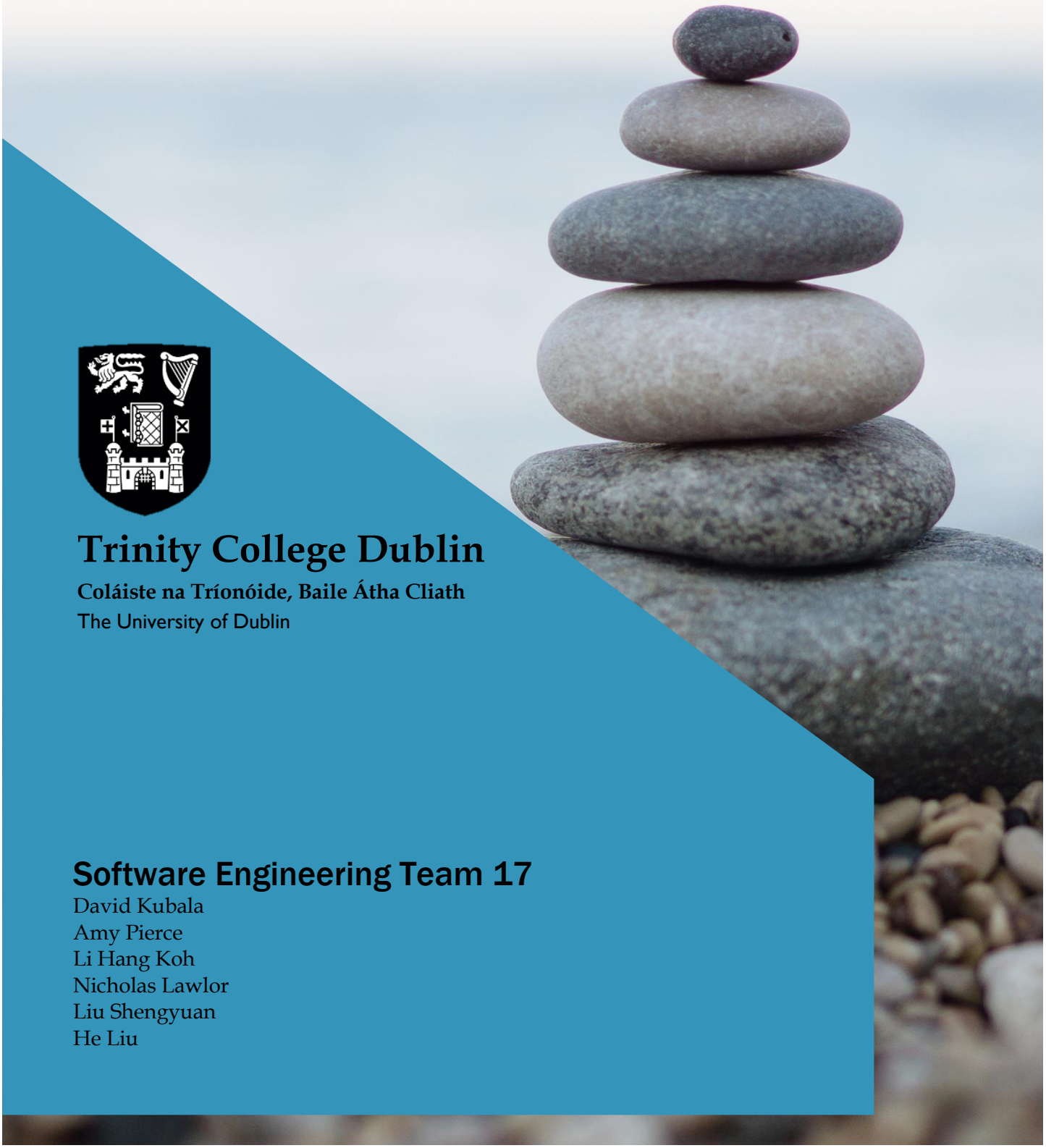
MCAM v2.1

Motion Capture Analysis Module

Technical Specifications Document



Trinity College Dublin

Coláiste na Tríonóide, Baile Átha Cliath

The University of Dublin

Software Engineering Team 17

David Kubala
Amy Pierce
Li Hang Koh
Nicholas Lawlor
Liu Shengyuan
He Liu

Contents

DATA PARSING

Prepping the Data

The first task in producing this model was to parse the data into a format that could be read by the machine learning (ML) model. A question that needed to be made early in the process was "what will our machine learning model classify?". There were several challenges the team faced regarding this question. We began with a total of ~350 megabytes of data. In the context of an ML problem, that is very little data with which to train a model. We had about 850 motions to start with, which would not be enough to train a classification model.

Instead of classifying full motions, we instead opted to classify individual frames. Instead of the model answering the much more complex question "What motion is this?", it would be trained to answer the simpler question "What motion is this frame most likely a part of?".

