**Experiments and Results Discussion**

Note that the experiment number in the table below corresponds to the same experiment number in ‘RNN Results.xlsx’. Hence, for more information about a single row in the table below (e.g. RNN model parameters used, the scripts and exact arguments used to obtain the results, etc.), please see the row in ‘RNN Results’ corresponding to the same experiment number. Note that ‘Data Shape’ is the total shape of the ‘x’ data that is used to train the model and includes both the training and testing components. Also note the different parts of the shape: the first is the number of samples, the second number is the sequence length, and the third is the number of features of each sequence.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Exper Number** | **Source Dir** | **Measurement** | **Output Type** | **Subject List** | **Data Shape** | **Results** |
| 1 | NSAA | Extracted statistical values | D/HC classify | 1 | (742, 10, 30) | Test Accuracy = 92.97% |
| 2 | NSAA | Extracted statistical values | Overall NSAA regress | 1 | (742, 10, 30) | MSE = 28.71, MAE = 2.90 |
| 3 | 6minutewalk-matfiles | Joint angles (from data cube) | D/HC classify | 2 | (8470, 60, 66) | Test Accuracy = 99.88% |
| 4 | 6minutewalk-matfiles | Joint angles (from data cube) | Overall NSAA regress | 2 | (8470, 60, 66) | MSE = 0.476, MAE = 0.403 |
| 5 | 6minutewalk-matfiles | Joint angles (from joint angle files) | D/HC classify | 3 | (2143, 60, 66) | Test Accuracy = 100.0% |
| 6 | 6minutewalk-matfiles | Joint angles (from joint angle files) | Overall NSAA regress | 3 | (2143, 60, 66) | MSE = 0.167, MAE = 0.292 |
| 7 | 6minutewalk-matfiles | Extracted statistical values | D/HC classify | 4 | (552, 10, 30) | Test Accuracy = 82.81% |
| 8 | 6minutewalk-matfiles | Extracted statistical values | Overall NSAA regress | 4 | (552, 10, 30) | MSE = 29.41, MAE = 3.56 |
| 9 | NSAA | Extracted statistical values | Single act classify | 1 | (742, 10, 30) | Ind Act Acc = 92.92%, All Act Acc = 79.69% |
| 10 | 6minutewalk-matfiles | Joint angles (from data cube) | Single act classify | 2 | (8470, 60, 66) | Ind Act Acc = 99.77%, All Act Acc = 98.5% |
| 11 | 6minutewalk-matfiles | Joint angles (from joint angle files) | Single act classify | 3 | (2143, 60, 66) | Ind Act Acc = 99.97%, All Act Acc = 99.74% |
| 12 | 6minutewalk-matfiles | Extracted statistical values | Single act classify | 4 | (552, 10, 30) | Ind Act Acc = 85.48%, All Act Acc = 73.44% |
| 13 | NSAA | Position | D/HC classify | 1 | (9379, 60, 69) | Test Accuracy = 95.04% |
| 14 | NSAA | Position | Overall NSAA regress | 1 | (8034, 60, 69) | MSE = 13.15, MAE = 2.22 |
| 15 | NSAA | Position | Single act classify | 1 | (8034, 60, 69) | Ind Act Acc = 94.4%, All Act Acc = 82.94% |
| 16 | NSAA | Velocity | D/HC classify | 1 | (9379, 60, 69) | Test Accuracy = 69.94% |
| 17 | NSAA | Velocity | Overall NSAA regress | 1 | (8034, 60, 69) | MSE = 38.11, MAE = 4.28, RMSE = 6.17, R^2 = -0.0004 |
| 18 | NSAA | Velocity | Single act classify | 1 | (8034, 60, 69) | Ind Act Acc = 71.51%, All Act Acc = 4.75% |
| 19 | NSAA | Acceleration | D/HC classify | 1 | (9379, 60, 69) | Test Accuracy = 77.48% |
| 20 | NSAA | Acceleration | Overall NSAA regress | 1 | (8034, 60, 69) | MSE = 48.36, MAE = 4.62, RMSE = 6.95, R^2 = -0.4 |
