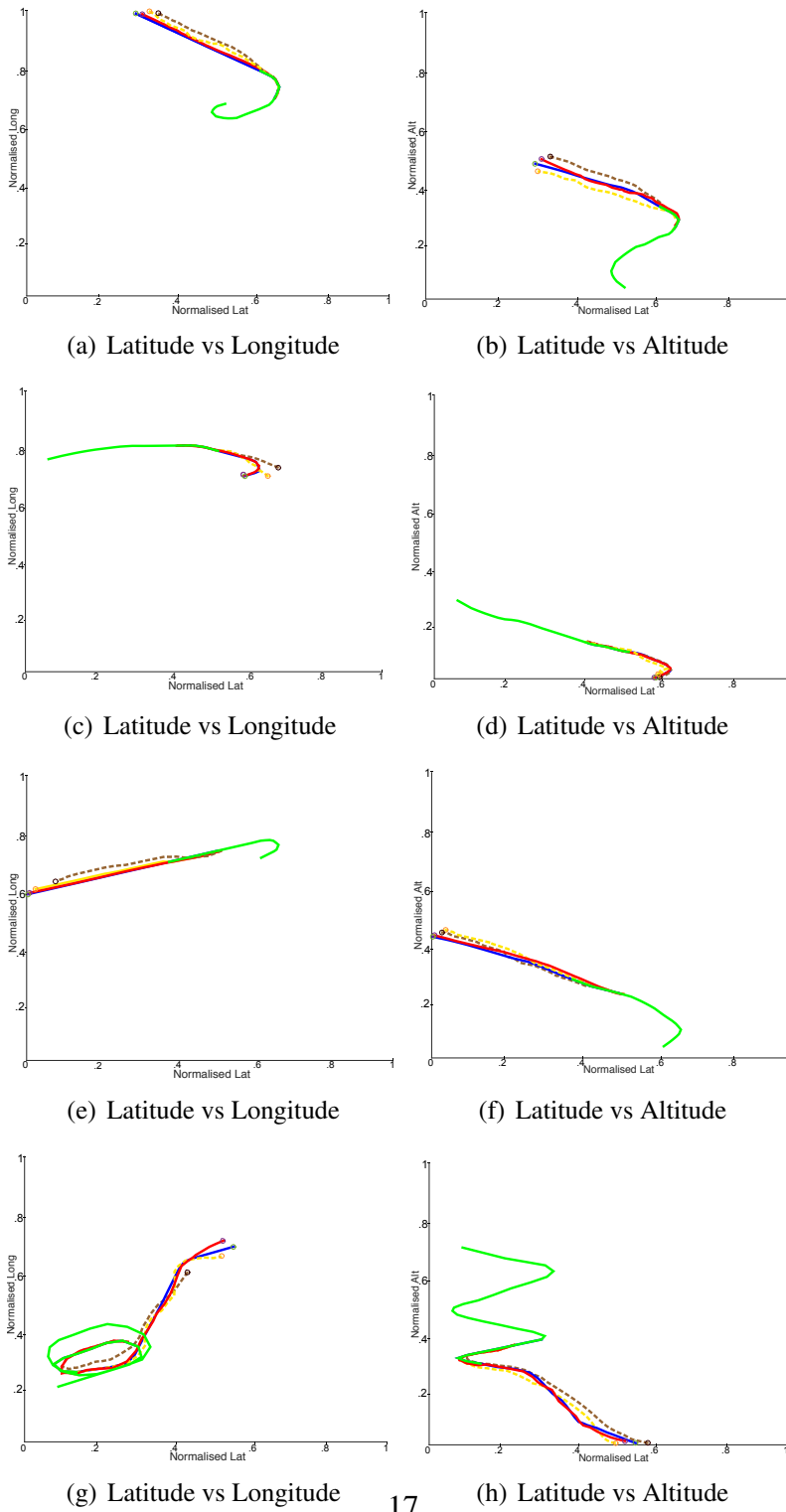(g) Latitude vs Longitude

(h) Latitude vs Altitude

17

Figure 6: Qualitative results under nominal conditions: Each row represents a particular example. Given (in green), Ground Truth (in Blue) and Predicted trajectories from the **TMN** model (in red), from the **DMN** model (in yellow), from the **HMM** model (in brown).

4.1.2. Handling different weather conditions

In order to verify the capability of the proposed model to understand and capture the effect of weather on aircraft trajectory prediction via historical trajectories alone, we conducted a separate experiment where the model is used to predict the air traffic on a stormy day. Severe storms affected South East Queensland on 28th of November 2015. We tested the model with the trajectories from 27th November to 29th November (1916 trajectories) as it allows a sufficient number of examples to initialise the memory module. These examples are not used for training the model. We compare the proposed model against the **HMM** [32], which explicitly incorporates weather information.

Metric	HMM	TMN
AE	1.689	**1.146**
CE	1.967	**1.513**
ALE	203.754	**88.397**

Table 2: Quantitative results for aircraft trajectory prediction under stormy conditions. For both methods the forecast trajectories are of length 25 frames. The first row reports the along track error (AE), the second row shows cross track error (CE) and the final row shows the altitude error (ALE).

Comparing Table. 2 with Table. 1, the accuracy of the **TMN** predictions are slightly reduced, but are still more accurate than [32], in which the error has increased dramatically indicating that the baseline model has not adapted well to the changed weather conditions.

Referring to the results presented in Fig. 7, it is evident that the non uniform nature of the air traffic in the stormy weather conditions is effectively modelled by the proposed approach. Even with the weather information, the **HMM** fails to effectively exploit this data and generates erroneous trajectories. The **TMN** model anticipates the uneven nature by looking at the recent history, while also mapping how correlated trajectories have behaved over the long term history.

Finally, we would like to emphasise that for the **TMN** model, even in storm conditions, the altitude error is within the +-100ft altimeter tolerances provided to private pilots [58].

4.2. Experiment 2: Pedestrian trajectory prediction

In this experiment we considered 3 months worth of trajectories from the Edinburgh Informatics Forum database [59]. We train our model on 60,000 trajectories and test on 12,000 trajectories. For all the models we observed the trajectory for