One of the initial difficulties faced was that the model would need to be fed data from a non-jagged array in order to be read properly. Due to this property in Tensorflow, each element in the frame array would need to be the same size, which led to adding padded 0's in the frame. Additionally, we began by training the model with both the positional and rotational data of each joint, but eventually switched to just using the position data. This tended to confuse the model less frequently with testing.

In summary, each motion was parsed into a collection of frames with positional tracking data for each joint.

```
0 : [0.006, 1.0, 0.021, 0.568, 0.076]
1 : [-26.925, 61.803, 231.953, 0.0, 2.0]
2 : [-25.888, 50.906, 229.653, 0.0, 2.0]
3 : [-35.594, 39.252, 231.184, 0.0, 2.0]
4 : [-13.275, 40.202, 228.826, 0.0, 2.0]
5 : [-41.147, 27.103, 238.174, 0.0, 2.0]
6 : [-2.371, 28.871, 230.304, 0.0, 2.0]
7 : [-52.368, 20.774, 232.222, 0.0, 2.0]
8 : [2.417, 28.092, 218.835, 0.0, 2.0]
9 : [-55.255, 20.474, 230.705, 0.0, 0.0]
10 : [2.492, 30.464, 218.053, 0.0, 2.0]
11 : [-24.954, 44.97, 231.609, 0.0, 2.0]
12 : [-22.023, 26.77, 236.721, 0.0, 2.0]
13 : [-17.979, 1.978, 242.757, 0.0, 2.0]
14 : [-24.764, -0.022, 239.49, 0.0, 2.0]
15 : [-10.794, 3.938, 239.838, 0.0, 2.0]
16 : [-20.841, -23.199, 237.727, 0.0, 2.0]
17 : [-7.688, -11.924, 220.608, 0.0, 2.0]
18 : [-20.871, -52.935, 241.014, 0.0, 2.0]
19 : [2.556, -36.001, 245.777, 0.0, 0.0]
```

