Generalised Structural CNNs (SCNNs) for time series data with arbitrary graph-topologies

Thomas Teh
Brain & Behaviour Lab, Imperial College London
London, UK
gim.teh16@imperial.ac.uk

Chaiyawan Auepanwiriyakul
Brain & Behaviour Lab, Imperial College London
London, UK
chaiyawan.aupenwiriyakul16@imperial.ac.uk

John A. Harston
Brain & Behaviour Lab, Imperial College London
London, UK
j.harston17@imperial.ac.uk

A. Aldo Faisal
Brain & Behaviour Lab, Imperial College London
London, UK
a.faisal@imperial.ac.uk

ABSTRACT

Deep Learning methods, specifically convolutional neural networks (CNNs), have seen a lot of success in the domain of image-based data, where the data offers a clearly structured topology in the regular lattice of pixels. This 4-neighbourhood topological simplicity makes the application of convolutional masks straightforward for time series data, such as video applications, but many high-dimensional time series data are not organised in regular lattices, and instead values may have adjacency relationships with non-trivial topologies, such as small-world networks or trees. In our application case, human kinematics, it is currently unclear how to generalise convolutional kernels in a principled manner. Therefore we define and implement here a framework for general graph-structured CNNs for time series analysis. Our algorithm automatically builds convolutional layers using the specified adjacency matrix of the data dimensions and convolutional masks that scale with the hop distance. In the limit of a lattice-topology our method produces the well-known image convolutional masks. We test our method first on synthetic data of arbitrarily-connected graphs and human hand motion capture data, where the hand is represented by a tree capturing the mechanical dependencies of the joints. We are able to demonstrate, amongst other things, that inclusion of the graph structure of the data dimensions improves model prediction significantly, when compared against a benchmark CNN model with only time convolution layers.

CCS CONCEPTS

• **Mathematics of computing** → **Geometric topology**; • **Computing methodologies** → **Supervised learning**; **Dimensionality reduction and manifold learning**; **Feature selection**;

KEYWORDS

Machine Learning, Neural Network, Convolutional Neural Network, Time-Series, Human Dynamics Modelling

, ,

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

ACM Reference Format:
Thomas Teh, Chaiyawan Auepanwiriyakul, John A. Harston, and A. Aldo Faisal. 2018. Generalised Structural CNNs (SCNNs) for time series data with arbitrary graph-topologies. In *Proceedings of* . ACM, New York, NY, USA, Article 4, 8 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

The success of deep learning, specifically convolutional neural networks (CNNs), in computer vision [22] has spurred applications of deep learning methods to domains such as natural language processing [5, 11, 20], speech recognition [14], human activity recognition [28, 30, 35] and weather forecasting [37]. By design, CNNs share parameters across the input features and have sparse connections between layers, making them effective and efficient models for exploiting the local stationarity and lattice topology of pixels in an image. Similarly, recurrent neural networks (RNNs) and sliding windows, which extract features from temporal data by reusing the model parameters across different time steps, implicitly assume a stationary distribution within the input.

In the realm of human activity modeling, the modelling processes can be broadly categorized into two categories - activity recognition and activity pattern detection. Activity recognition focuses on detection and classification of predetermined activities [30, 38] or surveillance technology [4, 28], while deep learning techniques have been used in modelling human activity, with most studies focusing on activity recognition [10, 18, 19, 30, 38] and limited set performing unsupervised learning on human-activity data [8, 16].

However, as with most CNN studies used outside of the Computer Vision domain, the use of CNNs in this case is improper, as CNNs are optimised for data with a lattice topology, such as the pixel array of an image. When used with data that doesn't conform to a lattice topology, CNN performance drops, as the convolutional function can not fully capture the correlations between neighbouring connected data nodes.

The application of deep learning models such as CNNs to human kinematics data is thus not straightforward, as the structure of human motion capture data is subjected to the constraints of human anatomy [24]. Unlike the regular lattice array of images, human motion capture data have a tree-like structure (each hand is attached to an arm, which is jointly attached to the trunk, etc.) [24]. Moreover, human kinematics data generally contains both

spatial and temporal features, and it is important to be able to capture spatio-temporal correlations between the features. Most deep learning models are only adept at modeling spatial and temporal features separately [33] or in a stage-wise manner [30, 32, 38] - the applications of deep learning models to model spatio-temporal features simultaneously requires significant ingenuity in the design of either architecture or new artificial neuron units [10, 18, 37].

We hereby demonstrate a novel CNN architecture that can deep learn time series data with an arbitrary graph structure. We combine work on adjacency matrices with traditional CNN and RNN architectures, to allow us to perform deep learning on human kinematics data. We present both a generative model and a predictive model, built with our novel architecture. We train and test several models including our own on in-house human kinematics data, and find that our Structural Convolutional Neural Networks (SCNNs) outperform time-based convolutional neural networks. We also find that within our Structural Convolution AutoEncoder (SCAE), the convolutional kernels learn to only represent ethologically relevant hand movements in a sparse manner.

1.1 Modelling Human Kinematics Data

Human kinematics data is most often represented as graph-structured spatio-temporal data. This proves a major hurdle to accurate modelling - most techniques in this regard have historically fallen short, in having no spatial or temporal convolution, or through restricting rather than incorporating graph structure, resulting in suboptimal prediction performance.