18

(a) Latitude vs Longitude

(b) Latitude vs Altitude

(c) Latitude vs Longitude

(d) Latitude vs Altitude

(e) Latitude vs Longitude

(f) Latitude vs Altitude

(g) Latitude vs Longitude
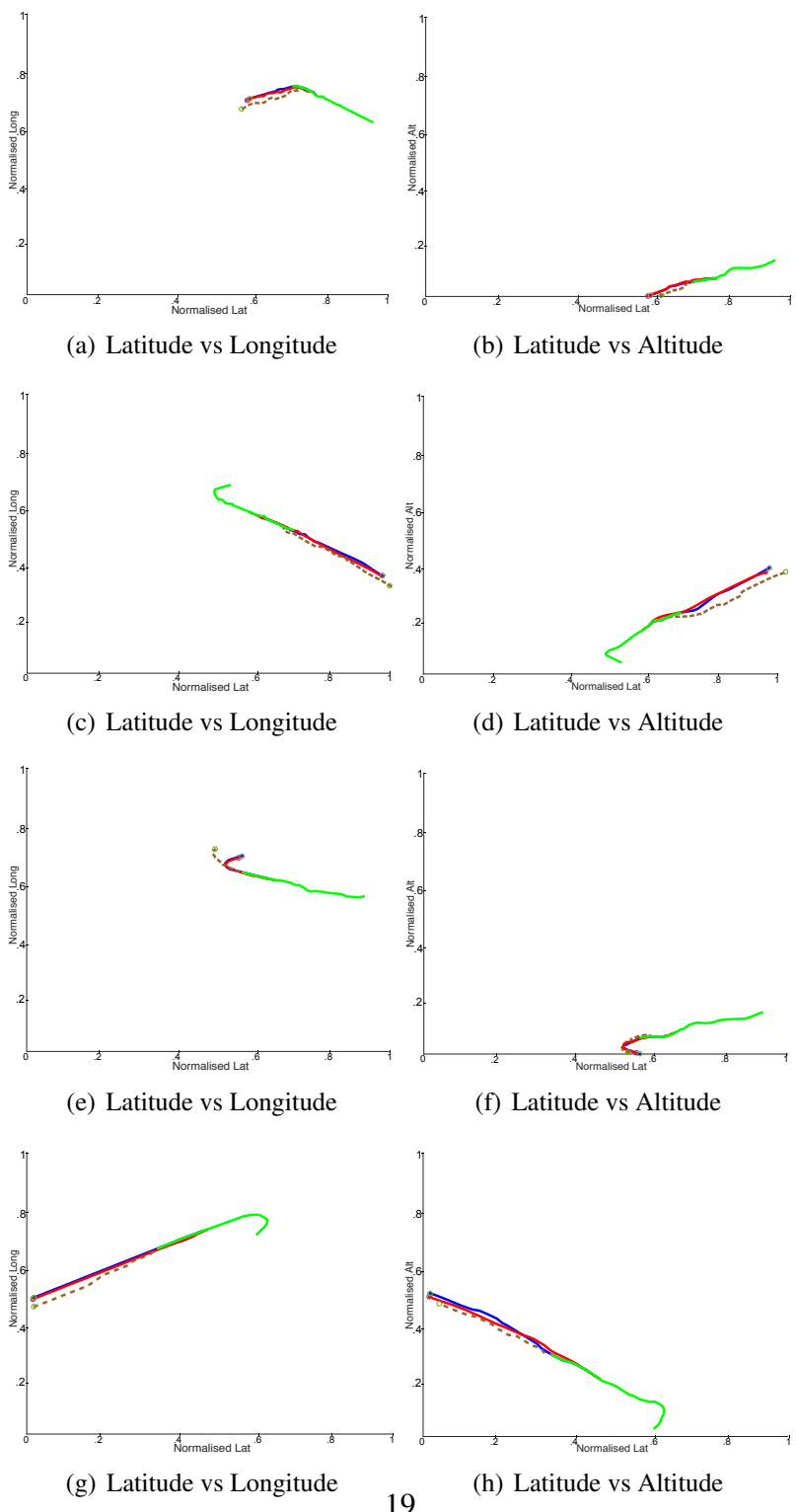
(h) Latitude vs Altitude

19

Figure 7: Qualitative results under storm conditions: Each row represents a particular example. Given (in green), Ground Truth (in Blue) and Predicted trajectories from **TMN** model (in red), from **HMM** model (in brown).

30 frames and predicted the trajectory for the next 30 frames. In this experiment the trajectories are represented as 2 dimensional data where each point $\mathbf{x_t^i}$ of the input sequence $X^i = [\mathbf{x_1^i}, \mathbf{x_2^i}, \ldots, \mathbf{x_T^i}]$ is represented as,

$$\mathbf{x_t^i} = \begin{pmatrix} x_t^i \\ y_t^i \end{pmatrix}. \tag{23}$$

For comparison we implemented Soft+hard wired attention (**SH-Atn**) model from [48], the Social LSTM (**So-LSTM**) model given in [47] and Dynamic Memory Networks (**DMN**) given in [14]. For the **So-LSTM** model, a local neighbourhood of size 32px was considered and the embedding dimension of all the LSTMs are set to 180 as recommended in [47]. For [48] we considered a neighbourhood size of 10 in the left, right and front directions and an embedding size of 300 hidden units. For **DMN** and **TMN** models we use the same experimental settings given the in previous experiment. Similar to [48, 47] we report prediction accuracy with the following 3 error metrics. We are predicting a trajectory for the period from $t = T^{obs} + 1$ to T^{pred} while observing the same trajectory from $t = 1$ to $T^{obs} + 1$. Let n be the number of trajectories in the testing set, $\hat{X}_t^i$ be the predicted position for the trajectory i at the tth time instance, and X_t^i be the respective observed positions then:

1. *Average displacement error (ADE):*

$$ADE = \frac{\sum_{i=1}^{n} \sum_{t=T^{obs}+1}^{T^{pred}} (\hat{X}_t^i - X_t^i)^2}{n(T^{pred} - (T^{obs} + 1))}. \tag{24}$$

2. *Final displacement error (FDE) :*

$$FDE = \frac{\sum_{i=1}^{n} \sqrt{(\hat{X}_{T^{pred}}^i - X_{T^{pred}}^i)^2}}{n}. \tag{25}$$

3. *Average non-linear displacement error (n-ADE):* The average displacement error for the non-linear regions of the trajectory.

$$n - ADE = \frac{\sum_{i=1}^{n} \sum_{t=T^{obs}+1}^{T^{pred}} I(\hat{X}_t^i)(\hat{X}_t^i - X_t^i)^2}{\sum_{i=1}^{n} \sum_{t=T^{obs}+1}^{T^{pred}} I(\hat{X}_t^i)}, \tag{26}$$

20

where,

$$I(\hat{X}_t^i) = \begin{cases} 1 & \text{if } \dfrac{d^2 y_t^i}{d(x_t^i)^2} \neq 0. \\ 0 & o.w \end{cases} \tag{27}$$

Metric	SH-Atn	So-LSTM	DMN	TMN
ADE	1.066	1.843	1.798	**1.051**
FDE	1.551	2.421	2.276	**1.398**
n-ADE	1.021	1.988	1.456	**0.987**

Table 3: Quantitative results for pedestrian trajectory prediction. In all the methods the forecast trajectories are of length 30 frames. The first row represents the average displacement error (ADE), the second row shows the final displacement error (FDE) and the third row shows the average non-linear displacement error (n-ADE).

As shown in Table 3, the proposed model outperforms the **SH-Atn** model, **So-LSTM** model and **DMN** in all three error metrics. The dataset is considered quite challenging as there are multiple source and sink locations, different crowd motion patterns are present, and motion paths are heavily crowded.

The **So-LSTM** model has the lowest accuracy, as its attention mechanism is limited to the immediately preceding state of the neighbourhood. The **DMN** model considers the short term history of the entire trajectory, leading to improved performance compared to **So-LSTM**. By incorporating local neighbourhood history, **SH-Atn** is able to further improve on performance. However, despite not explicitly modelling the neighbourhood as done by the **SH-Atn** and **So-LSTM** models, the proposed approach is able to outperform these state-of-the-art techniques. The **TMN** approach extends the notion of neighbourhood history to consider longer temporal dependencies. It not only considers the short term environment context, where temporally adjacent trajectories are, but also considers how similar trajectories have behaved over the long term history. This is further demonstrated by the results presented in Fig. 8.