| 21 | NSAA | Acceleration | Single act classify | 1 | (8034, 60, 69) | Ind Act Acc = 74.98%, All Act Acc = 17.25% |
| 22 | NSAA | Angular Velocity | D/HC classify | 1 | (9379, 60, 69) | Test Accuracy = 71.01% |
| 23 | NSAA | Angular Velocity | Overall NSAA regress | 1 | (8034, 60, 69) | MSE = 37.64, MAE = 4.27, RMSE = 6.13, R^2 = -0.002 |
| 24 | NSAA | Angular Velocity | Single act classify | 1 | (8034, 60, 69) | Ind Act Acc = 72.82%, All Act Acc = 19.12% |
| 25 | NSAA | Angular Acceleration | D/HC classify | 1 | X shape = (9379, 60, 69) | Test Accuracy = 71.17% |
| 26 | NSAA | Angular Acceleration | Overall NSAA regress | 1 | (8034, 60, 69) | MSE = 37.65, MAE= 4.42, RMSE = 6.14, R^2 = 0.03 |
| 27 | NSAA | Angular Acceleration | Single act classify | 1 | (8034, 60, 69) | Ind Act Acc = 73.07%, All Act Acc = 21.81% |
| 28 | NSAA | Sensor Free Acceleration | D/HC classify | 1 | X shape = (9379, 60, 51) | Test Accuracy = 75.59% |
| 29 | NSAA | Sensor Free Acceleration | Overall NSAA regress | 1 | (8034, 60, 51) | MSE = 40.10, MAE = 4.38, RMSE = 6.33, R^2 = 0.002 |
| 30 | NSAA | Sensor Free Acceleration | Single act classify | 1 | (8034, 60, 51) | Ind Act Acc = 74.87%, All Act Acc = 14.94% |
| 31 | NSAA | Sensor Magnetic Field | D/HC classify | 1 | (9379, 60, 51) | Test Accuracy = 99.57% |
| 32 | NSAA | Sensor Magnetic Field | Overall NSAA regress | 1 | (8034, 60, 51) | MSE = 4.49, MAE = 1.35, RMSE = 2.12, R^2 = 0.87 |
| 33 | NSAA | Sensor Magnetic Field | Single act classify | 1 | (8034, 60, 51) | Ind Act Acc = 99.32%, All Act Acc = 96.5% |
| 34 | NSAA | Joint Angle | D/HC classify | 1 | (9379, 60, 66) | Test Accuracy = 98.11% |
| 35 | NSAA | Joint Angle | Overall NSAA regress | 1 | (8034, 60, 66) | MSE = 3.95, MAE= 1.23, RMSE = 1.99, R^2 = 0.9 |
| 36 | NSAA | Joint Angle | Single act classify | 1 | (8034, 60, 66) | Ind Act Acc = 98.57%, All Act Acc = 93.38% |
| 37 | NSAA | Joint Angle XZY | D/HC classify | 1 | (9379, 60, 66) | Test Accuracy = 96.93% |
| 38 | NSAA | Joint Angle XZY | Overall NSAA regress | 1 | (8034, 60, 66) | MSE = 4.98, MAE = 1.36, RMSE = 2.23, R^2 = 0.88 |
| 39 | NSAA | Joint Angle XZY | Single act classify | 1 | (8034, 60, 66) | Ind Act Acc = 96.98%, All Act Acc = 89% |
| 40 | NSAA | Extracted statistical values | Overall NSAA regress | 1 | (742, 10, 30) | MSE = 28.71, MAE = 2.9 |
| 41 | NSAA | Extracted statistical values | Overall NSAA regress | 1 | (920, 10, 30) | MSE = 28.34, MAE = 2.26, RMSE = 5.32, R^2 = 0.39 |
| 42 | NSAA | Extracted statistical values | Overall NSAA regress | 1 | (1227, 10, 30) | MSE = 13.22, MAE = 1.56, RMSE = 3.64, R^2 = 0.69 |
| 43 | NSAA | Extracted statistical values | Overall NSAA regress | 1 | (1840, 10, 30) | MSE = 9.06, MAE = 1.13, RMSE = 3.01, R^2 = 0.79 |
| 44 | NSAA | Extracted statistical values | Overall NSAA regress | 1 | (3710, 10, 30) | MSE = 1.19, MAE = 0.52, RMSE = 1.09, R^2 = 0.97 |
| 45 | NSAA | Extracted statistical values | Overall NSAA regress | 1 | (7420, 10, 30) | MSE = 0.03, MAE = 0.16, RMSE = 0.18, R^2 = 0.99 |
| 46 | NSAA | Position | Overall NSAA regress | 1 | (1013, 60, 69) | MSE = 0.13, MAE = 0.18, RMSE = 0.37, R^2 = 0.52 |
| 47 | NSAA | Velocity | Overall NSAA regress | 1 | (1013, 60, 69) | MSE = 0.45, MAE = 0.44, RMSE = 0.67, R^2 = -0.38 |
