Machine and Deep Learning for Pirouette Movement Recognition

Blase Londono
Mechanical and Aerospace Engineering Department
University of California, San Diego
La Jolla, California, United States
blondono@ucsd.edu

***Abstract*— In this paper, a custom Long Short Term Memory Deep Learning Neural Network was built using the MATLAB Deep Learning Toolbox. This Deep Learning Neural Network was tasked with recognizing a specific ballet dance move, a pirouette, given only the dancers' physical properties (height, weight) and a dataset of inertial readings taken from an Inertial Measurement Unit device strapped to the dancers' waists. The Deep Learning Neural Network was able to classify the dancers' pirouettes with over 95% accuracy within 100 iterations. The Deep Learning Neural Network was then instructed to move a 3-dimensional mesh of the dancers based on its perceived analysis of the dancers' movements. The results show that the custom Long Short Term Memory Model was capable of classifying a complex multiphase movement (pirouette) based on limited dataset information and limited epochal span.**

Keywords*— *Neuron; Deep Learning; Analysis; Classification; Processing; Composition.

I. Introduction

With the advent of new technologies, sports performance analysis has taken considerable steps towards the personalization and accuracy of athlete care [1]. This change is due in part to the advancement of deep and machine learning networks [7]. Ballet, being both a sport and a cultural expression of art to many, is subject to this advancement as well [1, 3-5].

Deep Learning Neural Networks (DLNNs) are complex, layered programming structures composed of webs of many smaller, simpler processing thresholds called neurons [7]. These neurons are typically comprised of a mathematical system of many smaller weighted and biased functions which take inputs from other neurons or their environment (for example, a set of data) (Fig. 1) [6]. These neurons may shift their weights and biases across multiple iterations, or epochs, of the network's runtime [6,7]. To "train", or optimize, DLNNs to conform their outputs to a specific task, users must feed both the performance of the network, given by the "loss," or inaccuracy of the network's previous iteration, back to the network for tweaking [6,7]. DLNNs then take this data, and adjust their weights and biases to minimize the error function's "loss" [6].

DLNNs have become more popular in recent years for their ability to conquer tasks normally very difficult, if not impossible, for lower-level Machine Learning (ML) algorithms to solve [7]. Examples of these "difficult tasks" include image recognition [6], 3-body orbital prediction and higher-level physics simulations [6], and pattern recognition and connections between often-abstract or exotic input data [1, 3-7].

This study will touch on all three of these capabilities of DLNN algorithms. First, a large dataset of physics properties of four separate dancers will be fed to a DLNN as they perform multiple different ballet moves. The DLNN will then process the raw physics data into readable components, then attempt to classify which of the dancer's movements are a specific dance move– in this case, a pirouette– based on the refined physics data blocks. Finally, the DLNN will manipulate a 3D construct of the dancer to behave with its interpretation of a pirouette built from its analysis of the dancer datasets. The purpose of this study is to demonstrate the robustness of a multipurpose multilayer DLNN model, as well as some of the possible challenges that may present themselves when dealing with complex algorithms of this nature.

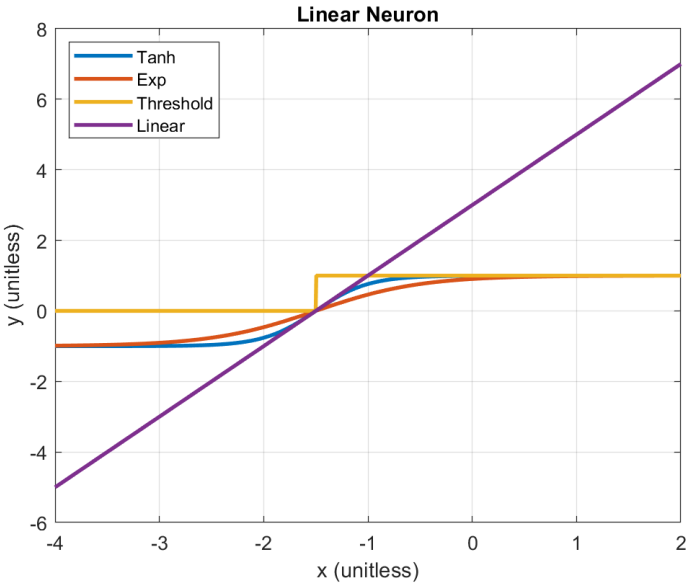

FIGURE 1. Example of a Linear Neuron Structure [6]