Fig. 8 shows prediction results for the **SH-Atn** model, **DMN** model and our model **TMN** on the EIF trajectory dataset. From the examples shown it is evident how different modes of human motion are captured and represented through the proposed memory module. For example in Fig. 8 (g) and Fig. 8 (k) the pedestrians exhibit a sudden change in motion which all baseline models fail to capture. But the proposed model has successfully anticipated that motion through recalling similar historic behaviour.
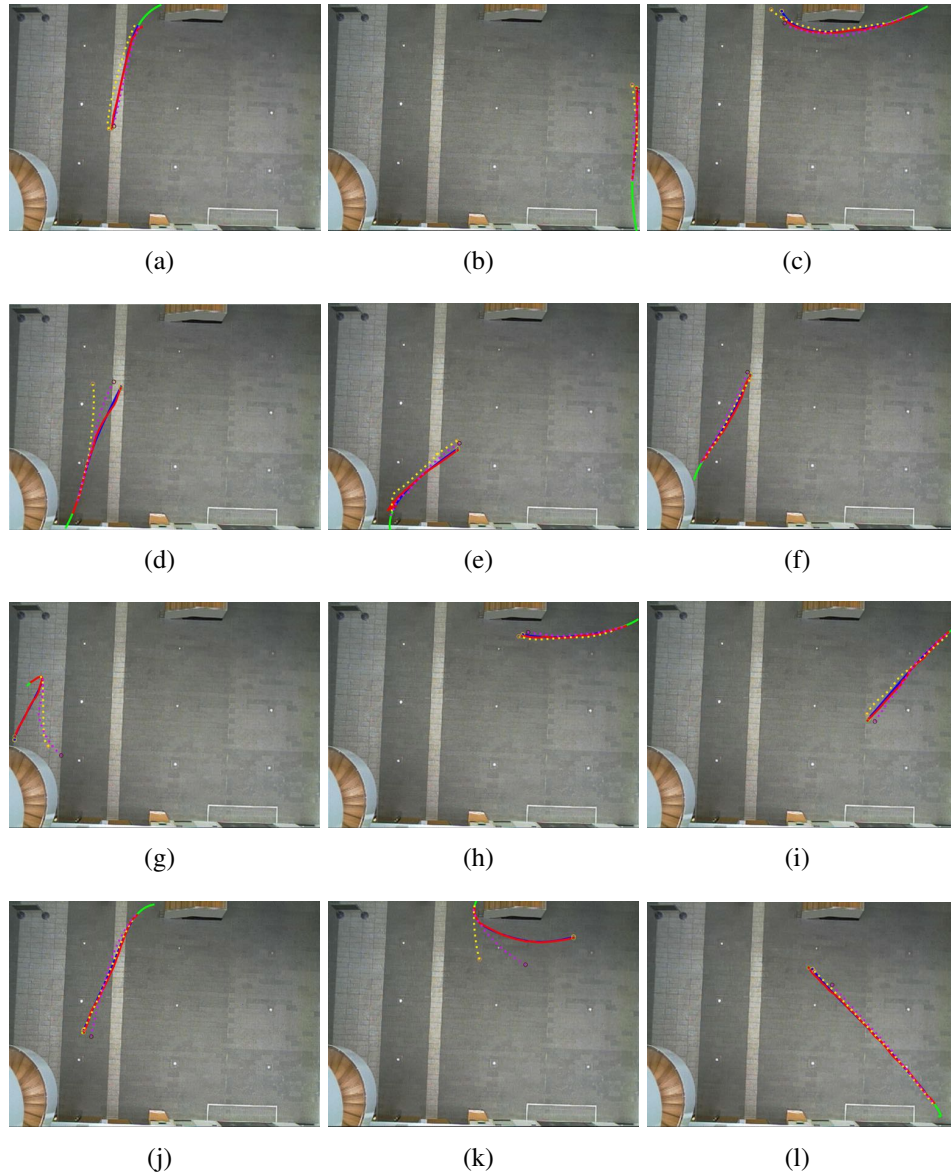
Figure 8: Qualitative results: Given (in green), Ground Truth (in Blue) and Predicted trajectories from **TMN** model (in red), from **DMN** model (in yellow), from **SH-Atn** model (in purple).

Fig. 9 shows the distribution of activations from the first layer of the proposed TMN module, for the first data point of the pedestrian trajectory given in Fig. 8
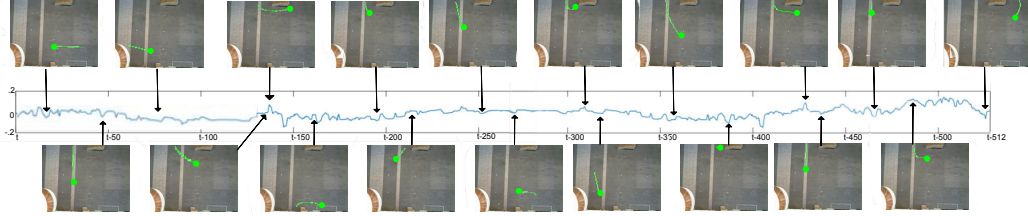
Figure 9: Distribution of memory activations from the first layer of the **TMN** module for the first data point of Fig. 8 (k). This layer contains 512 memory which are denoted t to $t - 512$ indicating the history that has been observed. For different peaks and valleys of the memory activations we also show what the memory has seen at those particular time steps. The model generates higher activations for trajectories that change the heading direction and perform a turn as in Fig. 8 (k), and activations closer to zero for cases where the pedestrian in demonstrating different behaviour. We effectively propagate this information from the first layer of the memory to the top most layer via combining the salient information in a hierarchical manner.

(k). As $p = 512$, there exist 512 memory slots in this layer. We denote the current time as t, hence the memory slots range from t to $t - 512$ in the history. For different peaks and valleys in the memory activations, we show what the model has seen at that particular time step.

The **TMN** provides higher responses for recent events as well as for similar trajectory patterns in the long term history. Considering both spatial locations as well as the velocity encoded by the spatial dispersion between the consecutive points, the model is able to anticipate that the pedestrian is more likely to change their heading, indicated by higher activations to similar trajectories that reside within the entire history captured by the memory module (see the activation peeks between $t - 100$ to $t - 200$ and $t - 300$ to $t - 500$).

The flat memory structure of the **DMN** doesn't have the ability to capture such long term dependencies due to the short term history dominance issues inherent with sequential LSTM architectures, which we further discuss and demonstrate in Sec. 5.3. With $p = 180$, the memory of the **DMN** has seen similar pedestrian behaviour to that in Figure 9 (k), but cannot identify the importance of those examples as the short term history dominates the output. This clearly verifies the importance of efficiently modelling dependencies with a hierarchical structure. It is not sufficient to just have a large (i.e. long history) memory module, the module also needs to effectively propagate relevant historic examples to the output module to generate better predictions.

23

5. Discussion

5.1. Flexibility of the TMN framework

We selected the two problem domains, aircraft trajectories (see Section 4.1) and pedestrian trajectories (see Section 4.2), to highlight the flexibility of the **TMN** model. Even though both domains consist of trajectories, the structure and dynamics of the domains show vast differences.

Aircraft trajectories capture 3 dimensional motion patterns, with rigid structure, following a particular flight schedule throughout the year. We do not offer any information to the model on to the specifics of the flight dynamics or manoeuvres. The model has to learn those characteristics directly from data by comparing and contrasting the temporal evolution of the large number of trajectories. Furthermore, there exist different trajectory patterns in take off, landing and cruising of flights; and between different aircraft types including commercial airliners, helicopters, surveillance flights, etc. We do not filter any of those categories from the data, and feed them all together to the model. The **TMN** model successfully understands these specifics through querying from the long term history.