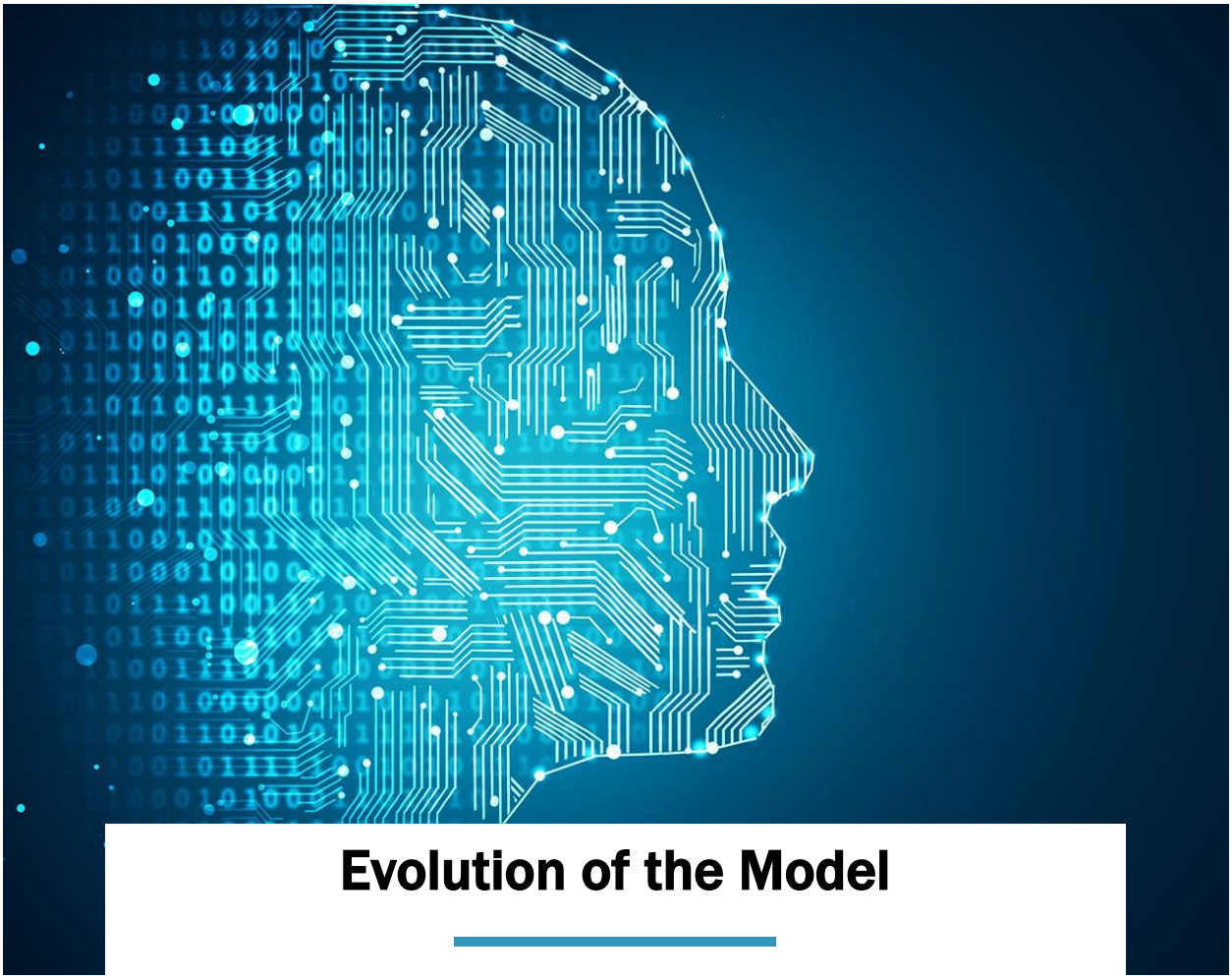
Structure of a Frame

Each number references a different index of the frame array data. Each line contains 5 float values.

```
0 : [<ground tracking data>]
1 : [<Head position X>, <Y>, <Z>,
<Padded 0>, <Tracking Status>]
2 : [<Neck position X>, <Y>, < Z>,
<Padded 0>, <Tracking Status>]
3 : ...
18 : [<Foot Right position X>, <Y>,
<Z>, <Padded 0>, <Tracking Status>]
```

Order of Joints

1. Head
2. Neck
3. Shoulder Left
4. Shoulder Right
5. Elbow Left
6. Elbow Right
7. Wrist Left
8. Wrist Right
9. Hand Left
10. Hand Right
11. Spine Shoulder
12. Spine Mid
13. Spine Base
14. Hip Left
15. Hip Right
16. Knee Left
17. Knee Right
18. Food Left
19. Foot Right



Evolution of the Model

Over the course of this project, our Machine Learning model went through several interesting iterations based on rearranging the neural network's structure, removing or adding data, data augmentation, changing how long the model was trained on the dataset, and working to lower the amount of misclassified data.