II. Methods

A. Physics Analysis

Before the DLNN model is able to begin classifying the different dancers' movements, it must first refine a large amount of data collected by the Inertial Measurement Unit (IMU) that was previously attached to each of the dancers throughout their performances (Fig. 2). This filtration of data is an important part of distinguishing dancer movements. The DLNN first begins by refining the IMU inertial data through a series of calculations predefined by the program technicians.

The first of these series of calculations determines the rotational torque experienced by the dancers as they undergo each of their classified dance movements. We can represent torque as a function of the inertia readings given by the IMU using Euler's equation [6]

$$\tau = I\dot{\omega} + \omega^{\times} I\omega$$

TABLE 1. Rotational Motion Variables

Variable	Representation
τ	*External Torque*
I	*Inertia*
ω	*Angular Rate*

Euler's equation here does not represent a single-dimensional torque reading [6]. Instead, properties of matrix multiplication can be implemented here to split the total external torque into its 3-dimensional components [6]. Below, we will first split the torque and angular rate variables into 3-dimensional matrices,

$$\tau = \begin{bmatrix} \tau_x \\ \tau_y \\ \tau_z \end{bmatrix}$$

$$\omega = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}$$

Then use the aforementioned properties of matrix multiplication in our original rotational motion equation to expand it to account for all dimensions.

$$\begin{bmatrix} \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{xy} & I_{yy} & I_{yz} \\ I_{xz} & I_{yz} & I_{zz} \end{bmatrix} \begin{bmatrix} \dot{\omega_x} \\ \dot{\omega_y} \\ \dot{\omega_z} \end{bmatrix}$$

We can use this general-form equation to then isolate the z-axis torque.

$$\tau_z = I_{xz}\dot{\omega_x} + I_{yz}\dot{\omega_y} + I_{zz}\dot{\omega_z}$$

Notably, the equation for any single dimensional torque within our expanded rotational motion equation can be represented by the corresponding row of the inertia matrix times the angular rate vector. Therefore, we can simplify our expanded multidimensional rotation equation:

$$\begin{bmatrix} \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} = \begin{bmatrix} I_{xz} \\ I_{yz} \\ I_{zz} \end{bmatrix} \dot{\omega_z}$$

Unfortunately, this equation still does not account for the separate torque experienced by the dancer's head as they "spot" during the turn, nor does it account for the displacement of the IMU from the dancer's center of mass (Fig. 3). We can model these discrepancies in our original equation using a couple extra terms:

$$\tau = I\dot{\omega} + \omega^{\times}[I\omega + uI_i(\dot{\delta_i} + \omega_z) + uI_h(\dot{\delta_h} + \omega_z) + u(\tau_i + \tau_h)]$$

Where,

$$\tau_i = I_i(\dot{\delta_i} + \dot{\omega_z})$$

Where,

$$\tau_h = I_h(\dot{\delta_h} + \dot{\omega_z})$$

And where,

$$F = m\ddot{z}$$

TABLE 2. Expanded Equation Variables

Variable	Representation
F	*Vertical Force*
m	*Mass*
z	*Vertical Direction*
I_i	*Internal Inertia (Control)*
I_h	*External Inertia (Spotting)*
δ_i	*Internal Moment (Resistance)*
δ_h	*External Moment (Resistance)*
u	*Z-Axis Unit Vector*

Together, these equations give the DLNN enough information to distinguish between the dancers' movesets.