In contrast to the structured pattern of the aircraft trajectories, pedestrian trajectories are highly unstructured. The model has to identify that pedestrians vary their velocity and heading directions more frequently and rapidly, often in response to other very recent observations, compared to the aircraft trajectories.

We do not provide any supervision to our model or change the structure of the network between the two experiments. Based on the experimental results in Tab. 1 and Tab. 3 the proposed model successfully identifies those differences and demonstrates flexibility in adaptation to the different conditions.

To further demonstrate the flexibility of the proposed model we use different trajectory lengths for the two experiments.

In the case of aircraft trajectories, the TAAATS samples aircraft position at a rate of 1 Hz. As we are using data only from the SEQ region of Australia, this gives us relatively short trajectory lengths, but with higher variability between the data points. In contrast pedestrian trajectories are sampled at 25 fps giving us more lengthy trajectories and less variability between the data points. Due to these differences in capture rate and the typical length of trajectories in each dataset, we use different length trajectories when predicting future behaviour: using 25 frames of data to predict the next 25 frames for aircraft trajectories, and using 30 frames to predict the next 30 for pedestrian motion prediction.

Results presented in Sections 4.1 and 4.2 show how the proposed **TMN** model is able to adapt to these changes without explicit supervision.

5.2. Hardware and Implementation Details

The TMN module doesn't require any special hardware such as GPUs to run and has 8.1M trainable parameters. We ran the test set of experiment 1 on a single core of an Intel Xeon E5-2680 2.50GHz CPU and the TMN algorithm was able to generate 1000 predicted trajectories with 50, 3 dimensional data points in each trajectory (i.e. using 25 observations to predict the next 25 data points) in 12.20 seconds.

In addition, we measured the time required to generate 1000 predicted trajectories for different lengths of the memory module, p, and different sequence lengths T. Results are presented in Fig. 10 along with the respective evaluations for **DMN** module. The runtime grows approximately logarithmically against the memory size, as multiple S-LSTM layers are added to accommodate the increasing size of the memory component. There is an additional cost for using the proposed **TMN** over the **DMN** for a given memory size, however this cost is roughly consistent and does not vary greatly with the size of the memory. The time efficiency against sequence length exhibits a linear relationship, as sequence length only affects the encoding and decoding of the trajectories. We held the memory length p=512 constant for both modules in this experiment. The DMN module exhibits a similar distribution of runtimes and is slightly more time efficient. The efficiency gains are largely due to the update mechanism, as the sequential update is much simpler than the hierarchical update.

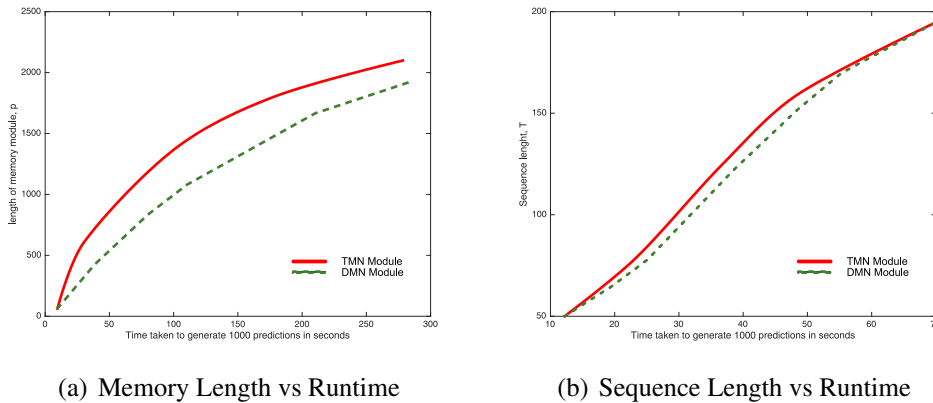(a) Memory Length vs Runtime   (b) Sequence Length vs Runtime

Figure 10: Evaluation of run times for different memory sizes and sequence lengths

The implementation of the TMN module presented in this paper is completed

using Keras [60] with the Theano [61] backend, and only accepts equal length trajectory proportions for observation and prediction. Yet this is only a limitation in our implementation of the TMN, and the proposed algorithm is flexible and able to handle variable proportions in observed and predicted trajectories.

5.3. Analysis of Memory Activations

In this section we analyse the memory activations of the **TMN** and **DMN** [14] models for various examples of the test set from Experiment 2 (see Section 4.2, Table 3 and Figure 8) in order to demonstrate that the sequential LSTM structure is biased towards recent history, and illustrate how the proposed model overcomes this via the hierarchical structure of the memory module.

In particular, we aim to show that the DMN model and the first layer of the proposed TMN approach have activations based heavily on the recent inputs to the model; while the last layer of the proposed approach has activations that are driven by the input itself rather than the recent history, due to it's ability to better capture the long term dependencies. We conduct this investigation with the pedestrian dataset as it is simpler to visualise.

5.3.1. Correlated activations from the first layer

We randomly select a memory cell from the first layer of the memory, and analysed the activations of the hidden states of that particular cell. Fig. 11 shows the results of our analysis. We searched our test set for common activation patterns. The first column of Fig. 11 shows the memory activations, with the most highly correlated memory patterns shown in red, green and blue; and the remainder of the activations shown in grey. The second column shows the input to the model (in green) and the predicted sequence (in blue). The previous 10 trajectories that are fed to the memory are shown in the third column. From black to white we have colour coded the most recent trajectory to the oldest trajectory. We expect the first layer of the memory module to generate similar activation patterns when there exists a similar set of historical trajectories, and if there exists a similarity between those historical trajectories and the input.

5.3.2. Correlated activations from the last layer

Fig. 12 illustrates the activations of the final memory cell (i.e last layer). Column descriptions are identical to the that of Fig. 11. The second and third columns of Fig. 12 provide visual evidence that the proposed memory module has successfully learnt relationships among input trajectories. If our memory module has enough capacity and if it is capturing long term dependencies, the final layer

should generate similar activations for similar input trajectory patterns, rather than being completely reliant on the current short term context. That is evident with the similarity shown in Fig. 12 where we observe similar activations (i.e column 1) for similar inputs patterns (i.e column 2). Importantly, we note that despite the similar activations, the recent history (i.e. column 3) differs significantly between cases; unlike in Fig. 11.

5.3.3. *Activations from the DMN [14] memory module*

In Fig. 13 we visualise the correlated activations from the DMN memory model shown in Fig. 3 (a). The column labels are identical to that of Fig. 11. We compare the activations in Fig. 12 to those in Fig. 13. When observing Fig 13 column 3 it is evident that hidden state activations are dominated by the most recent inputs to the memory, and the long term dependencies are of little importance. This is noted to be an inherent problem with sequential LSTM architectures [23]. Therefore regardless of the input to the model (shown in Fig 13 column 2) the memory module is generating similar activations and is only considering the short term context. Hence the prediction error is high. Furthermore we would like to highlight the similarity between Fig 13 and the first layer of the proposed memory module (shown in Fig. 11).

To further illustrate the limitations of the DMN model, we cluster the input trajectories and from one particular cluster we extract out the input trajectories with the highest correlation (shown in Fig. 14) within that cluster. Given that we have very similar inputs, we expect the DMN module to generate similar activations, however we observe that the DMN generates vastly different activations. In order to highlight the differences among memory activations we randomly selected 3 hidden units within the memory and illustrate their temporal evolutions (see the coloured lines in Fig. 14 column 1). From Fig. 14, we can see that the short term history across the examples is varied (see Fig. 14 column 3), and the memory module generates vastly different activations for each, ignoring the given input.