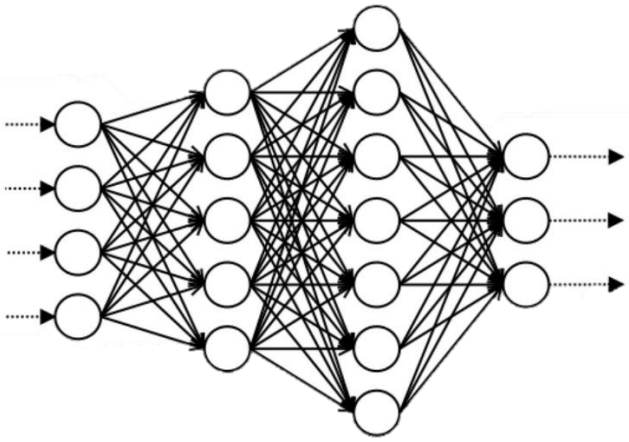
The finalized model predicts a given input motion with ~94% accuracy. Most falsely classified data, when checked, tend to be poorly tracked motion capture or ambiguously confusing (e.g. misclassifying a JumpForward as JumpHigh because the user did not jump forward a certain distance).

BASIC CLASSIFICATION MODEL

Structure

The model was structured very simply. Based on an input frame, the model would predict which one of the 6 motions it was most likely to be a part of: HopLeft, HopRight, JogSpot, JumpSide, JumpHigh, or JumpForward.

The neural network itself, at first, used a 100-dimensional input layer with 6 hidden layers of various sizes. The neurons in the hidden layers of the network used a Rectified Linear Unit (ReLU) activation with no active dropout. The final layer of the network (the output layer) uses a Softmax activation function. This neural network structure stuck early in the project and worked well throughout. There was not much need to update this structure as the project progressed.



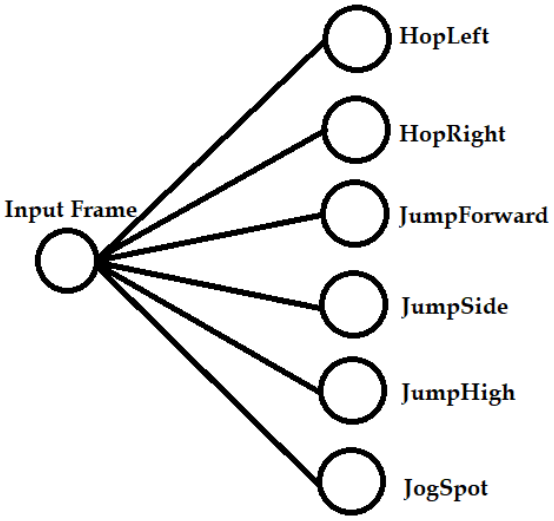
Sample : Simple Neural Network Structure
4-Dimensional input vector
2 Hidden Layers
3-Dimensional output vector

Data Flattening

In order to properly feed the data into the input layer of the Machine Learning Model, the data first had to be "flattened", or taken from a 2-dimensional array of floats to a nth dimensional vector. The structure of the array initially was a 20x5 array, meaning that it would need to be turned into a 100-dimensional vector. The output vector was a 6-dimensional vector, which output how confident the model was about which motion the frame most likely represented.

Motion Prediction

The motion overall was predicted by feeding each individual frame of a motion into the model. The overall decision for which motion the file represented was based on which motion was most frequently over all the frames.

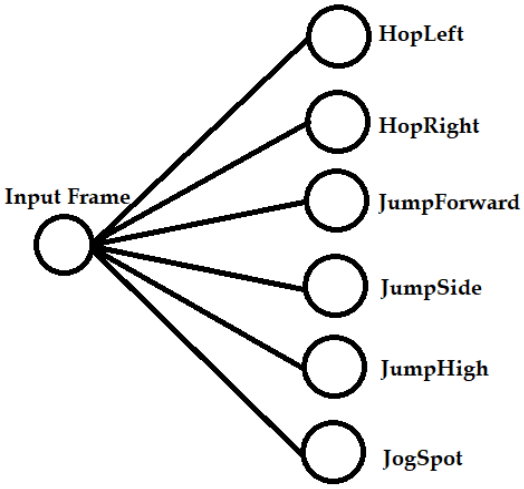


Basic Model Decision Tree

RESTRUCTURED MODEL

Issues with Basic Classification

Though the Basic Classification model worked nicely and cleanly, it only was predicting the given motion at about 80% accuracy. Thus, we began experimenting with different model structures.