B. Model GUI

After utilizing the data refined in the previous section, the DLNN will then move a 3D model of a ballet dancer mesh in order to illustrate what it believes the dancers' movements to look like. This model utilizes quaternion operations to represent the orientation of our dancer mesh within the 3D environment.

Quaternions save our DLNN resources because quaternion multiplication of the 3xN vector of our dancer's 3-dimensional physics variables is quicker and more efficient than transforming each property vector individually. We can utilize this feature of quaternions in MATLAB using a simple function (Fig. 10). With the increased computational efficiency granted by the quaternion functions in Fig. 10, our DLNN network can begin rigging the dancers' mesh to the appropriate movements. This is done in real-time, while the network is still running. Most programs run their simulations after their calculations have been completed, which allows for quick runtimes of data analysis (leaving the visualizations of the data as secondary tasks) [6]. However, due to the robustness of the DLNN software, this otherwise resource-intensive task can be done simultaneously with the aforementioned physics analysis and the categorization of the dancer movesets.

C. Dancer NN Structure

The Dancer DLNN was structured around a Long Short-Term Memory (LSTM) neural network skeleton. This structure was chosen because LSTM networks are better equipped at solving time-dependent data classification than other neural network models [6,7].

The LSTM structure within the higher neural network is given the refined physics dataset, broken down into six-second ranges (approximately the average span of a single- or double-pirouette, plus a buffer at the beginning and the end of the data collection). To account for human error in the dataset, the LSTM automatically prunes datasets that fall outside of the specified six-second range (Fig. 9).

In order for the bidirectional LSTM structure to output the correct reading for the dancers' pirouettes, the LTSM skeleton was tweaked slightly to allow for the 10 separate features it had to account for. These features included:

- Four Quaternion Measurements (Fig. 10).
- Three Rate Gyroscopic Measurements (Fig.4-6, Dataset).
- Three Accelerometer Measurements.

Where the four quaternion measurements were coupled together through the relationship:

$$1 = q_1^2 + q_2^2 + q_3^2 + q_4^2$$

The couple, while necessary, does notably slow down learning speed as it incurs a recursive computation within the DLNN's classification layers. This does not affect the DLNN's accuracy. Furthermore, we must tweak the LSTM DLNN output mode from the default 'first' to 'last', to allow for correct output after the GUI has been initialized (Fig. 8). The appended neural network flow skeleton can be found in Fig. 9.

FIGURE 2. Moveset Dancer Images [6]

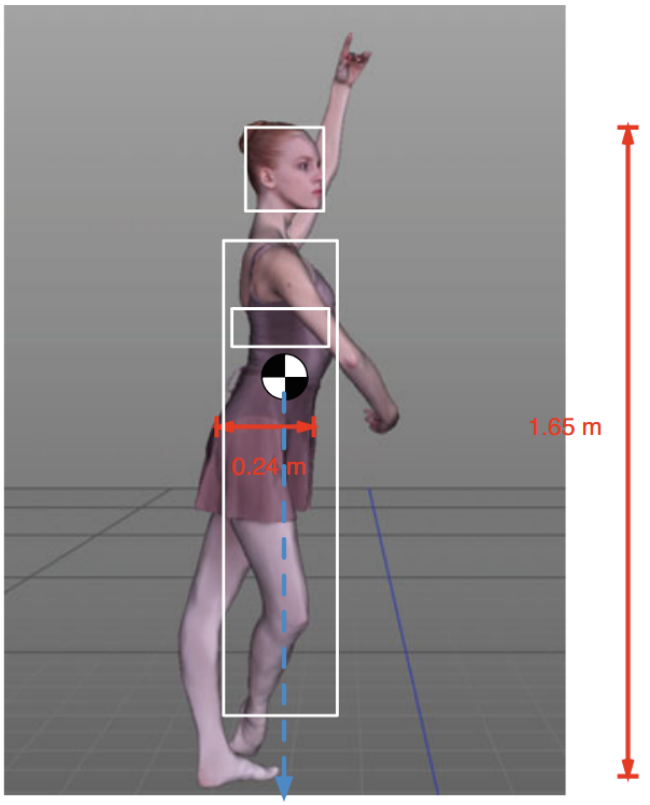

FIGURE 3. Ballerina IMU Displacement [6]