Considering Fig. 13 and Fig. 14 together, we can see that the activations are driven by the short term history. Similar short term histories with different inputs lead to similar activations (shown in Fig. 13 column 1); and different short term histories with similar inputs result in vastly different activations (shown in Fig. 14 column 1). This is in contrast to the proposed approach, where as shown in Fig. 12, at higher levels of the hierarchy memory activations are driven by the input.

6. Conclusion

The proposed tree memory network (TMN) model is a generalised architecture for modelling long term and short term relationships, which can be applied directly for any sequence-to-sequence mapping task. Through the evaluation results we demonstrated that our proposed memory architecture is able to outperform all considered baselines, and we provide visual evidence on the power of TMN which is able to capture both long term and short term relationships via an efficient tree structure. We have demonstrated our approach on two different trajectory prediction applications. The varied nature of these problems demonstrates how the proposed TMN model can be directly applied to any sequence-to-sequence prediction problem where modelling long term relationships is necessary. In future work we will be exploring the applications of TMN as an encoding mechanism for large scale multi-modal inputs such as videos (i.e. a sequence of images), where the encoded vector representation of the memory can be utilised to generate a sparse representation of the entire video sequence with its temporal relationships.

References

[1] E. ShafieiBavani, M. Ebrahimi, R. Wong, F. Chen, An efficient approach for multi-sentence compression, in: JMLR: Workshop and Conference Proceedings, 2016.

[2] K. Judah, A. P. Fern, T. G. Dietterich, et al., Active lmitation learning: formal and practical reductions to iid learning, The Journal of Machine Learning Research (JMLR) 15 (1) (2014) 3925–3963.

[3] J. Lafferty, A. McCallum, F. Pereira, et al., Conditional random fields: Probabilistic models for segmenting and labeling sequence data, in: International Conference on Machine Learning (ICML), Vol. 1, 2001, pp. 282–289.

[4] T. Dietterich, Machine learning for sequential data: A review, Structural, syntactic, and statistical pattern recognition (2002) 227–246.

[5] W. Bao, J. Yue, Y. Rao, A deep learning framework for financial time series using stacked autoencoders and long-short term memory, PloS one, 2017 12 (7).

[6] J. Xu, D. Chen, X. Qiu, X. Huang, Cached long short-term memory neural networks for document-level sentiment classification, conference on Empirical Methods in Natural Language Processing (EMNLP), 2016.

[7] S. Yi, H. Li, X. Wang, Understanding pedestrian behaviors from stationary crowd groups, in: IEEE International Conference on Computer Vision and Pattern Recognition (CVPR), 2015.

[8] C. Chen, A. Seff, A. Kornhauser, J. Xiao, Deepdriving: Learning affordance for direct perception in autonomous driving, in: IEEE International Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 2722–2730.

[9] H. Liu, T. Taniguchi, Y. Tanaka, K. Takenaka, T. Bando, Visualization of driving behavior based on hidden feature extraction by using deep learning, IEEE Transactions on Intelligent Transportation Systems, 2017.

[10] T. Fernando, S. Denman, S. Sridharan, C. Fookes, Going deeper: Autonomous steering with neural memory networks, in: The IEEE International Conference on Computer Vision (ICCV), 2017.

[11] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, H. Lee, Generative adversarial text to image synthesis, in: International Conference on Machine Learning (ICML), Vol. 3, 2016.

[12] R. E. Banchs, A principled approach to context-aware machine translation, European Chapter of the Association for Computational Linguistics (EACL) (2014) 70–74.

[13] J. Weston, S. Chopra, A. Bordes, Memory networks, International Conference on Learning Representations (ICLR), 2015.

[14] A. Kumar, O. Irsoy, J. Su, J. Bradbury, R. English, B. Pierce, P. Ondruska, I. Gulrajani, R. Socher, Ask me anything: Dynamic memory networks for natural language processing, International Conference on Machine Learning (ICML), 2016.

[15] M. Malinowski, M. Fritz, A multi-world approach to question answering about real-world scenes based on uncertain input, in: Advances in Neural Information Processing Systems, 2014, pp. 1682–1690.

[16] A. Joulin, T. Mikolov, Inferring algorithmic patterns with stack-augmented recurrent nets, in: Advances in Neural Information Processing Systems, 2015, pp. 190–198.

[17] T. Fernando, S. Denman, S. Sridharan, C. Fookes, Task specific visual saliency prediction with memory augmented conditional generative adversarial networks, Applications of Computer Vision (WACV), 2018 IEEE Winter Conference on.

[18] T. Fernando, S. Denman, S. Sridharan, C. Fookes, Tracking by prediction: A deep generative model for mutli-person localisation and tracking, Applications of Computer Vision (WACV), 2018 IEEE Winter Conference on.

[19] Ł. Kaiser, I. Sutskever, Neural gpus learn algorithms, International conference on learning representations (ICLR), 2016.

[20] X. Chen, C. L. Zitnick, A recurrent visual representation for image caption generation, IEEE international conference on computer vision and pattern recognition (CVPR), 2015.

[21] A. Neelakantan, Q. V. Le, I. Sutskever, Neural programmer: Inducing latent programs with gradient descent, International conference on learning representations (ICLR), 2015.

[22] T. Munkhdalai, H. Yu, Neural semantic encoders, European Chapter of the Association for Computational Linguistics (EACL), 2017.

[23] Q. Chen, X. Zhu, Z. Ling, S. Wei, H. Jiang, Enhancing and combining sequential and tree lstm for natural language inference, Annual meeting of the association for computational linguistics (ACL), 2017.

[24] X. Liang, X. Shen, J. Feng, L. Lin, S. Yan, Semantic object parsing with graph lstm, in: European Conference on Computer Vision (ECCV), 2016, pp. 125–143.

[25] X. Liang, L. Lin, X. Shen, J. Feng, S. Yan, E. P. Xing, Interpretable structure-evolving lstm, IEEE International Conference on Computer Vision and Pattern Recognition (CVPR), 2017.

[26] X. Zhu, P. Sobhani, H. Guo, Long short-term memory over recursive structures, International conference on machine learning (ICML), 2015.

[27] M. Prandini, J. Hu, J. Lygeros, S. Sastry, A probabilistic approach to aircraft conflict detection, IEEE Transactions on Intelligent Transportation Systems 1 (4) (2000) 199–220.

[28] R. A. Paielli, H. Erzberger, Conflict probability estimation for free flight, Journal of Guidance, Control, and Dynamics 20 (3) (1997) 588–596.

[29] P. P. Choi, M. Hebert, Learning and predicting moving object trajectory: a piecewise trajectory segment approach, Robotics Institute (2006) 337.

[30] A. de Leege, M. Van Paassen, M. Mulder, A machine learning approach to trajectory prediction, AIAA Guidance, Navigation, and Control Conference, 2013.

[31] L. F. Winder, Hazard avoidance alerting with markov decision processes, Ph.D. thesis, Massachusetts Institute of Technology, 2004.

[32] S. Ayhan, H. Samet, Aircraft trajectory prediction made easy with predictive analytics, in: International Conference on Knowledge Discovery and Data Mining (KDD), 2016, pp. 21–30.

[33] S. Mondoloni, D. Liang, Improving trajectory forecasting through adaptive filtering technique, in: Proceedings of 5th USA-Europe ATM Seminar, 2003.