| 48 | NSAA | Acceleration | Overall NSAA regress | 1 | (1013, 60, 69) | MSE = 0.38, MAE = 0.42, RMSE = 0.62, R^2 = -0.43 |
| 49 | NSAA | Angular Velocity | Overall NSAA regress | 1 | (1013, 60, 69) | MSE = 0.38, MAE = 0.38, RMSE = 0.61, R^2 = -0.24 |
| 50 | NSAA | Angular Acceleration | Overall NSAA regress | 1 | (1013, 60, 69) | MSE = 0.27, MAE = 0.34, RMSE = 0.52, R^2 = 0.03 |
| 51 | NSAA | Sensor Free Acceleration | Overall NSAA regress | 1 | (1013, 60, 51) | MSE = 0.33, MAE = 0.4, RMSE = 0.57, R^2 = 0.04 |
| 52 | NSAA | Sensor Magnetic Field | Overall NSAA regress | 1 | (1013, 60, 51) | MSE = 0.13, MAE = 0.17, RMSE = 0.37, R^2 = 0.54 |
| 53 | NSAA | Joint Angle | Overall NSAA regress | 1 | (1013, 60, 66) | MSE = 0.09, MAE = 0.12, RMSE = 0.3, R^2 = 0.65 |
| 54 | NSAA | Joint Angle XZY | Overall NSAA regress | 1 | (1013, 60, 66) | MSE = 0.14, MAE = 0.18, RMSE = 0.37, R^2 = 0.55 |
| 55 | NSAA | Position | Overall NSAA regress | 1 | (13365, 60, 69) | MSE = 16.27, MAE = 2.76, RMSE = 4.03, R^2 = 0.64 |
| 56 | NSAA | Position | Overall NSAA regress | 1 | (13314, 90, 69) | MSE = 17.51, MAE = 2.6, RMSE = 4.18, R^2 = 0.6 |
| 57 | NSAA | Position | Overall NSAA regress | 1 | (13315, 120, 69) | MSE = 17.89, MAE = 2.81, RMSE = 4.23, R^2 = 0.59 |
| 58 | NSAA | Position | Overall NSAA regress | 1 | (13313, 180, 69) | MSE = 15.12, MAE = 2.44, RMSE = 3.89, R^2 = 0.67 |
| 59 | NSAA | Joint Angle | Overall NSAA regress | 1 | (13365, 60, 66) | MSE = 4.46, MAE = 1.24, RMSE = 2.11, R^2 = 0.9 |
| 60 | NSAA | Joint Angle | Overall NSAA regress | 1 | (13314, 90, 66) | MSE = 4.99, MAE = 1.27, RMSE = 2.23, R^2 = 0.89 |
| 61 | NSAA | Joint Angle | Overall NSAA regress | 1 | (13315, 120, 66) | MSE = 4.21, MAE = 1.16, RMSE = 2.05, R^2 = 0.91 |
| 62 | NSAA | Joint Angle | Overall NSAA regress | 1 | (13313, 180, 66) | MSE = 2.65, MAE = 0.99, RMSE = 1.63, R^2 = 0.94 |
| 63 | NSAA | Sensor Magnetic Field | Overall NSAA regress | 1 | (13365, 60, 51) | MSE = 3.05, MAE = 1.13, RMSE = 1.75, R^2 = 0.93 |
| 64 | NSAA | Sensor Magnetic Field | Overall NSAA regress | 1 | (13314, 90, 51) | MSE = 2.47, MAE = 0.89, RMSE = 1.57, R^2 = 0.94 |
| 65 | NSAA | Sensor Magnetic Field | Overall NSAA regress | 1 | (13315, 120, 51) | MSE = 2.31, MAE = 0.84, RMSE = 1.52, R^2 = 0.95 |
| 66 | NSAA | Sensor Magnetic Field | Overall NSAA regress | 1 | (13313, 180, 51) | MSE = 7.07, MAE = 1.76, RMSE = 2.66, R^2 = 0.84 |
| 67 | NSAA | Joint Angle XZY | Overall NSAA regress | 1 | (13365, 60, 66) | MSE = 5.99, MAE = 1.44, RMSE = 2.45, R^2 = 0.86 |
| 68 | NSAA | Joint Angle XZY | Overall NSAA regress | 1 | (13314, 90, 66) | MSE = 4.29, MAE = 1.28, RMSE = 2.07, R^2 = 0.9 |
| 69 | NSAA | Joint Angle XZY | Overall NSAA regress | 1 | (13315, 120, 66) | MSE = 4.72, MAE = 1.24, RMSE = 2.17, R^2 = 0.9 |
| 70 | NSAA | Joint Angle XZY | Overall NSAA regress | 1 | (13313, 180, 66) | MSE = 2.62, MAE = 1.01, RMSE = 1.62, R^2 = 0.94 |
| 71 | NSAA | Extracted statistical values | Overall NSAA regress | 1 | (13240, 10, 10) | MSE = 0.03, MAE = 0.12, RMSE = 0.17, R^2 = 0.99 |
| 72 | NSAA | Extracted statistical values | Overall NSAA regress | 1 | (13240, 10, 20) | MSE = 0.02, MAE = 0.09, RMSE = 0.14, R^2 = 0.99 |
| 73 | NSAA | Extracted statistical values | Overall NSAA regress | 1 | (13240, 10, 30) | MSE = 0.01, MAE = 0.07, RMSE = 0.1, R^2 = 0.99 |