One of the earliest and simplest approaches to modelling human kinematics is the 'sliding window' method [8, 19, 30, 38], which outperforms all recurrent neural networks in short term prediction [12] for human activity recognition tasks [18, 30, 31, 38]. Whilst useful, this approach doesn't conserve the spatial correlations that exist within the input data. Another traditional approach to modelling human kinematics temporally involves building a single end-to-end architecture consisting of convolutional and recurrent layers in a stage-wise manner [30, 32]. This approach, however, lacks the capability to work on an arbitrary graph structure, as it features a regular convolution function.

To address the problems described above, several models have been proposed that feature a graph structured convolution. Li et al. [23] proposed 3 hand-crafted multi-stream bidirectional RNNs that models each part of the body separately. Even though these models have hierarchical feature extractions that allow them to achieve better classification accuracy, their fusing layers do not account for the correlation of data prior to passing into the bidirectional layers. In addition, 2 out of 3 models fail to account for the structure of and the correlation between the spatial features.

Another approach is the tree-based CNN, originally introduced in the natural language processing domain [9, 25–27]. In this model, the input to the neural network needs to be organized hierarchically in a tree graph which allows for hierarchical feature extraction. However, this model also restricts the data structure into that of a tree, not allowing for arbitrarily-defined structure. A structural recurrent neural network was proposed by [18] in order to model spatio-temporal data with such arbitrary graph structure. Whilst this approach might generate human-like motion, this model is prone to the long-term dependencies problem common to all RNN models.

2 METHODOLOGY

2.1 Data Acquisition & Preprocessing

We captured natural hand movements during daily life activities in our research group (following [2]).The glove was calibrated against optical motion tracking methods using [36]. All subjects gave written consent and the experimental procedure was approved by a local ethics committee. Subjects (N=10) wore a right-hand CyberGlove (CyberGlove Systems LLC, San Jose, CA, U.S.A.). The glove measures joint abduction in 22 hand joints using stretch sensors embedded in the material with a spatial resolution of <1 degree (see Fig. 1 for joints tracked) and a sampling rate of 90 Hz. We recorded multiple hours of data per subject, yielding over 5 million samples.

2.2 Structural Convolutional Neural Networks

To both capture the spatio-temporal correlation from within the graph, and to work with any arbitrary graph structure, we propose a novel deep learning architecture, the Structural Convolutional Neural Network (SCNN). Our network design builds on several studies [6–8, 10, 15, 18, 25–27, 29] that attempt to embed graph structure into the neural network itself. In contrast to these previous methods, our neural network architectures defines and uses specialized convolutional kernel with an arbitrarily definable adjacency matrix. This enables us to embed prior knowledge for example in the form of physically known neighbourhood relationships between sensors

To help explain our network architecture, we first defined the following for a graph with F number of nodes and the adjacency matrix $\vec{A} \in \mathbb{R}^{F\times F}$:

$$\vec{y}^{\ell-1} = \left[\vec{y}_1^{\ell-1}, \ldots, \vec{y}_F^{\ell-1}\right]^\top, \quad \textit{Previous layer's output} \tag{1}$$

$$\vec{y}^{\ell} = \left[\vec{y}_1^{\ell}, \ldots, \vec{y}_F^{\ell}\right]^\top, \quad \textit{Current layer's output} \tag{2}$$

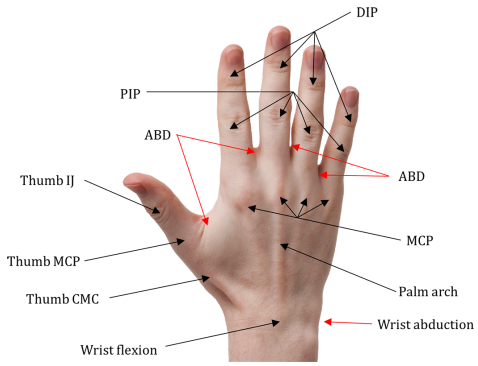
$$\vec{W}^{\ell} = \left[\vec{W}_1^{\ell}, \ldots, \vec{W}_F^{\ell}\right]^\top, \quad \textit{Current layer's weights} \tag{3}$$

$$\vec{b}^{\ell} = \left[\vec{b}_1, \ldots, \vec{b}_F\right]^\top, \quad \textit{Current layer's biases} \tag{4}$$

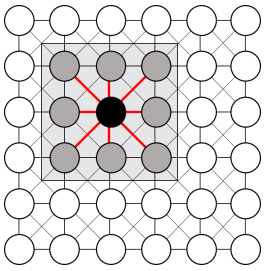
where

$$\begin{aligned}
\vec{y}^{\ell-1} &\in \mathbb{R}^{T\times F\times N},\\
\vec{y}^{\ell} &\in \mathbb{R}^{(T-(t-1))\times F\times M},\\
\vec{W}^{\ell} &\in \mathbb{R}^{F\times t\times F\times N\times M},\\
\vec{b}^{\ell} &\in \mathbb{R}^{F\times M},\\
\vec{y}_i^{\ell-1} &\in \mathbb{R}^{T\times 1\times N}, \forall i = 1, \ldots, F,\\
\vec{y}_i^{\ell} &\in \mathbb{R}^{(T-(t-1))\times 1\times N}, \forall i = 1, \ldots, F,\\
\vec{W}_i^{\ell} &\in \mathbb{R}^{t\times F\times N\times M}, \forall i = 1, \ldots, F,\\
\vec{b}_i^{\ell} &\in \mathbb{R}^{1\times M}, \forall i = 1, \ldots, F.
\end{aligned}$$