[34] R. Cole, S. Green, M. Jardin, B. Schwartz, S. Benjamin, Wind prediction accuracy for air traffic management decision support tools, in: USA/Europe Air Traffic Management Research and Development Seminar, 2000.

[35] C. Rekkas, C. Lefas, N. Krikelis, Three-dimensional tracking using on-board measurements, IEEE transactions on aerospace and electronic systems 27 (4) (1991) 617–624.

[36] D. Delahaye, Wind field update using radar track data, Ph.D. thesis, Masters thesis, Ecole Nationale de lAviation Civile, 1992.

[37] W. Hollister, E. Bradford, J. Welch, Using aircraft radar tracks to estimate wind aloft, The Lincoln Laboratory Journal 2 (3) (1989) 555–565.

[38] D. Helbing, P. Molnár, Social force model for pedestrian dynamics, Phys. Rev. E 51 (1995) 4282–4286.

[39] H. S. Koppula, A. Saxena, Anticipating human activities using object affordances for reactive robotic response., in: International Conference on Intelligent Robots and Systems (IROS), 2013, p. 2071.

[40] S. Pellegrini, A. Ess, L. J. V. Gool, Improving data association by joint modeling of pedestrian trajectories and groupings., in: European Conference on Computer Vision (ECCV), 2010, pp. 452–465.

[41] K. Yamaguchi, A. C. Berg, L. E. Ortiz, T. L. Berg, Who are you with and where are you going?, in: IEEE International Conference on Computer Vision and Pattern Recognition (CVPR), no. 1345-1352, 2011.

[42] J. Xu, S. Denman, C. B. Fookes, S. Sridharan, Unusual scene detection using distributed behaviour model and sparse representation, Advanced Video and Signal Based Surveillance (AVSS) (2012) 48 – 53.

[43] B. Zhou, X. Tang, X. Wang, Learning collective crowd behaviors with dynamic pedestrian-agents., Journal of Computer Vision 111 (2015) 50–68.

[44] X. Wang, K. T. Ma, G. W. Ng, W. E. L. Grimson, Trajectory analysis and semantic region modeling using a nonparametric bayesian model., in: IEEE International Conference on Computer Vision and Pattern Recognition (CVPR), 2008, pp. 1–8.

[45] T. M. Hospedales, S. Gong, T. Xiang, A markov clustering topic model for mining behaviour in video., in: International Conference on Computer Vision (ICCV), 2009, pp. 1165–1172.

[46] R. Emonet, J. Varadarajan, J.-M. Odobez, Extracting and locating temporal motifs in video scenes using a hierarchical non parametric bayesian model., in: IEEE International Conference on Computer Vision and Pattern Recognition (CVPR), 2011, pp. 3233–3240.

[47] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, S. Savarese, Social lstm: Human trajectory prediction in crowded spaces, IEEE International Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[48] T. Fernando, S. Denman, S. Sridharan, C. Fookes, Soft+ hardwired attention: An lstm framework for human trajectory prediction and abnormal event detection, Under review as a journal paper in Elsevier Neural Networks Journal.

[49] T. Fernando, X. Wei, C. Fookes, S. Sridharan, P. Lucey, Discovering methods of scoring in soccer using tracking data, KDD workshop on Large-Scale Sports Analytics, 2015.

[50] I. V. Serban, A. Sordoni, R. Lowe, L. Charlin, J. Pineau, A. Courville, Y. Bengio, A hierarchical latent variable encoder-decoder model for generating dialogues, AAAI Conference on Artificial Intelligence (AAAI), 2017.

[51] J. Li, M.-T. Luong, D. Jurafsky, A hierarchical neural autoencoder for paragraphs and documents, Annual meeting of the association for computational linguistics (ACL), 2015.

[52] H. Gammulle, S. Denman, S. Sridharan, C. Fookes, Two stream lstm: A deep fusion framework for human action recognition, in: Applications of Computer Vision (WACV), 2017 IEEE Winter Conference on, IEEE, 2017, pp. 177–186.

[53] T. Munkhdalai, H. Yu, Neural tree indexers for text understanding, European Chapter of the Association for Computational Linguistics (EACL), 2017.

[54] A. McFadyen, T. Martin, Terminal airspace modelling for unmanned aircraft systems integration, in: International Conference on Unmanned Aircraft Systems (ICUAS), 2016, pp. 789–794.

[55] C. Gong, D. McNally, A methodology for automated trajectory prediction analysis, in: AIAA Guidance, Navigation, and Control Conference, 2004, pp. 16–19.

[56] M. M. Paglione, R. D. Oaks, Implementation and metrics for a trajectory prediction validation methodology, in: AIAA Guidance, Navigation, and Control Conference, 2007.

[57] BOM, Australian bureau of meteorology,.
URL http://www.bom.gov.au/climate/data/stations/

[58] AIP, Altimeter checks and flight tolerances, airservices, australia, Aeronautical Information Package (2017) 39.

[59] B. Majecka, Statistical models of pedestrian behaviour in the forum, MSc Dissertation, School of Informatics, University of Edinburgh, 2009.

[60] F. Chollet, Keras, URL http://keras. io, 2017.