| 74 | NSAA | Extracted statistical values | Overall NSAA regress | 1 | (13240, 10, 40) | MSE = 0.01, MAE = 0.09, RMSE= 0.12, R^2 = 0.99 |
| 75 | NSAA | Extracted statistical values | Overall NSAA regress | 1 | (13240, 10, 50) | MSE = 0.02, MAE = 0.09, RMSE = 0.13, R^2 = 0.99 |
| 76 | NSAA | Extracted statistical values | Overall NSAA regress | 1 | (13240, 10, 30) | MSE = 0.01, MAE = 0.07, RMSE = 0.1, R^2 = 0.99 |
| 77 | NSAA | Extracted statistical values | Overall NSAA regress | 1 | (6550, 20, 30) | MSE = 1.13, MAE = 0.5, RMSE = 1.06, R^2 = 0.98 |
| 78 | NSAA | Extracted statistical values | Overall NSAA regress | 1 | (3185, 40, 30) | MSE = 1.05, MAE = 0.67, RMSE = 1.02, R^2 = 0.98 |
| 79 | NSAA | Extracted statistical values | Overall NSAA regress | 1 | (13240, 10, 30) | MSE = 0.01, MAE = 0.07, RMSE = 0.1, R^2 Score = 0.99 |
| 80 | NSAA | Extracted statistical values | Overall NSAA regress | 1 | (12794, 20, 30) | MSE = 0.26, MAE = 0.26, RMSE = 0.51, R^2 = 0.99 |
| 81 | NSAA | Extracted statistical values | Overall NSAA regress | 1 | (12195, 40, 30) | MSE = 0.53, MAE = 0.42, RMSE = 0.73, R^2 = 0.99 |
| 82 | NSAA | Extracted statistical values | Overall NSAA regress | 1 | (13240, 10, 30) | MSE = 0.01, MAE = 0.07, RMSE = 0.1, R^2 = 0.99 |
| 83 | NSAA | Extracted statistical values | Overall NSAA regress | 1 | (13265, 7, 30) | MSE = 0.05, MAE = 0.15, RME = 0.21, R^2 = 0.99 |
| 84 | NSAA | Extracted statistical values | Overall NSAA regress | 1 | (13305, 5, 30) | MSE = 0.11, MAE = 0.19, RMSE = 0.33, R^2 = 0.99 |
| 85 | NSAA | Extracted statistical values | Overall NSAA regress | 1 | (13453, 3, 30) | MSE = 0.12, MAE = 0.19, RMSE = 0.35, R^2 = 0.99 |
| 86 | NSAA | Joint Angle | Overall NSAA regress | 1 | (13365, 60, 66) | MSE = 4.46, MAE = 1.24, RMSE = 2.11, R^2 = 0.9 |
| 87 | NSAA | Joint Angle | Overall NSAA regress | 1 | (13314, 90, 66) | MSE = 4.99, MAE = 1.27, RMSE = 2.23, R^2 = 0.89 |
| 88 | NSAA | Joint Angle | Overall NSAA regress | 1 | (13315, 120, 66) | MSE = 4.21, MAE = 1.16, RMSE = 2.05, R^2 = 0.91 |
| 89 | NSAA | Joint Angle | Overall NSAA regress | 1 | (13313, 180, 66) | MSE = 2.65, MAE = 0.99, RMSE = 1.63, R^2 = 0.94 |
| 90 | NSAA | Joint Angle | Overall NSAA regress | 1 | (13314, 60, 66) | MSE = 3.36, MAE = 1.21, RMSE = 1.83, R^2 = 0.93 |
| 91 | NSAA | Joint Angle | Overall NSAA regress | 1 | (13315, 60, 66) | MSE = 3.43, MAE = 1.13, RMSE = 1.85, R^2 = 0.93 |
| 92 | NSAA | Joint Angle | Overall NSAA regress | 1 | (13313, 60, 66) | MSE = 3.13, MAE = 1.07, RMSE = 1.77, R^2 = 0.93 |
| 93 | NSAA | Joint Angle | Overall NSAA regress | 1 | (13240, 60, 66) | MSE = 2.4, MAE = 0.88, RMSE = 1.55, R^2 = 0.95 |
| 94 | NSAA | Joint Angle | Overall NSAA regress | 1 | (13051, 60, 66) | MSE = 1.06, MAE = 0.6, RMSE = 1.03, R^2 = 0.98 |
| 95 | NSAA | Joint Angle | Overall NSAA regress | 1 | (12436, 60, 66) | MSE = 1.48, MAE = 0.66, RMSE = 1.22, R^2 = 0.97 |
| 96 | NSAA | Joint Angle | Overall NSAA regress | 1 | (11530, 60, 66) | MSE = 0.35, MAE = 0.41, RMSE = 0.59, R^2 = 0.99 |
| 97 | NSAA | Joint Angle | Overall NSAA regress | 1 | (10130, 60, 66) | MSE = 0.35, MAE = 0.46, RMSE = 0.59, R^2 = 0.99 |
| 98 | NSAA | Joint Angle | Overall NSAA regress | 1 | (7481, 60, 66) | MSE = 0.73, MAE = 0.7, RMSE = 0.86, R^2 = 0.98 |