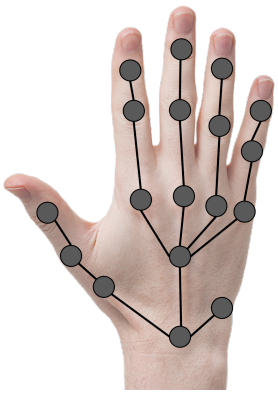
The kernel is made up of F sub-kernels and each of the sub-kernels i, which corresponds to node i, has weights W_i^{ℓ} with the dimension of $t \times F \times N \times M$. The sub-kernels are slid across the temporal dimension of the input, producing an output of $(T - (t - 1)) \times 1 \times M$ for each node i. The output is then passed through an



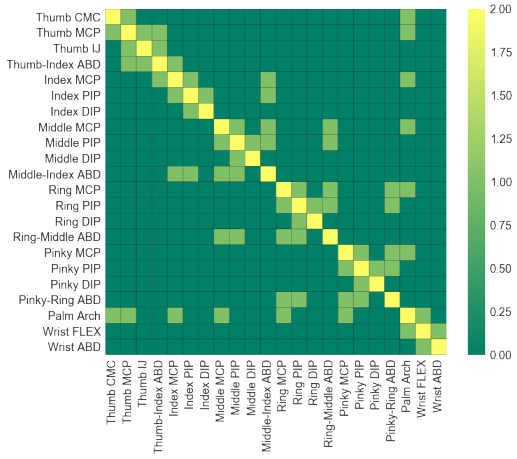
(a) Joint locations



(b) Graphical model of an image.



(c) Graphical Model of a human hand.



(d) Joint locations

Figure 1: (a) Locations of the 22 sensors embedded in CyberGlove, used to measure the angle of the joints (15 sensors), the abduction between fingers (4), wrist flexion, wrist abduction and palm arch (1 each). (b,c) The features for an image are arranged on a grid, for a given node on the lattice (black node), it has high correlation with its neighbors (grey nodes), whereas the features of the hand motion data set can be arranged according to the anatomical structure of the hand. (d) Adjacency matrix for the hand.

activation function g to produce:

$$\vec{y}_i^{\ell} = g\left(\vec{W}_i^{\ell} * \vec{y}^{\ell-1} + \vec{b}_i^{\ell}\right) \tag{5}$$

$$\vec{W}_i^{\ell} = \begin{bmatrix} \vec{w}_{i1}^{\ell} \\ \vdots \\ \vec{w}_{iF}^{\ell} \end{bmatrix} \tag{6}$$

where $*$ is the convolution operation,

$$\vec{W}_i^{\ell} * \vec{y}^{\ell-1} = \sum_{j=1}^{F} \vec{w}_{ij}^{\ell} * \vec{y}_j^{\ell-1} \tag{7}$$

and $\vec{w}_{ij}^{\ell}$ is the sub-kernel weights for the i node with its j neighbor,

$$\vec{w}_{ij}^{\ell} \in \begin{cases} \mathbb{R}^{t\times 1\times N\times M}, & if\ \vec{A}_{ij} \neq 0, \\ \mathbf{0}^{t\times 1\times N\times M}, & if\ \vec{A}_{ij} = 0. \end{cases} \tag{8}$$

where $\vec{w}_{ij}^{\ell}$ represents the sub-kernel and $\vec{A} \in \mathbb{R}^{F\times F}$ represents the adjacency matrix. Thus, if applied to the following graph:

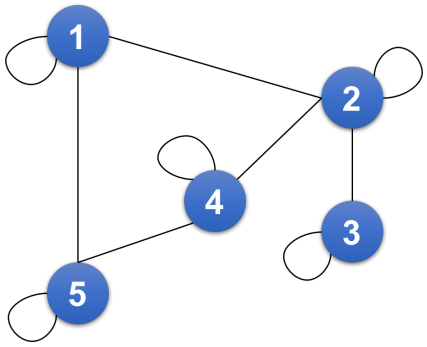


Figure 2: Example of dependency graph to be modeled by our structural convolutional neural networks.

The graph can be represented by the adjacency matrix, $\vec{A}$:

$$\vec{A} = \begin{bmatrix} 2 & 1 & 0 & 0 & 1 \\ 1 & 2 & 1 & 1 & 0 \\ 0 & 1 & 2 & 0 & 0 \\ 0 & 1 & 0 & 2 & 1 \\ 1 & 0 & 0 & 1 & 2 \end{bmatrix}. \tag{9}$$

The kernel weights $\vec{W}^{\ell}$ consists of the sub-kernels with the corresponding weights, $\vec{W}_i^{\ell}$ below:

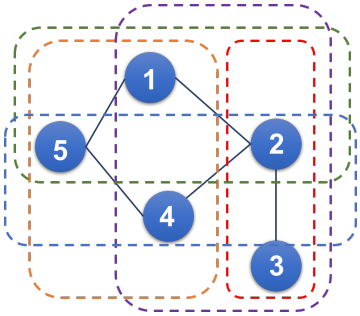
$$\vec{W}_1^{\ell} = \begin{bmatrix} \vec{w}_{11}^{\ell} & \vec{w}_{12}^{\ell} & \vec{0} & \vec{0} & \vec{w}_{15}^{\ell} \end{bmatrix}^{\top} \tag{10}$$

$$\vec{W}_2^{\ell} = \begin{bmatrix} \vec{w}_{21}^{\ell} & \vec{w}_{22}^{\ell} & \vec{w}_{23}^{\ell} & \vec{w}_{24}^{\ell} & \vec{0} \end{bmatrix}^{\top} \tag{11}$$