[61] J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, Y. Bengio, Theano: A cpu and gpu math compiler in python, in: Proceedings of 9th Python in Science Conference, 2010, pp. 1–7.
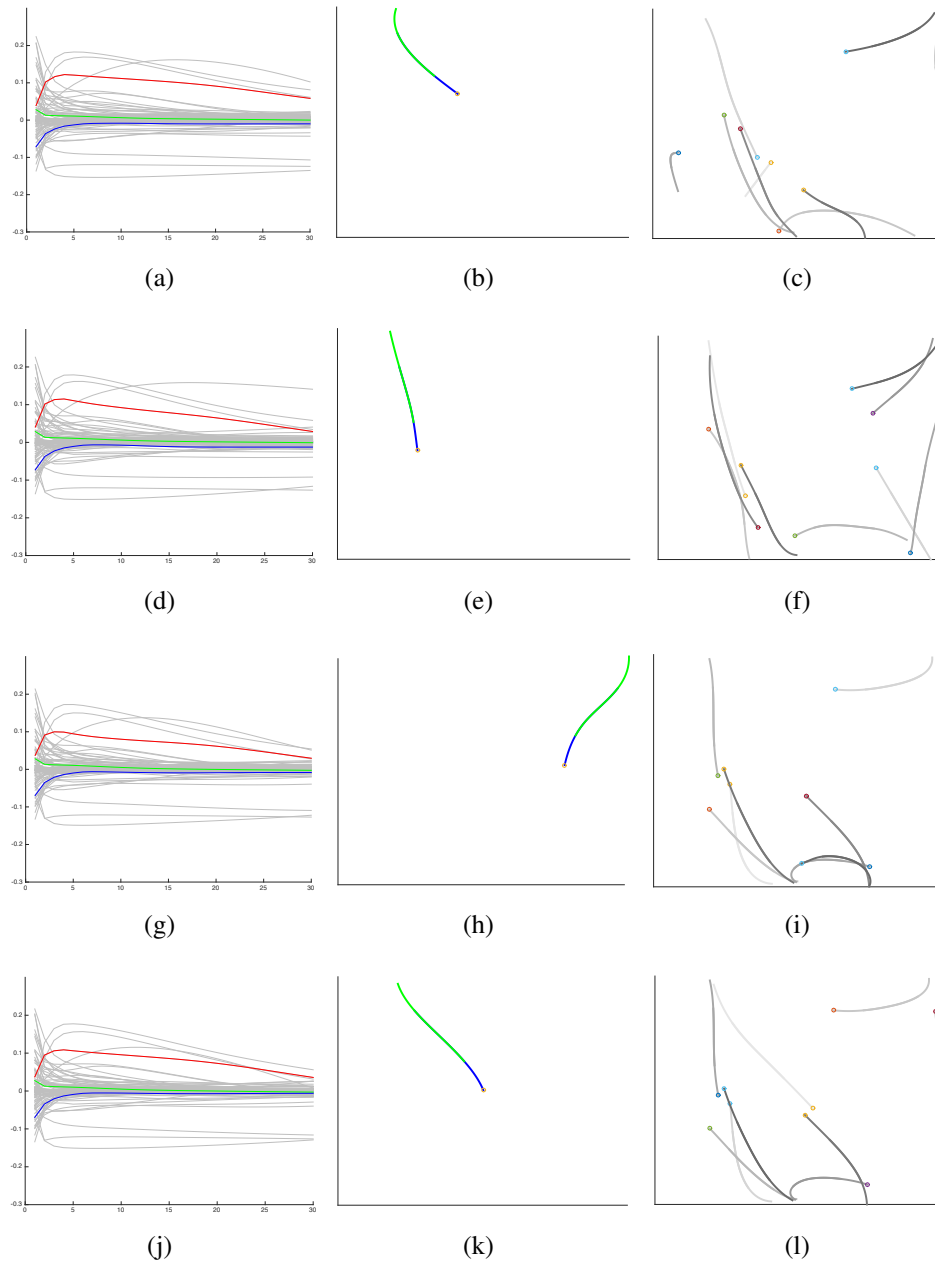
Figure 11: Pedestrian trajectories: Correlated activations from the first layer of the memory. First column: Highest correlated Memory activations for the pattern selected (in colours) and the rest of the activations (in grey) over time. Second column: The input (observed (in green) and predicted (in blue)) to the model at that time step. Third column: Previous 10 trajectories that reside in the memory. Black to white represents most recent to oldest.
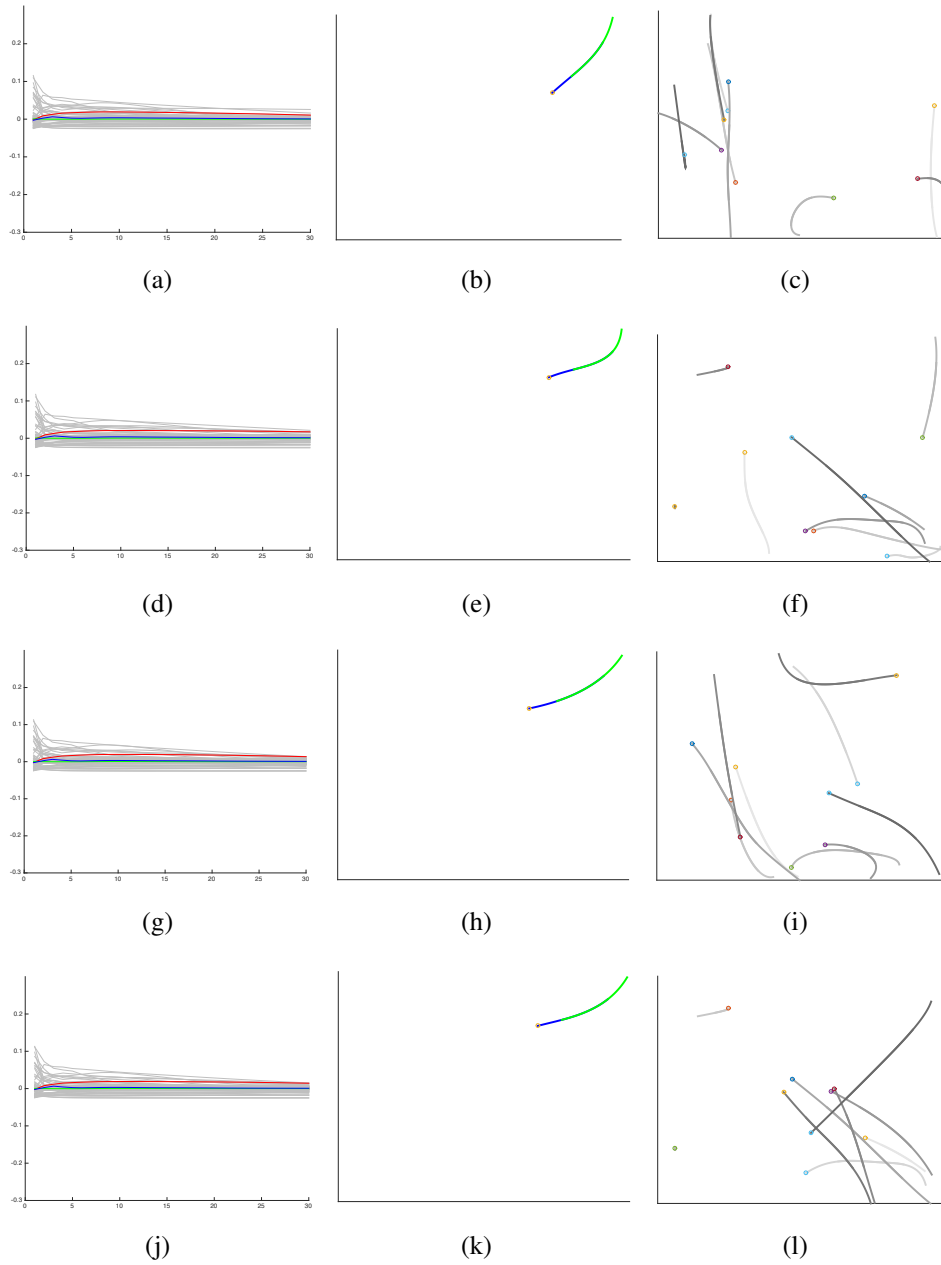
Figure 12: Pedestrian trajectories: Correlated activations from the last layer of the memory. First column: Highest correlated Memory activations for the pattern selected (in colours) and the rest of the activations (in grey) over time. Second column: The input (observed (in green) and predicted (in blue)) to the model at that time step. Third column: Previous 10 trajectories that reside in the memory. Black to white represents most recent to oldest.
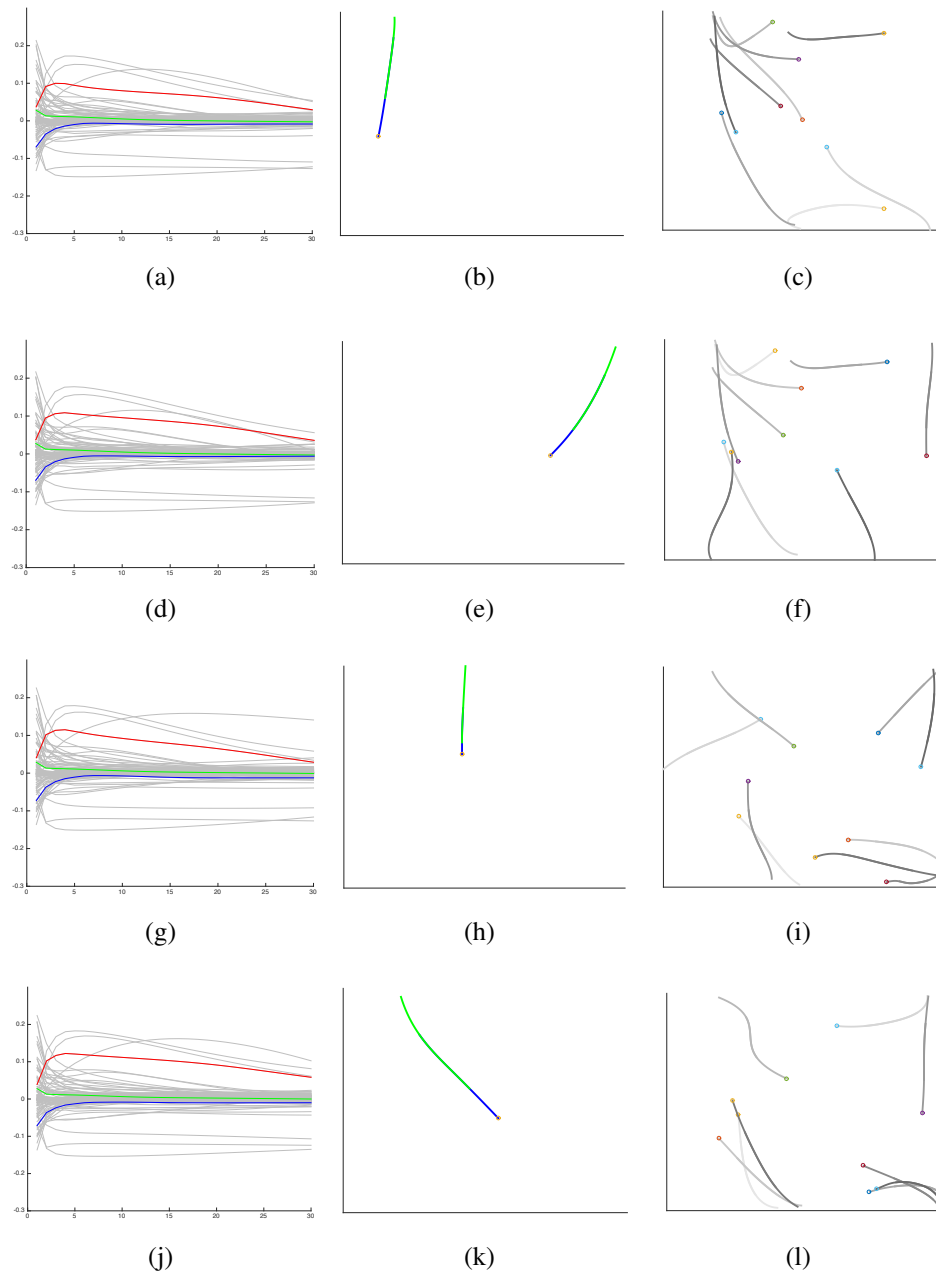
36

