Small Dynamic Neural Networks for Gesture Classification with The Rulers (a Digital Musical Instrument)

Vanessa Yaremchuk
IDMIL, CIRMMT
McGill University
Montreal, Canada
vanessa.yaremchuk@gmail.com

Carolina Brum Medeiros
IDMIL, CIRMMT
McGill University
Montreal, Canada
carolina.medeiros@mail.mcgill.ca

Marcelo M. Wanderley
IDMIL, CIRMMT
McGill University
Montreal, Canada
marcelo.wanderley@mcgill.ca

ABSTRACT

The Rulers is a Digital Musical Instrument with 7 metal beams, each of which is fixed at one end. It uses infrared sensors, Hall sensors, and strain gauges to estimate deflection. These sensors each perform better or worse depending on the class of gesture the user is making, motivating sensor fusion practices. Residuals between Kalman filters and sensor output are calculated and used as input to a recurrent neural network which outputs a classification that determines which processing parameters and sensor measurements are employed. Multiple instances (30) of layer recurrent neural networks with a single hidden layer varying in size from 1 to 10 processing units were trained, and tested on previously unseen data. The best performing neural network has only 3 hidden units and has a sufficiently low error rate to be good candidate for gesture classification.

This paper demonstrates that: dynamic networks outperform feedforward networks for this type of gesture classification, a small network can handle a problem of this level of complexity, recurrent networks of this size are fast enough for real-time applications of this type, and the importance of training multiple instances of each network architecture and selecting the best performing one from within that set.

Author Keywords

recurrent neural network, gesture, classification, digital musical instrument, sensor fusion

CCS Concepts

•**Computing methodologies** → **Neural networks;**
•**Hardware** → *Sensor applications and deployments;*
•**Applied computing** → *Sound and music computing;*

1. INTRODUCTION AND BACKGROUND

Digital Musical Instruments (DMIs) need to be sufficiently robust to work in a variety of performance settings where conditions (e.g., stage lighting or ambient temperature of a venue) might be inconsistent or unknown ahead of time. Instrument stability is also necessary for practice and learning. Consistent repeatable sensing is key, and it may be that no one sensor type is optimal for all playing circumstances. In these cases, sensor fusion practices offer a means to DMI stability and machine learning can be a useful tool as part of this approach.

The specific DMI discussed here includes multiple sensor types, and sensor performance depends on the class of gesture being made. Sensor fusion in this case requires the use of filters, machine learning, or other classifiers to select sensor input according to which class the current gesture belongs. There are many different machine learning approaches that have been used effectively in DMI design [2], and as the sophistication of these approaches increases it becomes important to include deeper exploration within a category of machine learning and not just comparisons across basic implementations from each category. To this end, multiple artificial neural networks (ANNs) of various sizes, both with and without recurrent connections, are trained offline on gesture data, and testing data is used to select the best performing one. The selected ANN would be kept static and not require additional training, so it can be simulated more efficiently for real-time use with the DMI.

Multilayer perceptrons (i.e., ANNs without any recurrent connections) have been used to customise gesture mapping in DMIs by training [3] or retraining [19] gesture recognition systems, retraining gesture primitive identifiers [9] or by retraining the mapping between gesture and sound synthesis model directly [6, 13]. In all of these cases, adjustments are made to account for differences in individual users, and the neural network training is not unlike an extended system initialisation process. The multilayer perceptrons are primarily recognising hand poses and positions and not movement or gesture directly. (See figure 1 for an example of a multilayer perceptron.)

Recurrent neural networks are ANNs that include feedback connections and tapped delay lines, allowing the order in which data is presented to have an effect. They have been used in hand gesture recognition for Japanese[11], Arabic[14], and Korean[10] sign language. They have been used as well in recognising hand gestures for image creation[4], recognising full-body movement [5], and recognising movements based on sensors in a hand-held device [1] or sensors embedded in gloves [20]. See figure 2 for an example of a recurrent neural network.

The method described in this paper finds small networks that are still able to distinguish classes of gesture and even out perform larger networks. It emphasises the value of trying multiple variations of network topology and demonstrates that recurrent connections improve classification performance for the types of gestures considered here. It further demonstrates the importance of training many identically structured networks and selecting the best performing one, which is a step that is frequently either not taken or not explicitly mentioned when ANNs are applied to specific problem sets. It also demonstrates that a recurrent neu-

NIME'19, June 3-6, 2019, Federal University of Rio Grande do Sul, Porto Alegre, Brazil.