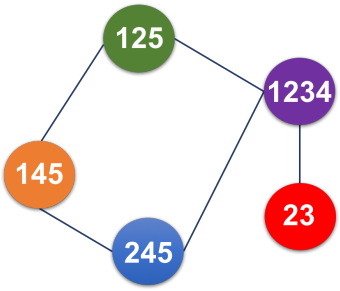
$$\vec{W}_3^{\ell} = \begin{bmatrix} \vec{0} & \vec{w}_{32}^{\ell} & \vec{w}_{33}^{\ell} & \vec{0} & \vec{0} \end{bmatrix}^{\top} \tag{12}$$

$$\vec{W}_4^{\ell} = \begin{bmatrix} \vec{0} & \vec{w}_{42}^{\ell} & \vec{0} & \vec{w}_{44}^{\ell} & \vec{w}_{45}^{\ell} \end{bmatrix}^{\top} \tag{13}$$

$$\vec{W}_5^{\ell} = \begin{bmatrix} \vec{w}_{51}^{\ell} & \vec{0} & \vec{0} & \vec{w}_{54}^{\ell} & \vec{w}_{55}^{\ell} \end{bmatrix}^{\top}. \tag{14}$$



(a) Mechanics of the sub-kernels on the input layer. The sub-kernels are represented by the dotted rounded rectangles.



(b) Structural convolutional layer. The number in the nodes represents the nodes in the input that are convolved.

Figure 3: The sub-kernels convolve only specific nodes in the input layer to produce the corresponding nodes in the convolutional output layer. For example, the sub-kernel that encompasses the input nodes 1, 2, 3 and 4 maps those input nodes to the purple node in the convolutional layer. The structure of the graph remains intact after the convolution operation. The recurrent edges are omitted for brevity.

Figure 3(a) and 3(b) shows the workings of the structural convolution for the graph in Figure 2, for input with a single channel and a single kernel. Each of the sub-kernels will only take some of the nodes of the graph for the convolution operation. Furthermore, each of the sub-kernels is distinct to the input nodes. For example, in Figure 3(a), the sub-kernel for node 2 (in purple) will take all the neighbors of node 2 that are 1 path length away for the convolution operation. The output for the convolution operation is then mapped to the corresponding node on the convolution layer.

Additionally, the use of an adjacency matrix allows an arbitrary graph structure to embed itself into the core convolution function and thus preserve any spatial correlations the data might possess prior to and after the convolution function. Furthermore, in contrast to the hard limit placed on the number of possible node connections in the previous study [29], our method allows all nodes that are reachable within a predetermined path to be covered by one convolution function and, as a result, allows for more efficient construction of CNNs for data that have large graph structures.

2.3 Neural Network Structure

We present two novel neural network architectures that leverage our graph-structural approach: 1. a structural convolutional autoencoder, and 2. structural convolutional neural network, in order to test our architecture in both supervised and unsupervised learning environments.

Structural Convolutional AutoEncoder (SCAE). For unsupervised learning, this study implements the structural convolutional autoencoder as shown in Figure 5. The model is trained initially without any regularization until weights are relatively stable. Thereafter, we impose the $L1$ regularization penalty to fine-tune the weights further.

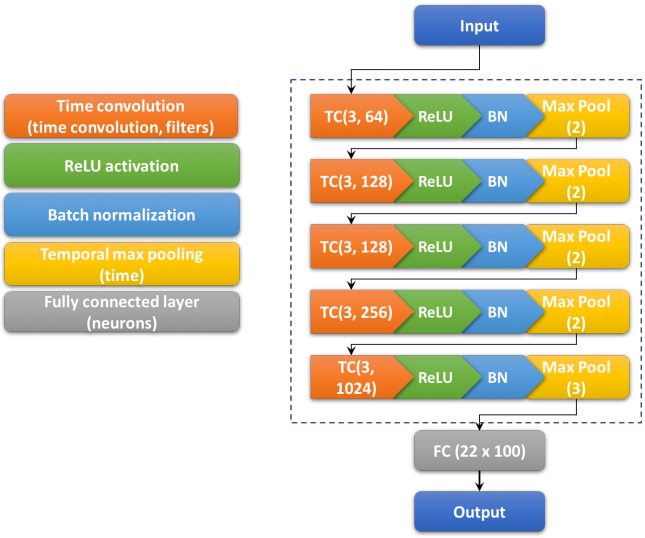


Figure 4: Time convolutional neural networks (TCNNs). Notations for the blocks are similar to Figure 6, with the exception of the convolutional layer. The convolutional layer in this model, TC(time steps, filters) only convolves the input temporally.

With the large number of parameters in the model, it is relatively easy to train such that it can reconstruct its input perfectly. However, such a model would not provide us with significant insight. As the $L1$ regularization penalty encourages sparsity within the model, by retraining the model with $L1$ regularization, we fine tune the kernel weights such that some weights will have zero-values. In essence, the model will subsequently prioritize features that are more representative of natural hand movement.