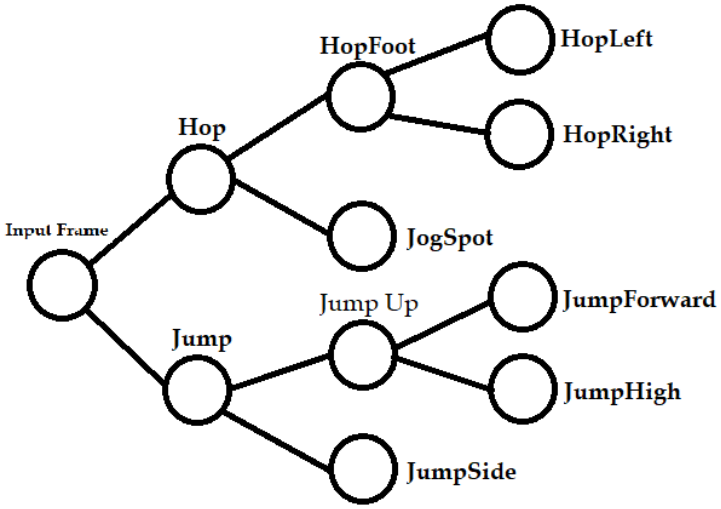


Original Model

Proposed Models

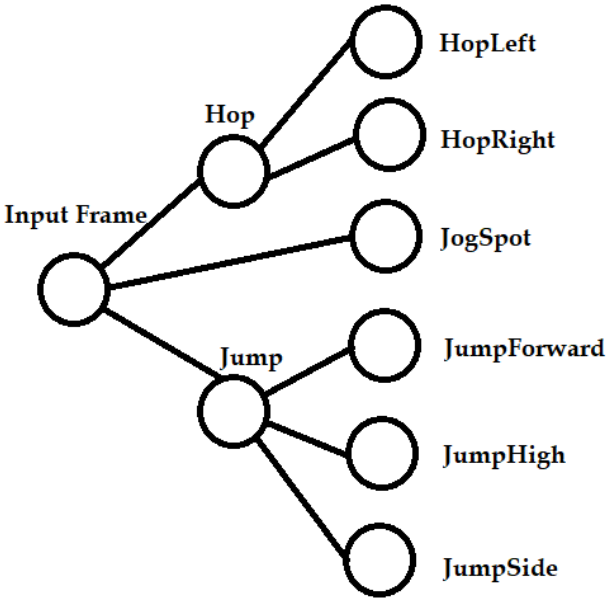
The newly proposed models are dependent on two or more secondary ML models that would narrow the decision further and remove some burden from the first layer to predict accurately.

The theory behind Proposed Model 1 was to use a binary crossentropy model rather than a sparse categorical crossentropy model for each layer. However, this introduced a host of issues. 3 decision layers meant that there was a chance of incorrect guesses in each layer, lowering the model's overall accuracy. Additionally, the model had trouble discerning between Hops and Jumps on the first layer, which broke down the functionality of the other layers.