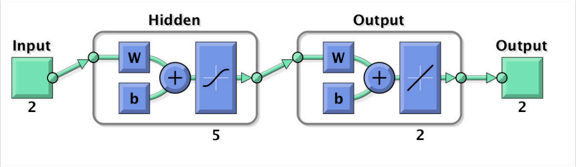

Figure 1: Multilayer Perceptron
A multilayer perceptron with one hidden layer containing 5 hidden units, where each processing unit has a vector of weights, W, and a bias, b. (This graphical diagram was generated using 'view' in Matlab.)

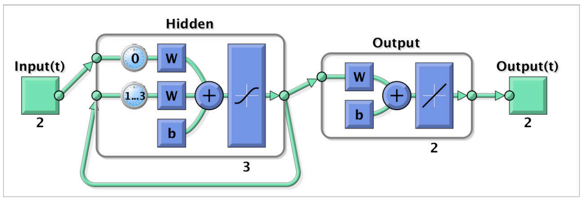

Figure 2: Layer Recurrent Network
A recurrent neural network with one hidden layer containing 3 hidden units, and tapped delay lines with 1, 2, and 3 previous iterations. Each processing unit has a vector of weights, W, and a bias, b. Hidden units have weights for inputs from the input layer as well as weights for feedback from tapped delay lines of the hidden layer output. (This graphical diagram was generated using 'view' in Matlab.)

ral network of this size can be fast enough for a real-time application such as a DMI.

2. CONTEXT

2.1 The Rulers

The Rulers (see figure 3) is a Digital Musical Instrument (DMI) with 7 aluminum beams, which includes infrared sensors, Hall sensors and strain gauges [15]. The beams (or tines) are of different lengths and are fixed at one end, and the sensors are used to estimate deflection when the beams are manipulated.

Medeiros et al. [17] recorded data from all 3 types of sensor simultaneously with detailed motion capture data to determine the accuracy of the various sensors and found that the relative measurement error of the sensors depends on the class of gesture the user is making. No single sensor type performed optimally across all gestures, so sensor fusion methods are used to combine the sensor data in order to produce an output that is more accurate overall than any one sensor type on its own.

2.2 Gesture Classification for Sensor Fusion

The gestures made when using The Rulers can be classified as either plucking or bending, where the former involves the user releasing the beam and allowing it to oscillate freely, and the latter involves the user pressing down or pulling up on a beam while maintaining contact with it. The relative measurement error of the sensors is different for these two types of gesture, with Hall sensors performing best for plucking motions and strain gauges performing best for bending motions [17].

Real-time performance requirements reduce the set of viable filter options and the human-driven input data was found to be sufficiently unpredictable that it does not follow a consistent physical model, so a modified multiple model Kalman filter was implemented [17]. A Kalman filter estimates a variable recursively using the previous time step's calculation and the current input measurements. The residuals between the Kalman filters and the sensor output are calculated, and it is this that is used as input to be classified as either pluck or bend. This classification then determines which processing parameters and sensor measurements are employed towards sound synthesis. This paper explores artificial neural networks (ANNs) for performing this classification task. Several works deal with the simultaneous application of neural networks and prediction filters, and primarily use training algorithms to tune the parameters of the Kalman filter such as the process and measurement noise distributions and the dynamic model itself [8].

Figure 3: The Rulers.
The Rulers. A DMI with 7 metal beams that uses infrared sensors, Hall sensors, and strain gauges. Pictured: Dr. Xenia Pestova Photo by: Vanessa Yaremchuk

3. RECURRENT NEURAL NETWORKS FOR CLASSIFICATION

3.1 Training data

Training data was generated by calculating and recording the differences between the Kalman filters estimates and the best sensor output for each gesture, while performing gestures of each of the two classes, pluck and bend. This was used to produce sequences of input and desired output pairs, where the inputs are the residuals from the Kalman filters and there are two outputs corresponding to pluck or bend respectively. That is, the input is the data from the sensors after being preprocessed by calculating the residual from a modified multiple model Kalman filter. This results in just two input units, as seen in figures 1 and 2, which contributes to keeping the networks small.

The target output is (1, -1) for pluck or (-1, 1) for bend. As there are just two classes, this could have been implemented with a single output, but using one output for each class reduces required training times. The actual network outputs are real values, so after training, the resulting classification for any input is indicated by the output with the largest value.

