# Predictive Musical Interaction with MDRNNs

**Charles P. Martin**
Department of Informatics
University of Oslo
charlepm@ifi.uio.no

**Jim Torresen**
Department of Informatics
University of Oslo
jimtoer@ifi.uio.no

## Abstract

This paper is about creating digital musical instruments (DMIs) where a predictive model is integrated into the interactive system. Rather than predicting symbolic music (e.g., MIDI notes), our systems predict future control data from the user and precise temporal information. We propose that a mixture density recurrent neural network (MDRNN) is an appropriate model for this task. The predictions can be used to fill-in control data for when the user stops performing, or as a kind of "filter" on the user's own input. We describe our motivations, two NIMEs applying this idea, and future directions.

## 1 Introduction

In this paper, we consider how mixture density recurrent neural networks (MDRNNs) (Bishop, 1994; Graves, 2013) can be applied to real-time gestural prediction in new interfaces for musical expression (NIMEs). While research applying deep learning to music *generation* is rapidly appearing, few of these systems have been applied in the service of real-time musical *performance*. We feel that deep ANNs can extend creative possibilities for NIME performers and designers; however, these users need better tools to make use of such models.

Present work in musical AI is usually focussed on high-level symbolic music; however, most NIME control data is better represented as low-level continuous sensor values. We propose to use MDRNNs to model this data, including time-deltas between each reading. This approach has the advantage of modelling music at the embodied (Leman et al., 2018) control level; such models imitate performing on instruments, not composing music. Another advantage is in representing rhythms absolutely—as a sequence of real-valued time-deltas—rather than being limited to a sixteenth-note grid. MDRNNs have previously been applied to control data in sketching (Ha and Eck, 2017) and handwriting (Graves, 2013), both creative tasks.

We have developed our own Keras MDN layer[1], along with a set of example applications, in order to accelerate development of musical MDRNN models. This layer uses factored multivariate normal distributions for each mixture component and so allows an arbitrary number of dimensions of control data, as well as time. We aim to expand this tool to provide better solutions for MDRNN-based NIME development to artists and computer musicians.

We imagine that artists could train MDRNN models on small datasets of creative interaction data, tailored to commercial or DIY interfaces applied in their practice. While small models may not represent all possible musical interactions, they might perform well enough to imitate aspects of an individual artist's style. Below, we discuss systems from our lab where MDRNNs have been applied for predictive musical interaction.

NeurIPS 2018 Workshop on Machine Learning for Creativity and Design, Montréal, Canada.

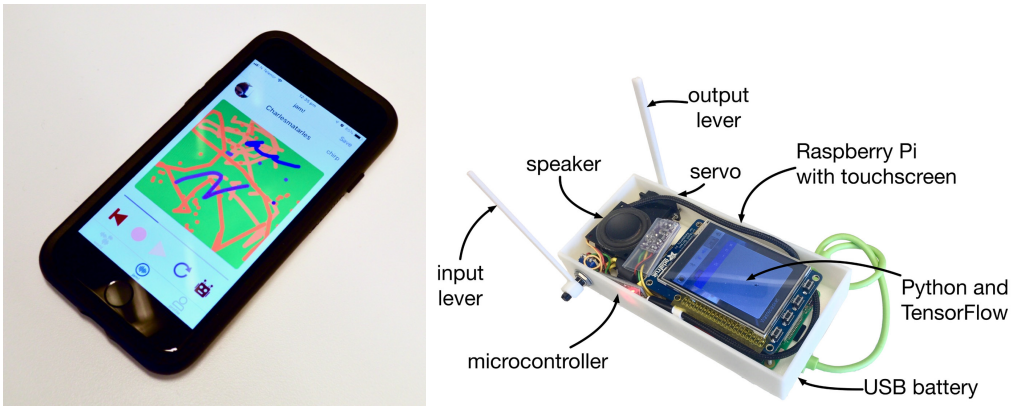


Figure 1: Musical applications of MDRNNs: (left) RoboJam replying to a performance in the MicroJam app, and (right), the EMPI self-contained NIME with enclosure open.

### 1.1 RoboJam

RoboJam (Martin and Torresen, 2018) is a cloud-based agent built into our MicroJam mobile music app. With this app, users can create short (5 second long) performances by swirling or tapping on the touch screen area. Extra layers of performance can be added over the top to create more complex performances by other users, or solo. RoboJam is designed to generate new 5s performance layers in response to one provided by the user. It does this by conditioning an MDRNN's hidden state with the given performance and then sampling new gestures until 5 more seconds have been produced. This new performance is layered with the old and played back with a different synthesis sound. RoboJam is presently implemented in Keras using a 3D MDN layer (x-position, y-position, time-delta) and runs on a cloud server. It is currently a live feature in the MicroJam app[2].