Figure 13: Pedestrian trajectories: Correlated activations from baseline memory model. First column: Highest correlated Memory activations for the pattern selected (in colours) and the rest of the activations (in grey) over time. Second column: The input (observed (in green) and predicted (in blue)) to the model at that time step. Third column: Previous 10 trajectories that reside in the memory. Black to white represents most recent to oldest.
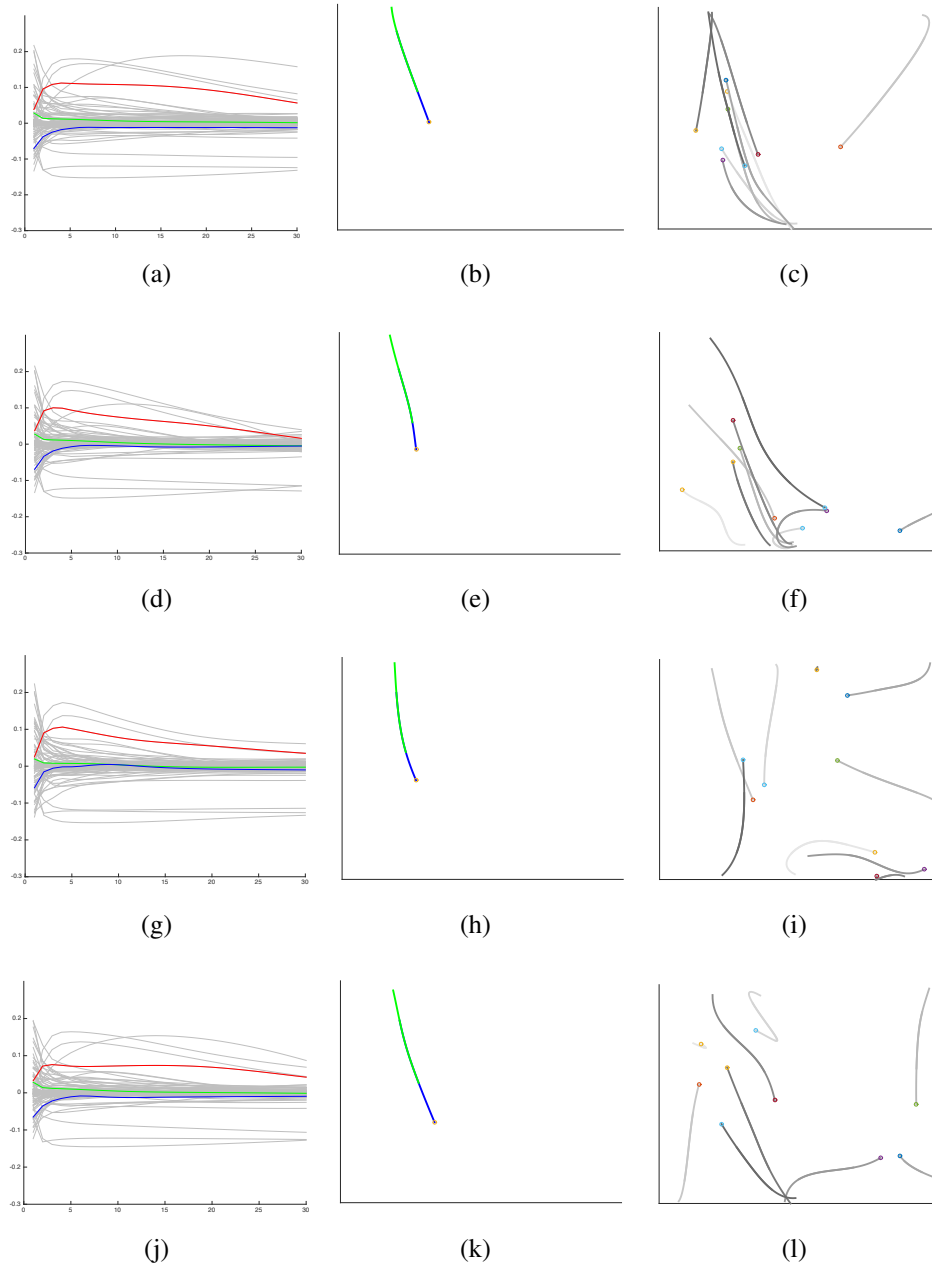
37

Figure 14: Pedestrian trajectories: Memory activations from DMN memory model for correlated inputs. First column: Memory activations over time, the pattern we are considering (in colours) and the rest of the activations (in grey). Second column: The correlated inputs (observed (in green) and predicted (in blue)). Third column: For the input selected in the second column, previous 10 trajectories that reside in the memory. Black to white represents most recent to oldest.