III. Results

The actual DLNN structure for this study was created through a mixture of many properties of classical neural network architecture. While the amount of neurons employed by this network was significantly smaller than the amount typical for larger, more robust neural network models (such as a Large Language Model (LLM)), it proved capable of categorizing the dancers based on their data and fit within the constraints of the hardware the MATLAB platform was run on (in this study, a low-level Framework laptop).

Prior to the DLNN LSTM skeleton call, the network was able to classify the raw inertial dataset from the dancers' IMU into multiple smaller components. Figure 4 shows the sample output of one dancers' IMU readings into multiple components, in this case the dimensional rotational rate and dimensional acceleration. These readings, coupled with the information given about each dancer (height, mass, etc), could then be used to predict other forces acting upon the dancers, such as the z-axis force and split torques acting upon the dancers' bodies while 'spotting' their pirouettes (Fig. 5). This data was used to build a quaternion-oriented GUI model of the dancer which the DLNN model was able to 'puppet'. (Fig. 6, 7).

This study ran the DLNN frame across 100 iterations, with a learning rate of 0.001 (impact for performance gradient). As a result, the DLNN model was able to take the filtered physics IMU data and classify which of the dancers' movements were pirouettes with over 95% accuracy towards the later epochs (Fig. 8).

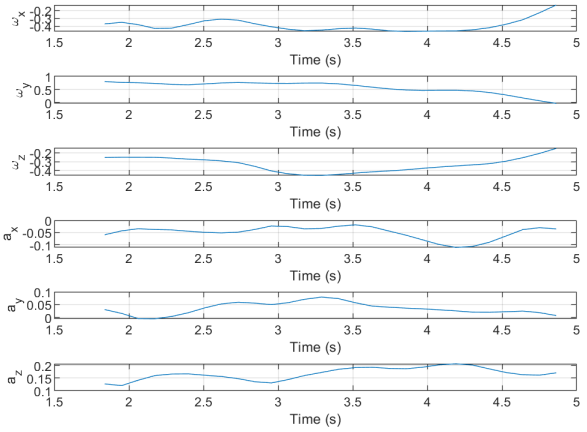

FIGURE 4. Sample Raw Ballerina Dataset

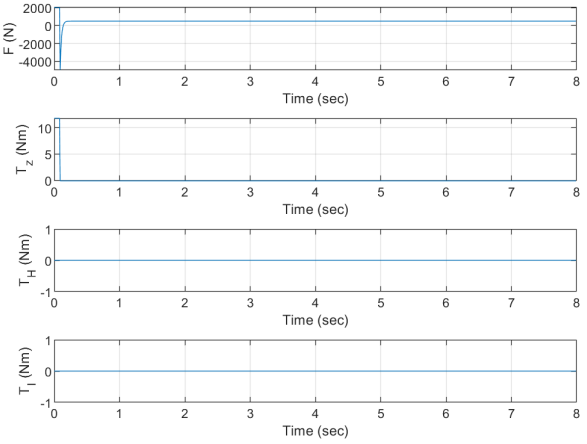