Inputs are also normalised to fall in the range -1 to 1, but this step is implemented as part of the ANN. This keeps preprocessing consistent across all incoming data, and also allows it to be encoded as part of the trained ANN that will eventually be simulated with stand-alone code. This negates the need for an additional step in the system and further restricts the potential overall impact of the ANN on latency.

The training data was made from 868 input and output pairs corresponding to pluck data and 982 input and output pairs corresponding to bend data. The networks were later tested on a separate batch of data containing 2505 and 3660 pluck and bend pairs respectively.

3.2 Artificial Neural Network Architecture

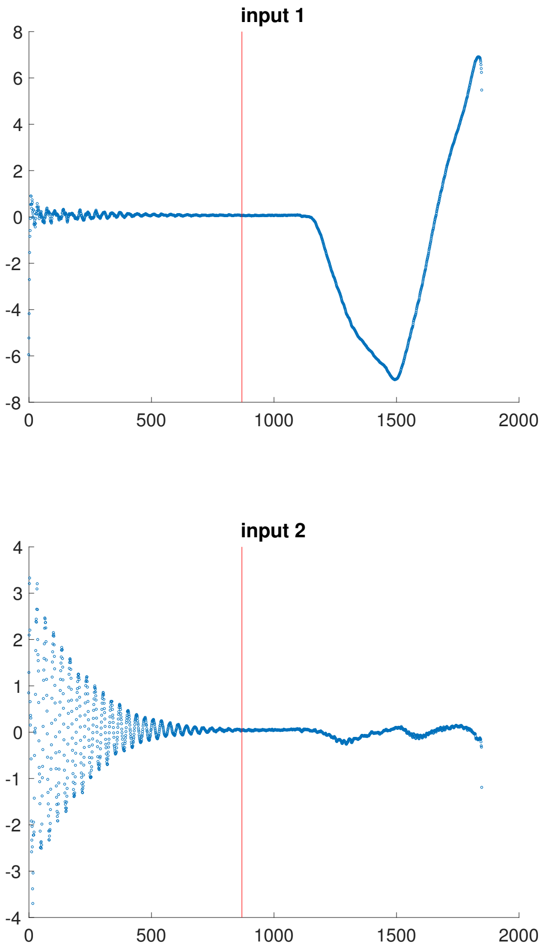

Figure 4: Example Input
An example of the input to the two input units of the ANN. The horizontal axis gives the sample number, and the vertical axis gives the value of the corresponding input unit. The data points before the red vertical line are the residuals of the Kalman filters for a pluck, and the data points after the vertical red line are the residuals for bending.

All connectionist architectures posses long-term memory in the form of connection weights[12], but the tapped delay lines associated with the feedback (i.e., recurrent) connections present in recurrent systems are one way of incorporating a form of short-term memory and enabling networks to manage patterns that vary across time. The long-term memory embodied in the connection weights remains static after network training is complete, but the short-term memory changes in response to input during both training and operation.

An example of the input from a pluck followed by some bending is shown in figure 4. When viewed as sequences of data, the input looks distinguishable, but there is definitely overlap (primarily between -1 and 1) where both gestures have data points.

Matlab's Neural Network Toolbox was used to implement feedforward neural networks and layer recurrent neural networks. The tapped delay lines in the recurrent networks were made with delays from 1, 2, and 3 previous iterations. Recurrent connections or some other means of accounting for sequence order was expected to be required for gesture classification, especially since the intersection of input values of each class, when viewed as isolated points, is non-zero for plucking and bending. Nevertheless, feedforward neural networks (which do not have an encoding of sequence or ordering by default) were trained as well to test this assumption.

All ANNs had 2 input units, 2 output units, and a single hidden layer varying in size from 1 to 10 processing units, and were trained using Bayesian regularization to improve generalisation. Regularization reduces overfitting by adjusting the performance function to result in smaller weights and biases in the trained network, and Bayesian regularization in particular does this by treating the weights and biases as random variables with unknown variances to be estimated [7]. Generalisation is also improved by keeping networks as small as possible, while still maintaining good performance, and this is why a range of sizes were trained and tested.

As is standard, networks are initialised with randomised weights and biases before training, and convergence during training can be effected by this starting state. For this reason, multiple instances (30) of each neural network size were trained to allow for this effect and to improve the chances of producing trained networks that perform well.