### 1.2 EMPI

The Embodied Musical Predictive Interface (EMPI)[3] is a self-contained NIME with a single dimension of continuous input and output through two physical levers, a Raspberry Pi, touchscreen, and speaker. EMPI was designed to explore the simplest predictive musical interactions: where one dimension of input is modelled along with time. The MDRNN model was trained on a 10-minute human performance with the input lever.

In the EMPI we explore multiple modes for positioning the MDRNN: as a "continuator" for call and response performance, as an "opposing" voice in polyphony with the input lever, or as a filter for the input lever's motion. This can be accomplished by different configurations of connection between input lever, RNN input and output, output lever, and synthesis mapping. The EMPI demonstrates that even a Raspberry Pi can be used for predictions from a small MDRNN in a real-time situation. EMPI is a working prototype and work is ongoing to assess its musical possibilities and evaluate its creative value; however, we have found it compelling so far in demo environments.

### 1.3 Future Directions

We think that predictive interaction could be applied widely in computer music software such as DAWs, synthesis environments and physical hardware controllers. One first step towards this is live prediction with ROLI's Lightpad Block (a soft touchscreen), this involves modelling x, y, dt, and pressure data. We have applied our Keras MDN-layer to create an MDRNN model for this interface and are presently integrating this into a computer music application. The success of tools such as Wekinator (Fiebrink, 2017), and interest in Google's Magenta project suggest that artists see the value of applying ML in their work. The flexibility of our MDRNN tools could be ideal for providing predictive interaction possibilities to these users.

[1] `https://github.com/cpmpercussion/keras-mdn-layer`
[2] `https://microjam.info`
[3] `https://github.com/cpmpercussion/empi`

**Acknowledgments**

This work is supported by The Research Council of Norway as part of the Engineering Predictability with Embodied Cognition (EPEC) project #240862.

## References

Bishop, C. M. (1994). Mixture density networks. Technical Report NCRG/97/004, Neural Computing Research Group, Aston University.

Fiebrink, R. (2017). Machine learning as meta-instrument: Human-machine partnerships shaping expressive instrumental creation. In Bovermann, T., de Campo, A., Egermann, H., Hardjowirogo, S.-I., and Weinzierl, S., editors, *Musical Instruments in the 21st Century: Identities, Configurations, Practices*, pages 137–151. Springer Singapore, Singapore.

Graves, A. (2013). Generating Sequences With Recurrent Neural Networks. *ArXiv e-prints*.

Ha, D. (2015). Recurrent net dreams up fake chinese characters in vector format with tensorflow. Blog Post.

Ha, D. and Eck, D. (2017). A neural representation of sketch drawings. *ArXiv e-prints*.

Leman, M., Maes, P.-J., Nijs, L., and Van Dyck, E. (2018). What is embodied music cognition? In Bader, R., editor, *Springer Handbook of Systematic Musicology*, pages 747–760. Springer Berlin Heidelberg, Berlin, Heidelberg.

Martin, C. P. and Torresen, J. (2018). Robojam: A musical mixture density network for collaborative touchscreen interaction. In Liapis, A., Romero Cardalda, J. J., and Ekárt, A., editors, *Computational Intelligence in Music, Sound, Art and Design*, pages 161–176, Cham. Springer International Publishing.

## A Supplementary Material: Keras MDN Layer

Our Keras MDN layer[4] is designed to allow artists, as well as machine learning researchers, to apply MDRNNs to their creative work. We aim to provide a clear implementation of an MDN layer that can be applied to many problems. We have included several examples in the repository, following Bishop (1994), Ha's Kanji model (2015), and our RoboJam model.

### A.1 What is an MDN?

The idea of a mixture density network (MDN), is to use the outputs of a neural network as the parameters of a Gaussian mixture model (GMM), as shown in Figure 2. Such a model "mixes" a number of Gaussian (or normal) distributions with weights corresponding to the likelihood of each of these mixture components. This allows the model to represent phenomena that appear to be drawn from multiple different distributions.

In a creative process such as musical improvisation, multiple choices for the next note to perform or action to take could be artistically valid. This observation suggests that some kind of multi-modal distribution, such as a GMM, would be appropriate to accurately model such a process. An MDN following a typical LSTM-RNN thus forms a useful network for regression problems involving multiple correct answers, or that require a certain amount of stochasticity when sampling, as in creative tasks.

### A.2 Loss Function

One of the complexities of an MDN is the loss function which is derived from the probability density function of the mixture model. While this is straightforward for a 1-dimensional case (as given by Bishop (1994)), mixtures of higher-dimensional Gaussian distributions are more complex. Our MDN

[4] `https://github.com/cpmpercussion/keras-mdn-layer`

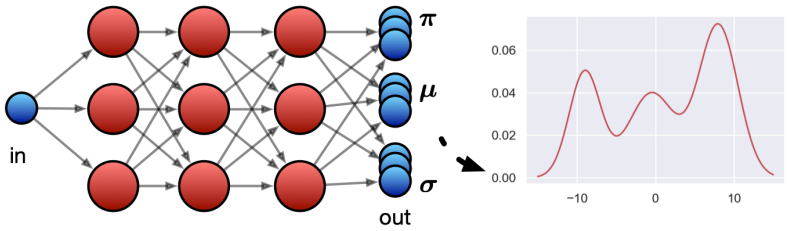


Figure 2: A mixture density layer usess the outputs of a neural network as the parameters of a Gaussian mixture model.