FIGURE 5. Dancer Torque Readout

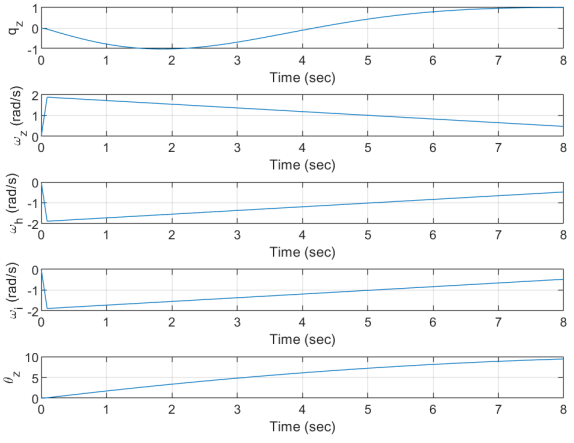

FIGURE 6. DLNN Network Simulated Rotation Analysis

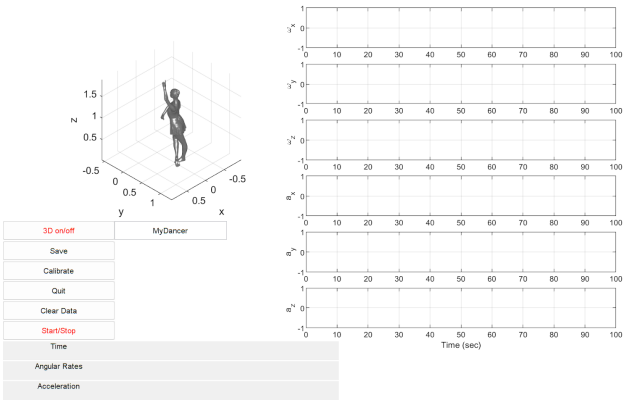

FIGURE 7. Dancer Model Simulation GUI

G

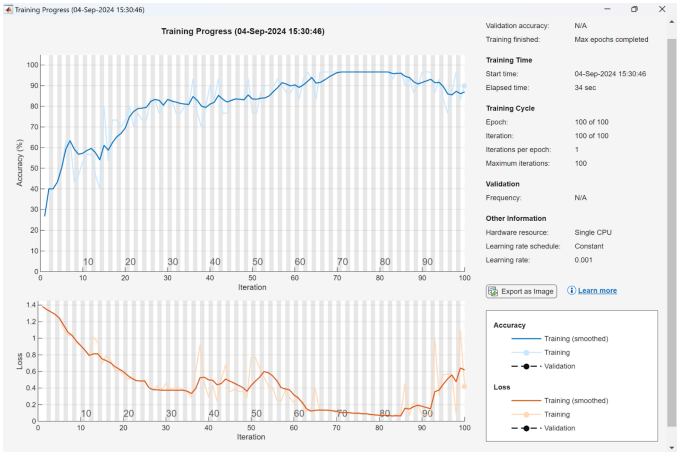

FIGURE 8. Neural Network GUI Output

```
>> DancerNN
36 remaining data sets out of 40 total.
  Ryoko data sets 6
  Shaye data sets 10
  Emily data sets 10
Matanya data sets 10
  5×1 Layer array with layers:

     1   ''   Sequence Input          Sequence input with 10 dimensions
     2   ''   BiLSTM                 BiLSTM with 400 hidden units
     3   ''   Fully Connected        4 fully connected layer
     4   ''   Softmax                softmax
     5   ''   Classification Output   crossentropyex
```

FIGURE 9. DLNN Command Window Output (Classification)

QuaternionToMatrix.m

```
function m = QuaternionToMatrix( q )

m = [ q(1)^2+q(2)^2-q(3)^2-q(4)^2,...
      2*(q(2)*q(3)-q(1)*q(4)),...
      2*(q(2)*q(4)+q(1)*q(3));...
      2*(q(2)*q(3)+q(1)*q(4)),...
      q(1)^2-q(2)^2+q(3)^2-q(4)^2,...
      2*(q(3)*q(4)-q(1)*q(2));...
      2*(q(2)*q(4)-q(1)*q(3)),...
      2*(q(3)*q(4)+q(1)*q(2)),...
      q(1)^2-q(2)^2-q(3)^2+q(4)^2];

```

FIGURE 10. Sublevel Quaternion Transformation Function

IV. Conclusion

This study used a custom-build Deep Learning Neural Network algorithm to distinguish between different ballet movements performed by four separate dancers. The DLNN algorithm was able to accomplish this task through a multilayered data analysis approach, where the large general inertial dataset was first refined, then classified, then inputted into the actual learning neural web for processing.