Between the variety of sizes and the two architectures (i.e., feedforward and recurrent), there are 20 different networks, and training 30 instances of each results in 600 networks in total. In order to accommodate training so many networks, the training algorithm was modified to run across multiple CPU cores by using Matlab's Parallel Computing Toolbox, and executed on a supercomputer. This level of computation is only relevant for the training phase, and only then because of the large number of networks being trained. As ANN training is so readily done in parallel, it is also an option to use the increasingly available range of inexpensive cloud computing resources for this task. Running a single already trained network will take comparatively quite few resources.

3.3 Network Evaluation and Selection

Testing data was generated in the same manner as the training data and kept separate from the training data so that all ANNs could be tested on previously unseen data. Testing sets for the recurrent networks were constructed to ensure that all 4 types of transition were present. That is, the transition from bend to bend, bend to pluck, pluck to bend, and pluck to pluck. For the feedforward networks this ordering is irrelevant, and makes no difference in the network output. There were 2505 input output pairs generated from plucking and 3660 pairs from bending, and this data is distinct from the data used in training.

The best performing feedforward network had 5 hidden units and correctly classified 92.1% of the test data and 93.0% of the training data. Figure 5 shows the output for this network on testing data. As the order of the input does not have an effect, all of the pluck data comes first, followed by the bend data, to make it easier to follow. The two outputs are just reflections of each other, so anything above 0 in the output 1 graph is considered to be classified as pluck and anything above 0 in the output 2 graph is classified as bend. The misclassifications are scattered throughout the output, which could make for jarring sudden changes mid-gesture of the processing parameters and sensor measurements that are employed towards sound synthesis.

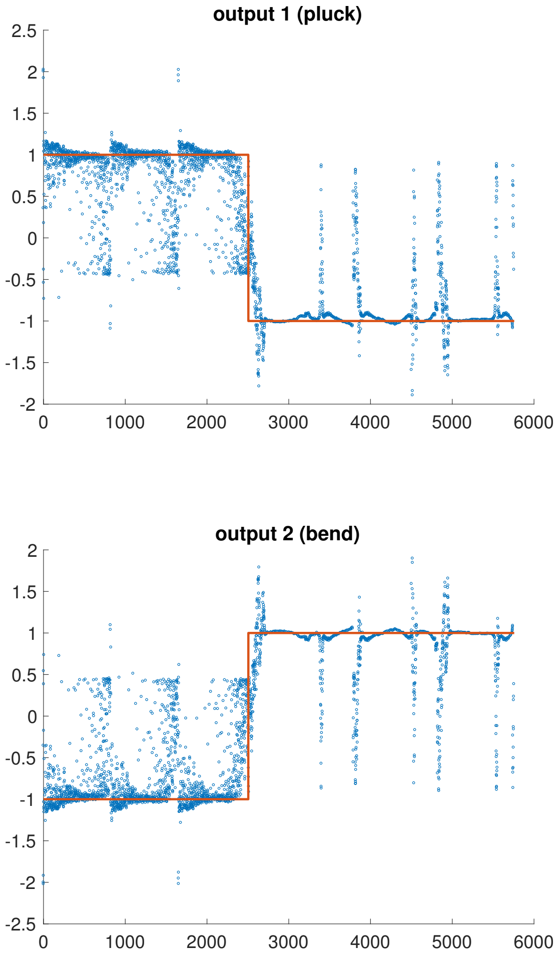

Figure 5: Output from the best performing feedforward ANN.
Output from the best performing feedforward ANN. The horizontal axis gives the sample number, and the vertical axis gives the response of the output unit. The desired output is plotted in red.

The best performing recurrent network has 3 hidden units and correctly classified 97.5% of the test data and 99.6% of the training data. Figure 6 shows the output for this network on testing data that alternates between plucks and bends, and figure 7 shows the output for this network on testing data that presents a series of plucks followed by a series of bends. The errors in these figures occur briefly and only take place when transitioning between gestures (including sometimes gestures of the same type as occurs between the second and third pluck in figure 7).

The best performing recurrent neural network has a sufficiently low error rate, and enough consistency within a gesture, to be a good candidate for gesture classification. The feedforward networks (i.e., without recurrent connections) were found to have too many mid-gesture classification errors to perform smoothly.