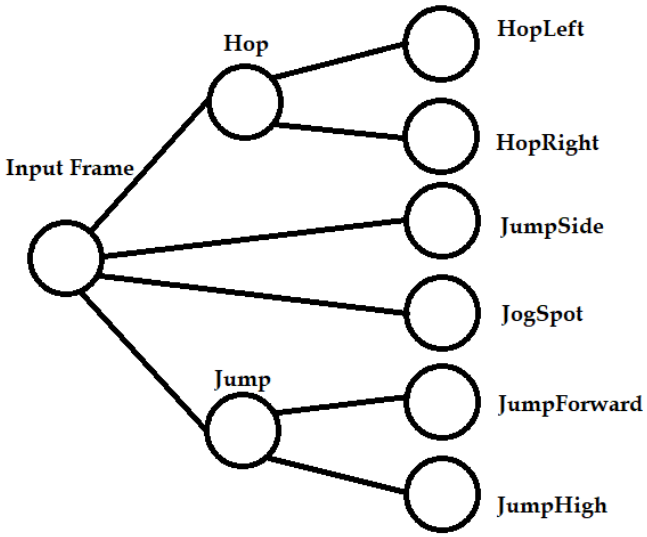
Proposed Model 1

Proposed Model 2 switched back to sparse categorical crossentropy for each model, but still had issues making direct distinction between Hops and Jumps. The binary crossentropy model that was being used ended up being less accurate than the sparse categorical crossentropy model when the ReLU activation function was used. Fortunately, this structure began to progress to the finalized structure, taking into account the confusion matrix produced by Model 1.



Proposed Model 2

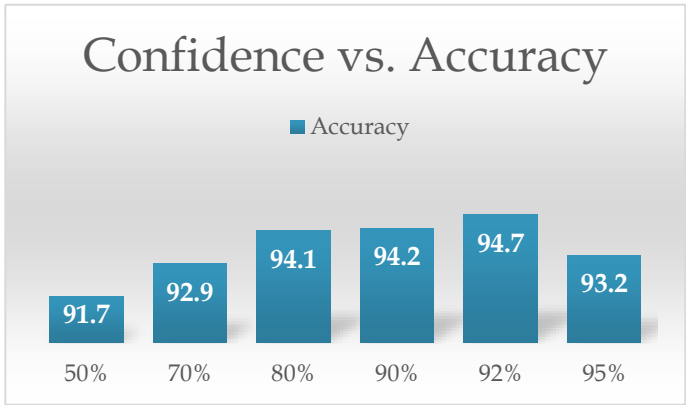
The finalized model took motions that were most often confused for one another and used two additional machine learning models that were specialized to differentiate between them. Using this structure boosted the overall accuracy of the final product from ~85% to around ~92%.



Finalized Model

Confidence Checking

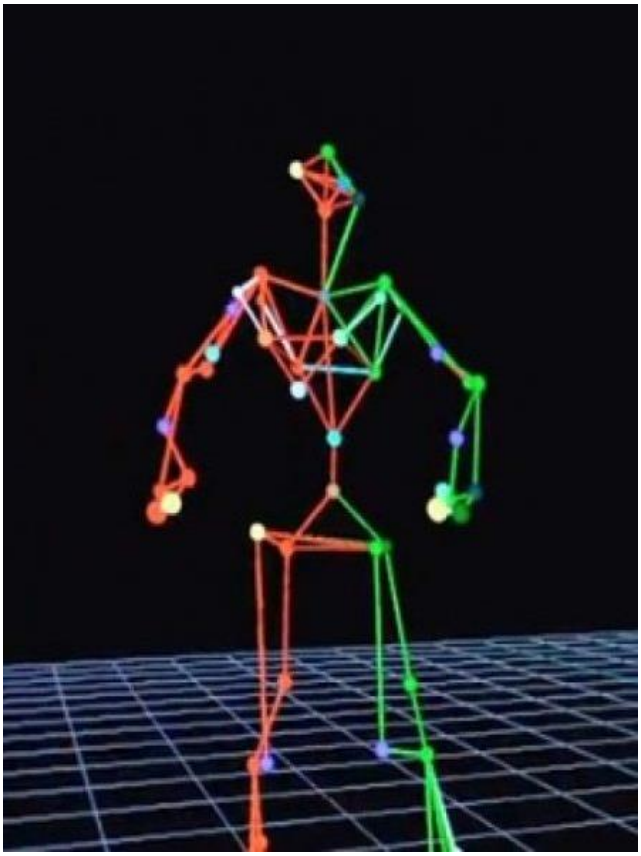
In order to boost accuracy further and remove some ambiguity from the decisions the model makes, only frames that are guessed with above 92% confidence are considered in the final calculation. Some testing was done with the value of confidence percentage, and 92% was found to do best against the test data.