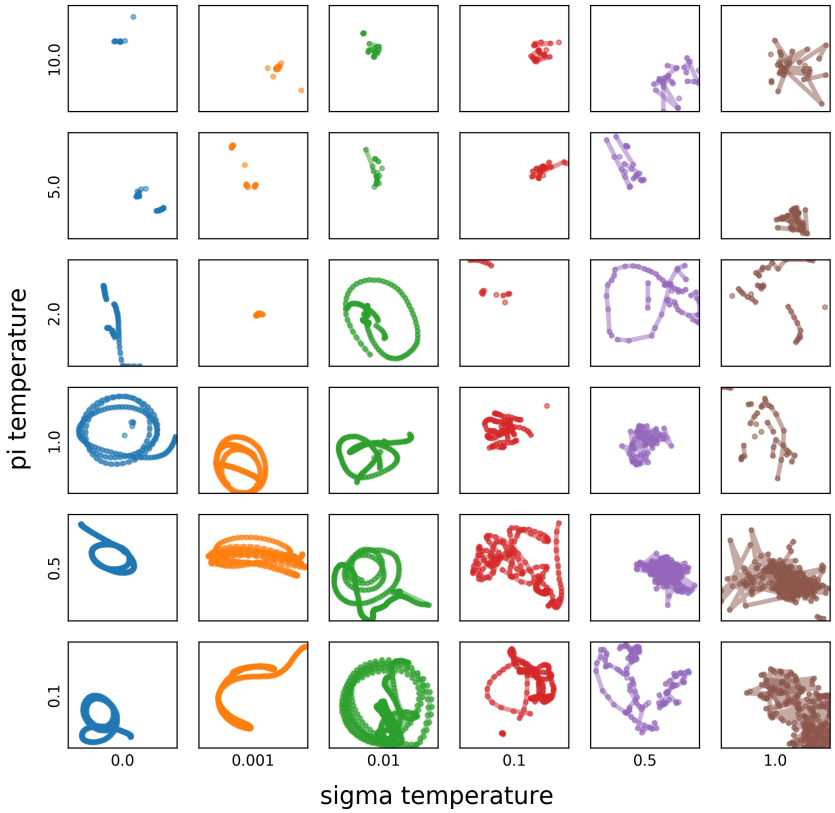


Figure 3: Sampling touchscreen performances from RoboJam's MDRNN at different temperatures for the categorical and Guassian parts of the mixture model.

layer uses a factored multivariate Gaussian distribution for each mixture component. That is, each mixture component is a Gaussian distribution with a diagonal covariance matrix. This means that the loss function is much more manageable, and can be calculated using TensorFlow Probability's Mixture, Categorical, and MultivariateNormalDiag functions. The loss function for an MDN relies on the number of mixture components as well of the dimension of each component, so we provide a function to generate the correct loss function on demand for these parameters.

### A.3 Sampling

The output from an MDN is the set of parameters for the GMM: the weights for each mixture component ($\pi$s), and the means and standard deviations for each multivariate Gaussian ($\mu$s and $\sigma$s). Predictions must be generated from these parameters by sampling from the categorical (softmax)

distribution formed by the $\pi$s, and then sampling from the Gaussian distribution chosen by that outcome.

The concept of adjusting the temperature of a categorical distribution will be familiar to those who have used RNNs to learn creative sequences such as text or symbolic music. The MDN's categorical distribution can be adjusted in the same way with very low temperature values favouring the maximum value in the distribution, and high values producing a more uniform distribution. This adjustment could be called "$\pi$-temperature". The temperature of the Gaussian distributions can also be adjusted by scaling the standard deviation. High values result in a wider spread of predictions, and low values are closer to the selected mean. We call this "$\sigma$-temperature".

In RoboJam, we have found adjusting the $\sigma$- and $\pi$-temperature to be very important for making useful predictions. In Figure A.3 we show unconditioned touchscreen performances sampled at different temperatures. Our MDRNN tends to produce high standard deviations, resulting in jagged output, but by reducing the $\sigma$-temperature to close to zero, we can generate smooth paths. In production, we have left the $\sigma$-temperature at $0.01$ to allow for some unpredictable variation in the predicted responses.

We can use the $\pi$-temperature to control the appearance of different swipes and taps to some extent. At low $\pi$-temperature values, an MDRNN will have trouble changing modes. For RoboJam, this means it will continue one swirl and never start again somewhere else on the screen. At very high $\pi$-temperature, RoboJam taps without completing any significant swipes. Exploring how $\pi$ and $\sigma$ sampling temperature can be applied in our predictive interaction systems is a topic of our future research.