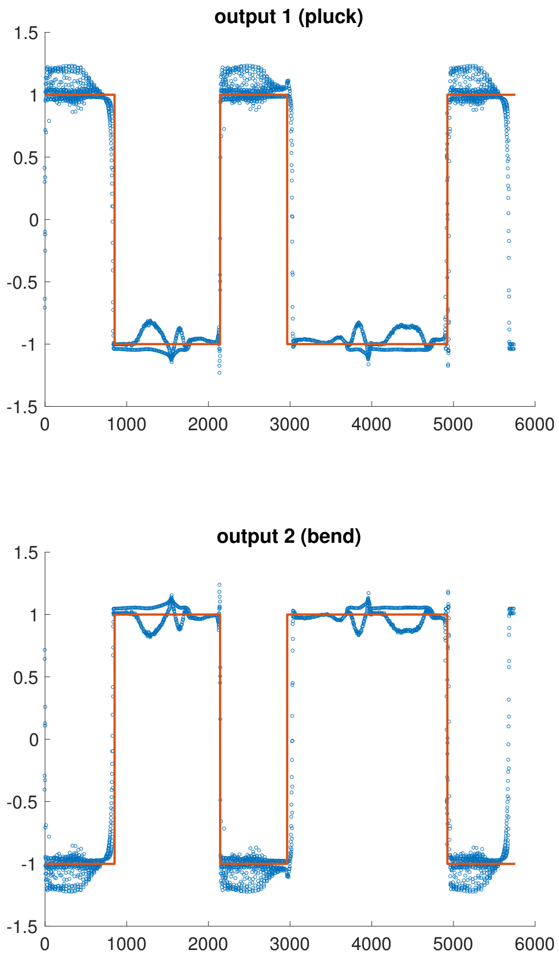

Figure 6: Output from the best performing recurrent ANN.
Output from the best performing recurrent ANN in response to alternating pluck and bend data. The horizontal axis gives the sample number, and the vertical axis gives the response of the output unit. The desired output is plotted in red.

By comparison, of the 30 recurrent networks with 3 hidden units that were trained, the worst performing one correctly classified only 57.4% of the test data and 56.9% of the training data. The training conditions were the same as those for the best performing network, but the (randomly initialised) starting state for the weights and biases led to a very different result. Figure 8 shows the significant performance range over the 30 recurrent networks of each type that were trained. This is why it is important to not just train one network.

4. RESULTS AND DISCUSSION

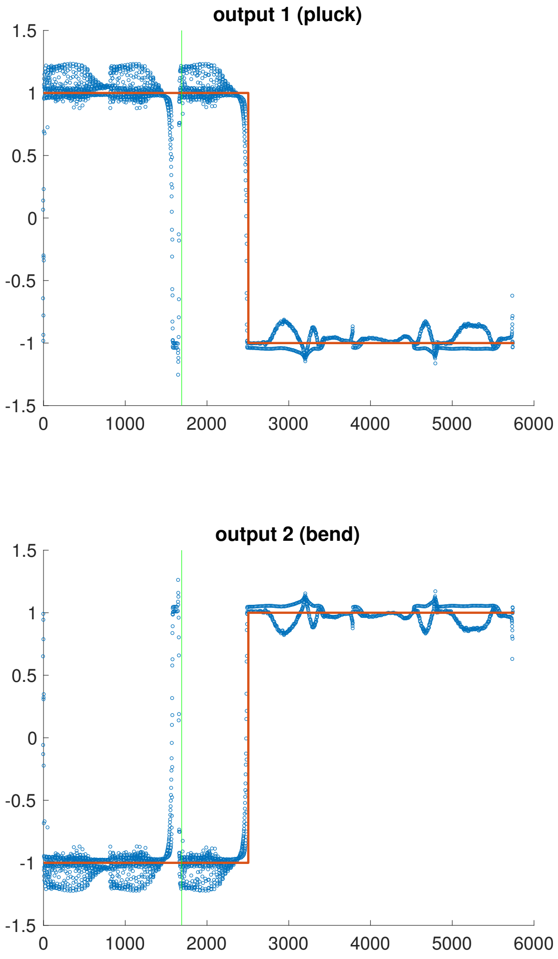

Figure 7: Output from the best performing recurrent ANN.
Output from the best performing recurrent ANN in response to pluck data followed by bend data. The horizontal axis gives the sample number, and the vertical axis gives the response of the output unit. The desired output is plotted in red. The green vertical line indicates the transition between the second and third pluck.