DATA AUGMENTATION

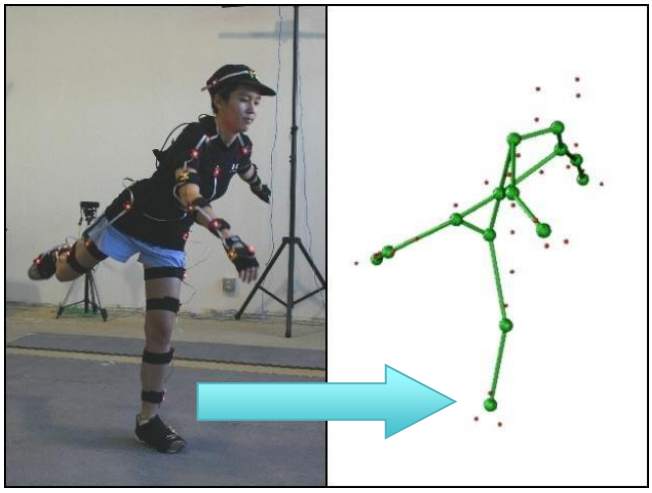
Augmenting the Data

There were many options we explored to potentially expand our relatively small training dataset. Short of procedurally generating motion capture files, a costly and inefficient procedure, we had very limited options. We settled on augmenting our dataset to train our model.



Option 1 : Jitter

The first option was to introduce random jitter into each joint. The joint's X, Y, and Z positions would be offset by a random relatively insignificant value, as to introduce a touch of randomness into the data, without affecting how the machine learning models read the data to a significant degree. This plan, unfortunately, led to unforeseen complications. By augmenting the data in this fashion, the finalized results became too markedly different from the original dataset for the model to read accurately and train properly.



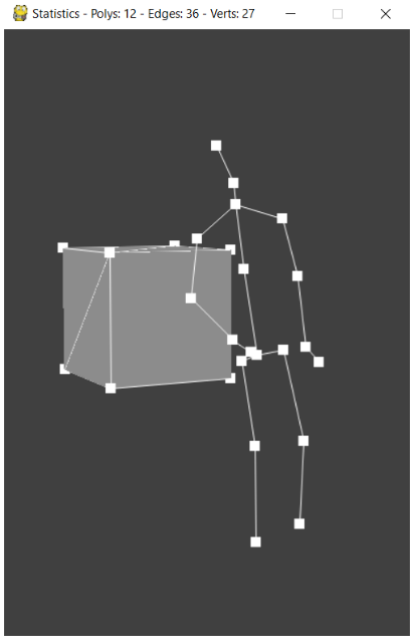
Option 2 : Shifting

The second option was significantly more effective, and much simpler to implement. By simply adding 1 to every joint's X coordinate, the full skeleton was shifted over by 1 unit. This led to drastically better results; the data of the original file was still maintained, while looking "different " enough to effectively double our dataset. This allowed the machine learning models to train adequately without losing any of the original data.

3D Graphics and Visualization

A small (though not insignificant) bonus was added to this project near the end of production. Python did not have a workable 3D graphics package for this project, yet we needed an easy, quick way to visualize the motion capture files to determine how well the machine learning model was working to predict the given motions.

Therefore, a graphics package was developed that would be able to visualize the motion capture file using perspective projection. This was built custom for this project by David Kubala from scratch with Pygame to render the results. Once we had the graphics package working for our debugging purposes, we decided that it would be easily implemented into the frontend for a user to visualize motion capture files, and thus it was included in the form of a "Show Animation" button in the frontend.



Conclusion

The finalized model predicts with ~94% accuracy.

Should anyone want to pick back up on this project to attempt to improve the model, it should be easily expandable. Given more data and innovative augmentation techniques, these models can be additionally trained and even fully replaced in the GUI.

If one were to have any questions about the models or the functionality of this project overall, please don't hesitate to email **kubalad@tcd.ie** for more information.

Thank you so much for including us on this project, we hope this project will assist with your research.

Sincerely,

David Kubala
Amy Pierce
Li Hang Koh
Nicholas Lawlor
Liu Shengyuan
He Liu