= Experiment Set 1 = Experiment Set 6

= Experiment Set 2 = Experiment Set 7

= Experiment Set 3 = Experiment Set 8

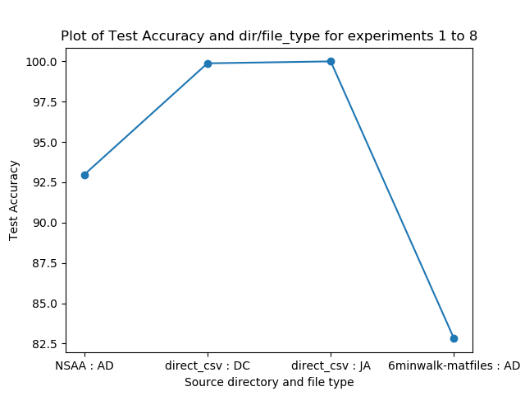
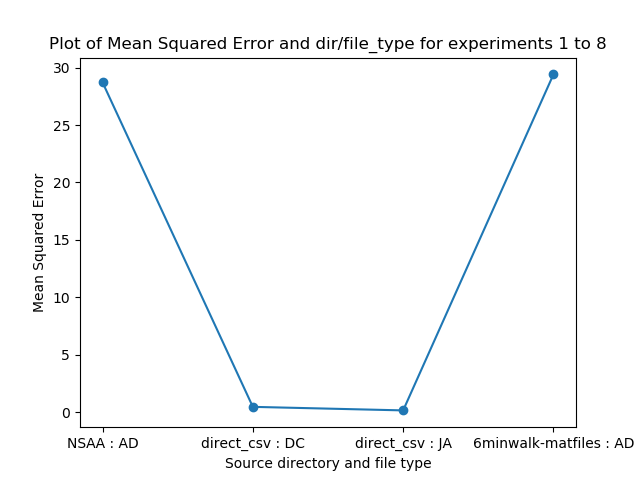
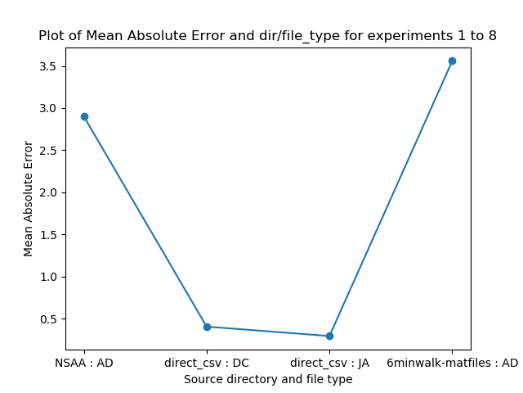
= Experiment Set 4 = Experiment Set 9