Structural Convolutional Neural Networks (SCNN). For the prediction task, we implement two different models: structural convolutional neural networks (SCNN) as in Figure 6 and temporal convolutional neural networks (TCNN) as in Figure 4. The latter TCNN model is used as a baseline from which to benchmark the relative performance of the SCNN model. The input to both models is the subsample of the time series with a 500-step time window, while the output is the subsequent 100-step time shift.

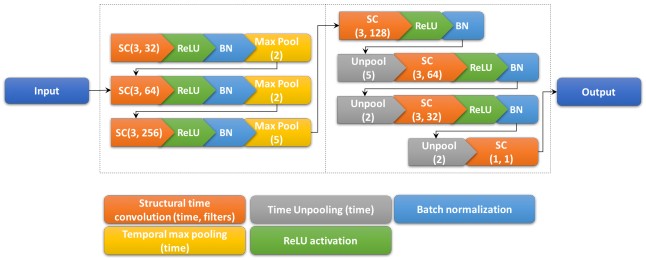


Figure 5: Structural convolutional autoencoder (SCAE). The convolution operation is done on both the spatial and temporal dimensions. SC(time steps, filters) denotes the number of time steps to convolve and the number of feature maps produced from the structural convolution. ReLU and BN denotes the ReLU activation layer and batch normalization respectively. MaxPool(time steps) denotes the temporal max pooling, Unpool(time steps) denotes temporal unpooling layer and FC(number of neurons) denotes a fully connected layer.

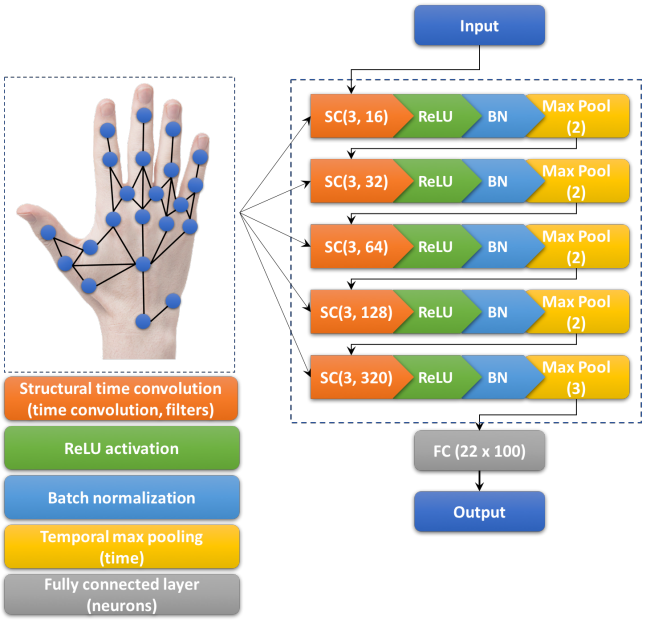


Figure 6: Structural convolutional neural networks (SCNN). The convolution operation is done on both spatial and temporal dimensions. SC(time steps, filters) denotes the number of time steps to convolve and the number of feature maps produced from the structural convolution. ReLU and BN refer to the ReLU activation layer and batch normalization respectively. MaxPool(time steps) denotes the temporal max pooling and FC(number of neurons) denotes a fully connected layer.

2.4 Neural Network Implementation

We implemented our models using the Tensorflow package [1]. In addition, we also implement $L1$ regularization on the parameter optimization. The Xavier initialization [13] was implemented to randomly initialize the weights of the convolution kernels. All biases were initialized to 0.5. An ADAM optimization technique was implemented to carry out our parameter optimization [21]. Since our system can train several neural networks in parallel, a batching process was implemented to split the training data into 32 batches. Additionally, to ensure fast convergence of the parameter values [3] and to prevent overfitting, the training batches are not constructed by subsampling sequentially, but instead by randomly shuffling into batches. Lastly, batch normalization [17] was implemented to help reduce internal covariance shift, and thus ensure fast convergence of the trainable parameters.

3 RESULTS

To train, test, and validate our neural network structure, the joint angle time series data were segregated into training, test and validation data with 55%, 35%, and 10% proportions. For standardization purposes, the training dataset attributes such as mean and variance were subsequently used to standardize every dataset to have zero mean and unit variance. For the predictive model, the inverse transform was applied at the output layer of the SCNN. A 500-time-step sliding window (equivalent to 3.56 seconds at 140 Hz) was applied to separate each dataset into multiple time frames. Additionally, a 100-step time shift was also applied to create prediction windows.

3.1 Kernels & Activation Layers Visualisation

We selectively visualize the first layer kernels of the SCAE model to understand the representations of human motor dynamics. (Figure 7). As the effects of the $L1$ regularization force most of the kernel weights to zero, the non-zero weights represent the most prominent motion features in the data. These also provide indications that we can further reduce the number of parameters in our model.

For the kernels, the weights have the same sign across the time convolution, which implies the first layer captures the positions of the joints across time. Properties of the motion dynamics, such as velocity and acceleration, are likely to be captured in the deeper layers of the network.

During training we notice a quasi-periodicity in the neural network activations. To confirm this phenomenon, we produced a recurrence plot of the activations (Figure 8), which shows a stereotypical quasi-periodic pattern, by using the following binary function:

$$R(t_i, t_j) = \begin{cases} 1, & \| a(t_i) - a(t_j) \|_2 \leq \varepsilon \\ 0, & \text{otherwise} \end{cases} \tag{15}$$

where $a(t)$ denotes the activation at time t and ε is the predetermined threshold.