Once deployed, continuing to train ANNs is often unnecessary, and can even be counter productive when used with DMIs in particular, as further training can work against DMI stability and result in something that doesn't behave consistently enough for practicing to be satisfying. This is especially the case for applications where the ANN is part of a sensor fusion approach, and not specifically tied to some element of the interface that needs to be customisable or tuneable.

Trained ANNs can be simulated with stand-alone executables, libraries or code. Stand-alone code for the best performing recurrent ANN was generated and its accuracy was tested against the full neural network on the test data and training data. The difference in output between the stand-alone code and the full neural network it was simulating was compared for each input in the data set, and never exceeded $1.66e^{-11}$, which for this application is negligible. For ANNs trained using Matlab in particular, Matlab Coder^{TM} can also be used to generate C or C++ code that simulates the trained network, which in some circumstances can further reduce running time.

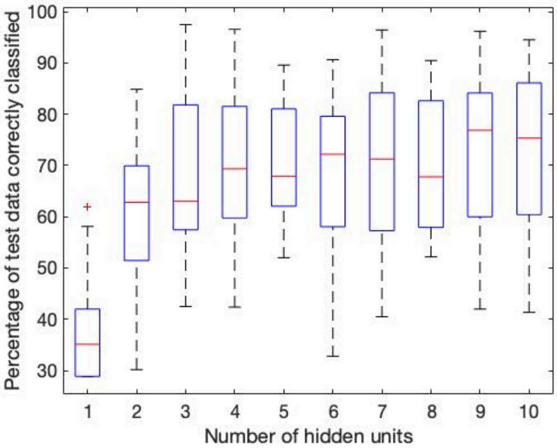

Figure 8: Classification performance of trained recurrent networks.
Box plots showing the variation in percentage of test data correctly classified over the 30 trained recurrent networks of each size (from one hidden unit to 10 hidden units). This demonstrates the importance of training and comparing multiple networks with same hyperparameters.

An advantage to using such a small network is that its impact on latency is minimal, even when run on somewhat older hardware. For instance, the stand-alone network, when run on a 2012 Mac mini with a 2.6GHz Intel Core i7 with 16GB of memory, processes 14807 input pairs per second. Most of the NIME applications use Arduino as a microprocessor [16]. The standard analog to digital conversion for Arduino boards has 10 bits resolution and their successive approximation circuitry can operate at a maximum input clock frequency of 200kHz [18]. The maximum time cost to operate a single ADC conversion is 38.5 clock cycles, which corresponds to 0.1925 ms. The time cost to convert three analog channels (strain gauge, IR, and Hall sensors) is then 0.5775 ms. A safe reading rate would be a measurement interval of 10 times the conversion cost (approximately 173 Hz), and the maximum reading rate for the application in this paper is 1,732 Hz. The maximum serial communication baud rate recommended for Arduino boards is 115,200 bps and the default data packet is 8 data bits, with no parity, one start bit, and one stop bit. In order to send 10 bits per channel, 2 bytes per channel are needed. For 3 channels, 6 bytes of information are needed, which leads to a maximum baud rate of 1,920 bytes per second or 1,920 Hz for the measurements. Finally, we are limited both in the ADC side and in the serial communication side on how fast we can acquire data. A 1,500 fps capture rate for the analog sensors would produce 1,500 pairs of residuals per second for the ANN input. The accumulated delay caused by the ANN for a second of data would be $(1500 \times 1000)/14807 = 101$ ms. That is, a frame is computed every 0.67 ms, and for each frame the ANN contributes with a delay of 0.0672 ms. Finally, we can guarantee that for the capture rate of 1500 fps, the delay added by the NN is approximately 10% of the interval between measurements.

5. CONCLUSIONS

Increasingly, researchers have access to the computing resources required to rapidly train large numbers of ANNs and test for ones that perform well. Training multiple ANNs makes it possible to experiment with sizes and find the smallest effective network which improves how well the solution generalises to new data. It also makes it possible to compare networks of the same size with different randomly initialized starting weights, and control for any effect that may come from that.

Simulations of trained ANNs require comparatively quite minimal resources to run, and this makes it a potentially viable option to use in real-time applications, such as sensor fusion. This is especially the case with ANNs as small as the ones discussed in this paper.