= Experiment Set 5 = Experiment Set 10

**Subjects Lists**

|  |  |
| --- | --- |
| **Subject List 1** | D2, D3, D4, D4V2, D5, D6, D7, D9, D10, D11, D12, D14, D15, D16, D17, D18, D19, HC3, HC4, HC5, HC6, HC7, HC8, HC9, HC10 |
| **Subject List 2** | D2, D3, D4, D5, D6, D7, D9, D10, D11, D12, D14, D15, D17, D18, D19, HC1, HC2, HC3, HC4, HC5, HC6, HC7, HC8, HC9, HC10 |
| **Subject List 3** | D2, D3, D4, HC2, HC5, HC6 |
| **Subject List 4** | D2, D3, D4, D5, D6, D7, HC1, HC2, HC3, HC4, HC5, HC7, HC8, HC9, HC10 |

**Experiment Set 1: Performance of RNNs on Different Source Data**

The purpose of this experiment set is to determine whether and how well RNN models regresses on different types of source data. By this, we mean data that is in the same format (i.e. ‘.mat’ files in the same organizational structure) but representing different measurements from different source directories. These are thus:

* **NSAA : AD** – statistical values extracted from the ‘all-data’ (‘AD’) files by the ‘comp\_stat\_vals.py’ script, whose files are sourced from the ‘NSAA’ directory; hence these are stat values of the subjects performing the NSAA activities, which are then used to train model(s).
* **direct\_csv : DC** – these are the joint angle values (i.e. raw measurements, not computed stat values) that are sourced from the data cube; this data cube contains the 6-minute walk data from various subjects, several of which aren’t included in the standard joint angle files of subjects doing the 6-minute walk.
* **direct\_csv : JA** – similar to ‘direct\_csv : DC’ as described above, using the same source directory type and raw measurement, but containing joint angle files that aren’t necessarily included within the data cube.
* **6minwalk-matfiles : AD** – again, uses the files corresponding to subjects’ 6-minute walk assessments, with the difference this time being that we aren’t using raw joint angle files either in the data cube format or as ‘loose’ files, but rather computing statistical values via ‘comp\_stat\_vals.py’ script; in this sense, it’s the same as ‘NSAA : AD’ models but using different assessment data (6-minute walk rather than NSAA).

For the second and third diagrams above, with regards to the output type, all these file types and directory sources outlined above are used to train models trained to regress on the overall NSAA score for a given sequence from a file. This value is able to range between 0 and 34 (though based on how the assessment is done, it generally ranges between 15 and 34). Hence, for a given type of source data, if it has a MAE = 0.5, that means that, on test data of sequences from files of the given type of source data, the model predicts for each of them a score of between 0 and 34 on average ‘0.5’ away from the true value for that sequence (the true value for a sequence being the overall NSAA score of the file that the sequence comes from). However, for the first diagram above, it tests the different files types on its ability to classify whether a sequence that come from a file are from a ‘D’ or ‘HC’ subject.