It can be observed that the SCAE is able to learn both the spatial and temporal structure in the data. The fact that only selected nodes have significant activations within the feature maps shows that the model learns the prominent spatial features. In addition, the quasi-periodicity of the activations indicates that the SCAE model

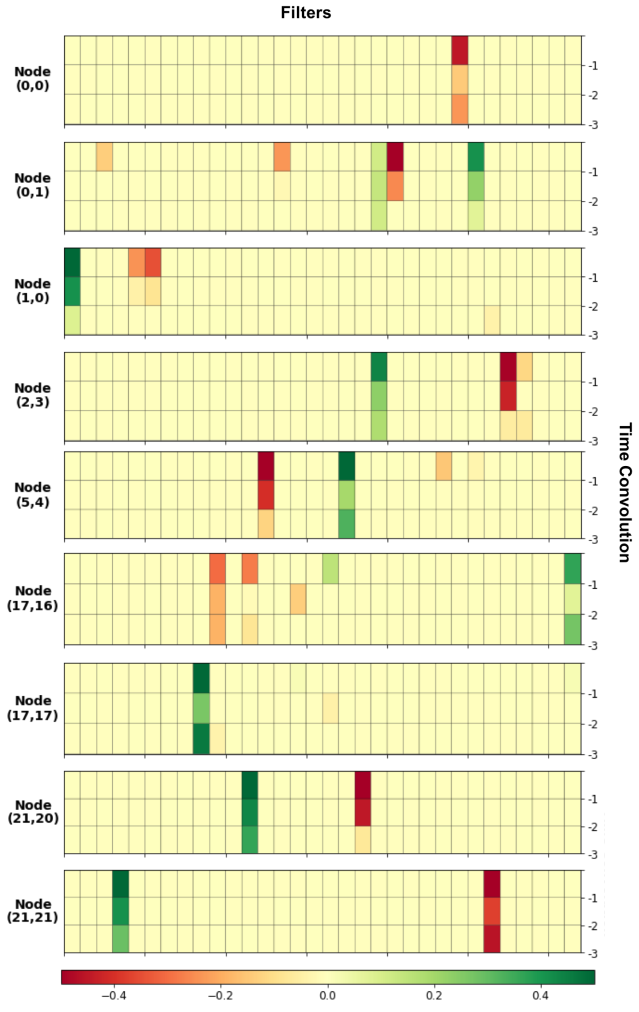


Figure 7: Visualization of the sub-kernel weights for the first layer. By imposing the $L1$ regularization, a large number weights are set to zero, and the non-zero weights are sufficient to reconstruct the input.

also captures the temporal structure in the deeper layers of the model.

3.2 Hand Movement Prediction

The inclusion of the fully connected layer enables the model to predict with higher accuracy. However, unlike regular classification tasks where the number of classes tend to be small. For a regression task, the fully-connected layer requires a much larger number of points to obtain accurate predictions for a longer prediction horizon.

Our model extends the work in [34] by including the graph structure of the features and predicting a fixed horizon instead of merely the next time step. We plot the aggregated RMSE in Figure 9.

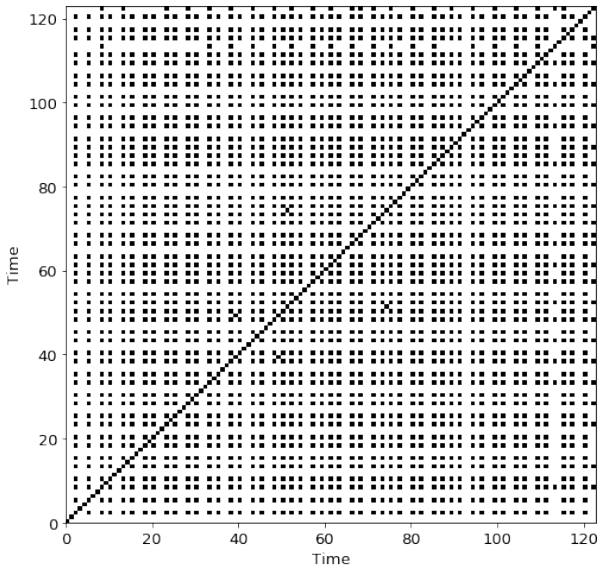


Figure 8: Recurrence plot for the activations of a specific feature map node in Layer 3. Threshold used to compute the recurrence is 1×10^{-4}.

The RMSE for the TCNN model is consistently higher than that for the SCNN model (Figure 9). Since the TCNN only incorporates the correlations between the spatial features at the fully-connected layer, the inclusion of the dependency graph of the spatial features in constructing the convolution layer is beneficial to the model's predictive power.