6. ACKNOWLEDGMENTS

This research was enabled in part by support provided by Calcul Québec (www.calculquebec.ca) and Compute Canada (www.computecanada.ca). Computations were made on the supercomputer Guillimin from McGill University, managed by Calcul Québec and Compute Canada. The operation of this supercomputer is funded by the Canada Foundation for Innovation (CFI), the ministère de l'Économie, de la science et de l'innovation du Québec (MESI) and the Fonds de recherche du Québec - Nature et technologies (FRQ-NT).

The authors would also like to thank the anonymous referees for their valuable comments and helpful suggestions.

7. REFERENCES

[1] G. Bailador, D. Roggen, G. Tröster, and G. Triviño. Real time gesture recognition using continuous time recurrent neural networks. *Proceedings of the ICST 2nd international conference on Body area networks*, page 15, 2007.

[2] B. Caramiaux and A. Tanaka. Machine Learning of Musical Gestures. *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 513–518, 2013.

[3] A. Cont, T. Coduys, and C. Henry. Real-time gesture mapping in pd environment using neural networks. In *Proceedings of the 2004 Conference on New Interfaces for Musical Expression*, NIME '04, pages 39–42, Singapore, Singapore, 2004. National University of Singapore.

[4] A. Corradini and P. Cohen. Multimodal speech-gesture interface for handfree painting on a virtual paper using partial recurrent neural networks as gesture recognizer. *Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN'02 (Cat. No.02CH37290)*, 3, 2002.

[5] Y. Du, W. Wang, and L. Wang. Hierarchical recurrent neural network for skeleton based action recognition. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 07-12-June-2015:1110–1118, 2015.

[6] S. Fels and G. Hinton. Glove-talkii-a neural-network interface which maps gestures to parallel formant speech synthesizer controls. *Neural Networks, IEEE Transactions on*, 9(1):205 –212, jan. 1998.

[7] F. D. Foresee and M. T. Hagan. Gauss-newton approximation to bayesian learning. In *Neural Networks, 1997., International Conference on*, volume 3, pages 1930–1935. IEEE, 1997.

[8] S. S. Haykin. *Kalman Filtering and Neural Networks.* John Wiley & Sons, Inc., New York, NY, USA, 2001.

[9] T. Ilmonen and T. Takala. Conductor following with artificial neural networks. In *Proceedings of the 1999 International Computer Music Conference, ICMC 1999, Beijing, China, October 22-27, 1999*, 1999.

[10] J. S. Kim, W. Jang, and Z. Bien. A dynamic gesture recognition system for the Korean Sign Language (KSL). *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 26(2):354–359, 1996.

[11] M. Kouchi and H. Taguchi. Gesture Recognition using Recurrent Neural Networks. *Acm*, pages 237–242, 1991.

[12] S. C. Kremer. Spatiotemporal Connectionist Networks: A Taxonomy and Review. *Neural Computation*, 13:249–306, 2001.

[13] M. A. Lee, G. Garnett, and D. Wessel. An adaptive conductor follower. In *International Computer Music Conference*, pages 454–455, 1992.

[14] M. Maraqa and R. Abu-Zaiter. Recognition of Arabic Sign Language (ArSL) using recurrent neural networks. *1st International Conference on the Applications of Digital Information and Web Technologies, ICADIWT 2008*, 2012(February):478–481, 2008.

[15] C. B. Medeiros and M. M. Wanderley. Evaluation of Sensor Technologies for the Rulers, a Kalimba-Like Digital Musical Instrument. *Smc'11*, pages 518–525, 2011.

[16] C. B. Medeiros and M. M. Wanderley. A comprehensive review of sensors and instrumentation methods in devices for musical expression. *Sensors (Basel, Switzerland)*, 14(8):13556–13591, 2014.

[17] C. B. Medeiros and M. M. Wanderley. Multiple-model linear kalman filter framework for unpredictable signals. *IEEE Sensors Journal*, 2014.

[18] Microchip. *ATmega48A/PA/88A/PA/168A/PA/328/P datasheet*, 11 2018. Accessed in April 3rd, 2019.

[19] P. Modler. Neural networks for mapping gestures to sound synthesis. In M. Wanderley and M. Battier, editors, *Trends in Gestural Control of Music.* Ircam - Centre Pompidou, 2000.

[20] P. Vamplew and A. Adams. Recognition and anticipation of hand motions using a recurrent neural network. *Proceedings of IEEE International Conference on Neural Networks*, 3:2904–2907, 1995.