**Results Discussion and Conclusions**

In using just the raw joint angle values from ‘DC’ or ‘JA’ files, we achieve an approximate **99%** accuracy (i.e. for each sequence of 60 rows of 66 joint angle values, the model can predict with 99% accuracy whether it came from a ‘D’ file or an ‘HC’ file); however, looking at more measurements (e.g. position, accelerometer values, etc.), performing manual feature extraction via computing of the statistical values and then reducing the dimensionality, and then training the model provides a much worse classification accuracy of **82.81%** for data that comes from the same assessment (6-minute walk). This can also be seen when the same data sources are then used to train the RNN to perform regression for the overall NSAA score: the raw joint angle data gives a much smaller **MAE = 0.4037** (meaning that it predicts a score of between 0 and 34 which on average is 0.4037 away from the true value in either direction), compared with a much worse **MAE = 3.56** from 6-minute walk ‘AD’ statistical value files. A further observation can be made about the experiments concerning the raw joint angle files in that they were performing much better than we were expecting them to be: by simply considering only the joint angle measurement of a subjects suit data, given 1 second’s worth of an input sequence to the RNN, it can correctly classify whether the frame comes from a healthy control subject or one with DMD to a very high accuracy of 99% and predict the overall NSAA score to within 0.4037 of the true value of between 0 and 34. This is extraordinarily high, much better than the ability of medical professionals and, most notably, this is only the first iteration of the experiments with raw measurement files.

This large difference between raw measurements and computed statistical features may seem counterintuitive at first glance, as the former is just using data direct from the provided ‘.mat’ files, while with the latter we actually process it further to ideally extract more important features from the data. One possible reason for raw joint angle models performing so well could be that there aren’t that many subjects with these files that are provided to us in comparison with ‘AD’ files, so not as diverse a training set is used; this means that there would be a narrower range of overall NSAA scores to regress towards, making it easier for the model to approximate the true score with a smaller margin of error. Another possible reason could be that the features that the RNN models extracts from its inputted data are more useful for it to train on and approximate an overall score: a prominent benefit of using neural networks is that they are traditionally noted to perform better with raw data rather than manually extracted features. Furthermore, when it trains on the stat value data from ‘comp\_stat\_values.py’ for the ‘AD’ file types, it still computes its own features within the RNN, and so this ‘features from features’ behaviour might have proved to be problematic for the network.

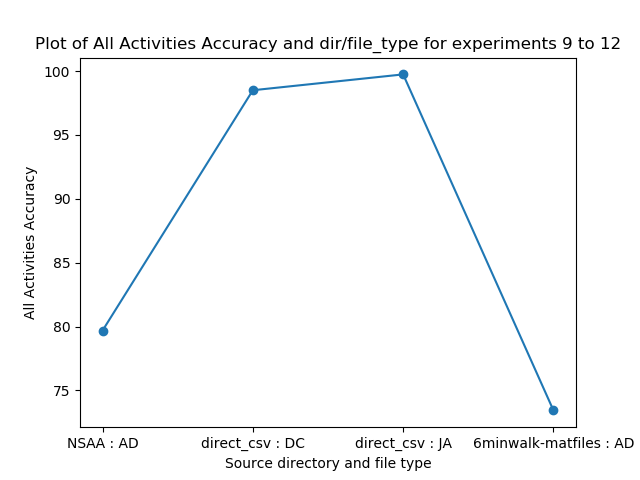
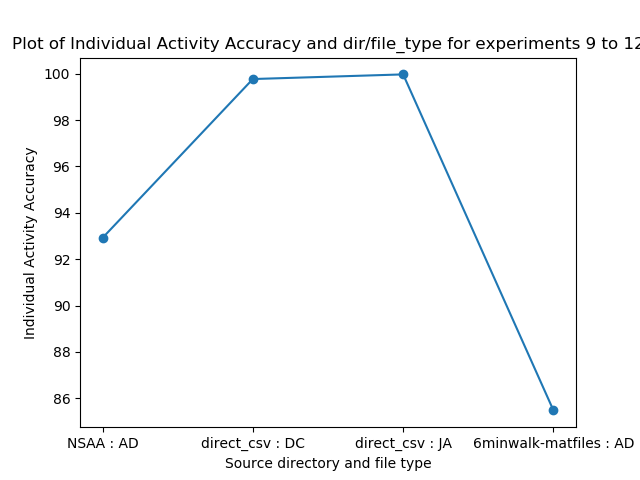
It should also be noted that training the RNN with raw joint angle data requires far fewer training epochs (~20) than for statistical values (~100). This will primarily be due to the larger amount of raw joint angle data that is fed through the network; in the case of training on all files within the data cube, the ‘x’ input shape is (8470, 60, 66) while for the corresponding AD files we only have (552, 10, 30) samples due to how computing statistical values dramatically reduces the raw amount of available data. This decrease in amount of available data also might help to account for the reduction in accuracy when using data sourced from AD files.