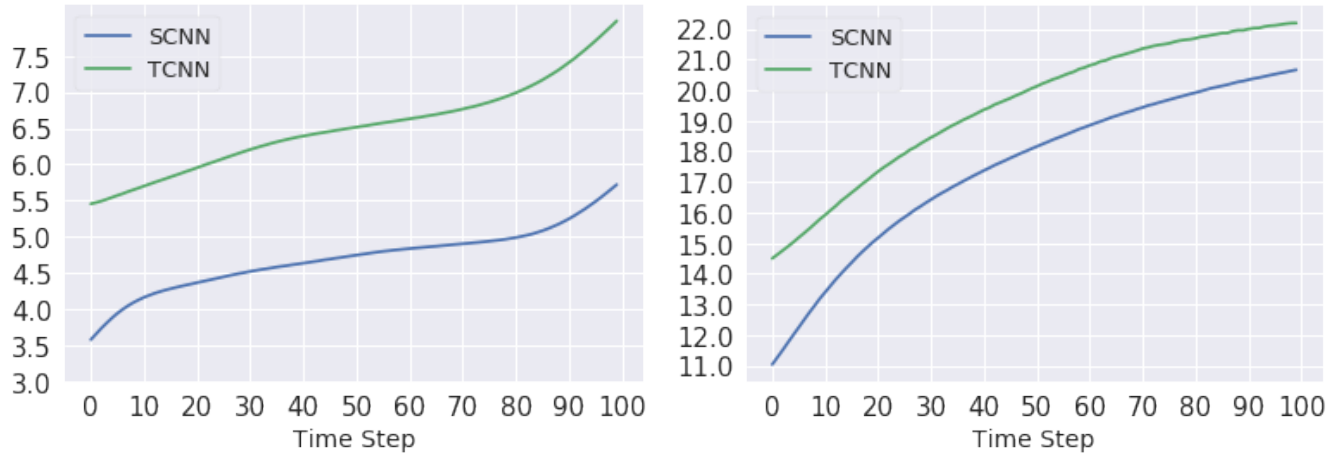
Also from Figure 9, it is observed that the RMSE for both model worsens for predictions that are many time-steps ahead. The deterioration of the RMSE across time-steps is well within our expectations, as predictions that are significantly further ahead are naturally much less reliable.

Overall, both of our models can predict the movements of the data, even at a distant prediction horizon, however they fail to capture the magnitude of those movements. The SCNN outperforms the TCNN in terms of RMSE attained. These results validate the observations from [8, 10] that the inclusion of the graph structure for human motion capture related tasks improves the prediction quality and allows us to lengthen the prediction horizon.

4 DISCUSSION

Our main methodological contribution is the introduction of the structural convolutional neural network, which allows efficient design of bespoke convolutional kernels via the specification of the dependency graph of the features. While this study focuses on the application of the model to human hand motion data, the proposed model can actually be applied to data with arbitrary topology. These special cases can be derived by specifying the adjacency matrices of the features in a specific manner.

Prediction Model. For prediction tasks, we compared two models: our structural convolutional neural networks (SCNNs) and the



(a) Training data

(b) Test data

Figure 9: RMSE for TCNN and SCNN aggregated across all features. The TCNN is shown to perform consistently worse than the equivalent SCNN model.

well-known time convolutional neural networks (TCNNs). The difference between these two networks is that of inclusion of the adjacency matrix of the spatial features in the convolution masks. Based on the predictions we obtained for both models, we observed the following:

- The SCNN model outperforms the TCNN model in terms of the RMSE and R^2 values. By embedding the topology of the spatial features of the data, the model is able include the local spatio-temporal interactions between the different joints in the early stages of the model.
- The improvement of the prediction for the SCNN stems from the inclusion of the graph structure allowing the neural network to extract more meaningful representations of movement dynamics, allowing for a higher accuracy across a longer prediction horizon.

Our approach allows us to design bespoke convolutional kernels using the adjacency matrix of the spatial features. We demonstrate here that our approach improves the prediction quality and extends the prediction horizon significantly. This efficiency comes at a price: Unlike the structural RNN by [18] and the graph CNN by [29], our current approach does not support directed edges or edges features and is limited to undirected graphs. Thus, a natural extension of this study would be the inclusion of RNNs to the SCNNs by constructing a structural convolutional recurrent neural network similar to the convolutional LSTM in [37], as a combined architecture may be better able to capture long-term spatio-temporal correlations. Beyond the inclusion of RNNs our convolutional kernel construction method can be improved by unsupervised graph structure estimation from the data.

We applied this approach to the graph structure of the human body kinematics time series, and show that we outperform conventional time convolutional neural networks. Our approach allows the development of deep learning models trained on arbitrary graph structured data, be it medical data (e.g. fMRI-based brain network activity), economic data (e.g. airline travel numbers on the airport connectivity graph) or social data (e.g. social networks variables). The largest benefit for our method's flexibility is its scalability - the structural convolution requires a single adjacency matrix for all the spatial features.

ACKNOWLEDGMENTS

This work has received funding from the European Union's Horizon 2020 research and innovation programme under grant eNHANCE (grant no 644000) – www.enhance-motion.eu.

REFERENCES

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467* (2016).

[2] Jovana J Belić and A Aldo Faisal. 2015. Decoding of human hand actions to handle missing limbs in neuroprosthetics. *Frontiers in computational neuroscience* 9 (2015).

[3] Yoshua Bengio. 2012. Practical recommendations for gradient-based training of deep architectures. In *Neural networks: Tricks of the trade*. Springer, 437–478.

[4] Lokesh Boominathan, Srinivas SS Kruthiventi, and R Venkatesh Babu. 2016. Crowdnet: a deep convolutional network for dense crowd counting. In *Proceedings of the 2016 ACM on Multimedia Conference*. ACM, 640–644.

[5] James Bradbury, Stephen Merity, Caiming Xiong, and Richard Socher. 2016. Quasi-Recurrent Neural Networks. *arXiv preprint arXiv:1611.01576* (2016).