The dataset was refined through multiple physics algorithms which turned the rough, general-form inertial readings from the IMU attached to each of the dancers, and divided them into multiple smaller categories which the DLNN learned to separate based on the dancers' given attributes (weight, height, size, center of mass, etc). This allowed the DLNN to classify the dancers' movements.

Classification was accomplished through a modified multi layered Long Short Term Memory neural network skeleton. The LSTM section of the Dancer DLNN was able to sort through the time-dependent physics dataset (see previous paragraph), and from that data was able to classify the dancers' pirouette movements with over 95% accuracy within 100 iterations.

In the end, the LTSM DLNN was able to categorize the movesets of four separate ballet dancers and isolate which of their movements were specified as pirouettes, purely from the constant dancer data fed to it (the weight, height, and shape of the dancers) and the raw inertial reading dataset from the IMU. This success proves the robustness of the DLNN in this study, as well as the usefulness of Deep Learning and Machine Learning in otherwise-uncommon applications within the scientific field. Furthermore, the success of the DLNN model shows its usefulness in the application of personalized sports improvement technologies as referenced in the introduction [1,7].

Acknowledgment

The author would like to acknowledge the support of his MAE08 instructor, Phuong Truong, as well as the teaching assistant Devyansh Agrawal, for their unwavering support throughout the span of the course. The author would also like

to thank his family and friends for their efforts and encouragement throughout the writing of this paper.

References

[1] E. E. Cust, A. J. Sweeting, K. Ball, And S. Robertson, "Machine And Deep Learning For Sport-Specific Movement Recognition: A Systematic Review Of Model Development And Performance," Journal Of Sports Sciences, Vol. 37, No. 5, Pp. 568–600, 2018. [Online]. Available: Https://Doi.Org/10.1080/02640414.2018.1521769.

[2] A. Mohanty, P. Vaishnavi, P. Jana, A. Majumdar, A. Ahmed, T. Goswami, And R. R. Sahay, "Nrityabodha: Towards Understanding Indian Classical Dance Using A Deep Learning Approach," Signal Processing: Image Communication, Vol. 47, Pp. 529-548, 2016. [Online]. Available: Https://Doi.Org/10.1016/J.Image.2016.05.019.

[3] H. Matsuyama, S. Aoki, T. Yonezawa, K. Hiroi, K. Kaji, And N. Kawaguchi, "Deep Learning For Ballroom Dance Recognition: A Temporal And Trajectory-Aware Classification Model With Three-Dimensional Pose Estimation And Wearable Sensing," Ieee Sensors Journal, Vol. 21, No. 22, Pp. 25437-25448, Nov. 15, 2021, Doi: 10.1109/Jsen.2021.3098744.

[4] X. Liu And K. Y. Chun, "The Use Of Deep Learning Technology In Dance Movement Generation," Frontiers In Neurorobotics, Vol. 16, 2022. [Online]. Available: Https://Www.Frontiersin.Org/Articles/10.3389/Fnbot.2022.911469.

[5] S. Karumuri, R. Niewiadomski, G. Volpe, And A. Camurri, "From Motions To Emotions: Classification Of Affect From Dance Movements Using Deep Learning," In Proc. Of The Acm Symposium On Applied Perception, 2019, Pp. 1-8, Isbn 9781450359719. [Online]. Available: Https://Doi.Org/10.1145/3290607.3312910.

[6] M. Paluszek, S. Thomas, And E. Ham, *Practical Matlab Deep Learning: A Projects-Based Approach*, Apress, 2022. [Online]. Available: Https://Drive.Google.Com/File/D/1mnixkrzxljyoxwkiwk7vfxyr8s_9qemi/View

[7] J. Schmidhuber, "Deep Learning In Neural Networks: An Overview," Neural Networks, Vol. 61, Pp. 85-117, 2015. [Online]. Available: Https://Doi.Org/10.1016/J.Neunet.2014.09.003.