[6] Michael M. Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. 2016. Geometric deep learning: going beyond Euclidean data. *CoRR* abs/1611.08097 (2016). arXiv:1611.08097 http://arxiv.org/abs/1611.08097

[7] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. 2013. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203* (2013).

[8] Judith Bütepage, Michael Black, Danica Kragic, and Hedvig Kjellström. 2017. Deep representation learning for human motion prediction and classification. *arXiv preprint arXiv:1702.07486* (2017).

[9] Michael Collins and Nigel Duffy. 2002. Convolution kernels for natural language. In *Advances in neural information processing systems*. 625–632.

[10] Yong Du, Wei Wang, and Liang Wang. 2015. Hierarchical recurrent neural network for skeleton based action recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1110–1118.

[11] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional Sequence to Sequence Learning. *arXiv preprint arXiv:1705.03122* (2017).

[12] Felix A Gers, Douglas Eck, and Jürgen Schmidhuber. 2001. Applying LSTM to time series predictable through time-window approaches. In *International Conference on Artificial Neural Networks*. Springer, 669–676.

[13] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. 249–256.

[14] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*. IEEE, 6645–6649.

[15] Mikael Henaff, Joan Bruna, and Yann LeCun. 2015. Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163* (2015).

[16] Daniel Holden, Jun Saito, and Taku Komura. 2016. A deep learning framework for character motion synthesis and editing. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 138.

[17] Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167* (2015).

[18] Ashesh Jain, Amir R Zamir, Silvio Savarese, and Ashutosh Saxena. 2016. Structural-RNN: Deep learning on spatio-temporal graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 5308–5317.

[19] Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. 2013. 3D convolutional neural networks for human action recognition. *IEEE transactions on pattern analysis and machine intelligence* 35, 1 (2013), 221–231.

[20] Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aaron van den Oord, Alex Graves, and Koray Kavukcuoglu. 2016. Neural machine translation in linear time. *arXiv preprint arXiv:1610.10099* (2016).

[21] Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[22] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*. 1097–1105.

[23] Jiwei Li, Minh-Thang Luong, and Dan Jurafsky. 2015. A hierarchical neural autoencoder for paragraphs and documents. *arXiv preprint arXiv:1506.01057* (2015).

[24] John Lin, Ying Wu, and Thomas S Huang. 2000. Modeling the constraints of human hand motion. In *Human Motion, 2000. Proceedings. Workshop on*. IEEE, 121–126.

[25] Lili Mou, Rui Men, Ge Li, Yan Xu, Lu Zhang, Rui Yan, and Zhi Jin. 2015. Natural language inference by tree-based convolution and heuristic matching. *arXiv preprint arXiv:1512.08422* (2015).

[26] Lili Mou, Rui Men, Ge Li, Yan Xu, Lu Zhang, Rui Yan, and Zhi Jin. 2016. Recognizing entailment and contradiction by tree-based convolution. *arXiv preprint* (2016).

[27] Lili Mou, Hao Peng, Ge Li, Yan Xu, Lu Zhang, and Zhi Jin. 2015. Discriminative neural sentence modeling by tree-based convolution. *arXiv preprint arXiv:1504.01106* (2015).

[28] Natalia Neverova, Christian Wolf, Griffin Lacey, Lex Fridman, Deepak Chandra, Brandon Barbello, and Graham Taylor. 2016. Learning human identity from motion patterns. *IEEE Access* 4 (2016), 1810–1820.

[29] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. 2016. Learning convolutional neural networks for graphs. In *International Conference on Machine Learning*. 2014–2023.

[30] Francisco Javier Ordóñez and Daniel Roggen. 2016. Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition. *Sensors* 16, 1 (2016), 115.

[31] Daniel Roggen, Alberto Calatroni, Mirco Rossi, Thomas Holleczek, Kilian Förster, Gerhard Tröster, Paul Lukowicz, David Bannach, Gerald Pirkl, Alois Ferscha, et al. 2010. Collecting complex activity datasets in highly rich networked sensor environments. In *Networked Sensing Systems (INSS), 2010 Seventh International Conference on*. IEEE, 233–240.

[32] Tara N Sainath, Oriol Vinyals, Andrew Senior, and Haşim Sak. 2015. Convolutional, long short-term memory, fully connected deep neural networks. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 4580–4584.

[33] Karen Simonyan and Andrew Zisserman. 2014. Two-stream convolutional networks for action recognition in videos. In *Advances in neural information processing systems*. 568–576.

[34] Thomas Teh. 2017. Deep Learning Sensorimotor Representations for Human Robotics. (2017).

[35] Alexander Toshev and Christian Szegedy. 2014. Deeppose: Human pose estimation via deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1653–1660.

[36] Alexandre P Vicente and A Aldo Faisal. 2013. Calibration of kinematic body sensor networks: Kinect-based gauging of data gloves âĂIJin the wildâĂİ. In *Body Sensor Networks (BSN), 2013 IEEE International Conference on*. IEEE, 1–6.

[37] SHI Xingjian, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. 2015. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In *Advances in Neural Information Processing Systems*. 802–810.

[38] Jianbo Yang, Minh Nhut Nguyen, Phyo Phyo San, Xiaoli Li, and Shonali Krishnaswamy. 2015. Deep Convolutional Neural Networks on Multichannel Time Series for Human Activity Recognition.. In *IJCAI*. 3995–4001.