Overall, given our limited data and without any way to currently increase it, we come to several conclusions:

1. **RNN models can successfully regress and classify on data 6-minute walk and NSAA assessment data**
2. **Not only this, but raw joint angles perform phenomenally well in both classification and regression**
3. **The raw data measurements might be better put through RNNs than extracted statistical features**
4. **This is probably due to extracted statistical features being a lot smaller of a data set, RNNs better utilizing raw measurements over pre-computed features, and there being a small range of target values with files sourced as ‘JA’ or ‘DC’**

We shall shortly be examining ways to improve the performance of models trained on computed statistical values by increasing sequence overlap, modifying sequence length, and so on.

**Experiment Set 2: Performance of RNNs for Single Activity Classification**



This experiment utilizes the same source data types as experiment set 1. Hence, we won’t discuss what each of these data types represent in the x-axes of the graphs above, as these are exactly the same files as used previously. The difference with this experiment set, however, is that it is looking at a different output type: while experiment set 1 looked at performance for classification of D/HC labels and overall NSAA scores for given test sequences, this experiment set looks at predicting multiple classification values for a single sequence. Here, the models are trained to predict a sequence of 17 values of numbers of either 0, 1, or 2. These represent the single activity scores for a single sequence that correspond to the single activity scores of the file that the sequence is sourced from. It’s worth noting that, as overall NSAA score is the sum of these, a sequence will have an overall NSAA score that is equal to the sum of the same sequence’s 17 single act scores. Furthermore, when predicting single act scores, the RNN architecture will be different: for D/HC classification or overall NSAA score regression, there will be only 1 output neuron (though predicting different ranges of values for the two tasks), while for single acts there are 17 output neurons, 1 for each single act it is predicting. The overall aim of this experiment is thus to see if a model is able to predict, given a sequence of values from a file (that corresponds to a subject, e.g. ‘D4’), what that file’s single-act scores are. Note that the two metrics that we are using (that are computed by each RNN model over all its testing data) is individual activity accuracy and all activities accuracy: the former is the percentage of activities in the testing data set that were correctly predicted to be a 0, 1, or 2, while the latter is the percentage of whole activity sets (i.e. single RNN output of 17 values for a single test sequence) that were correctly predicted.

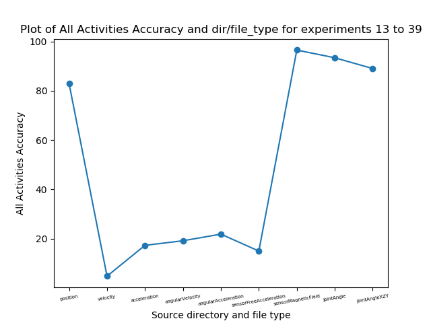
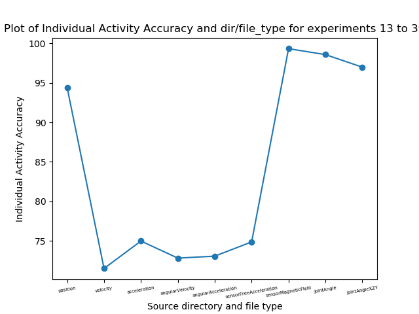
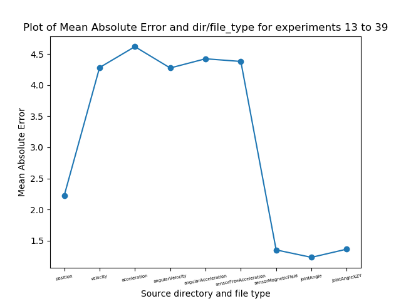
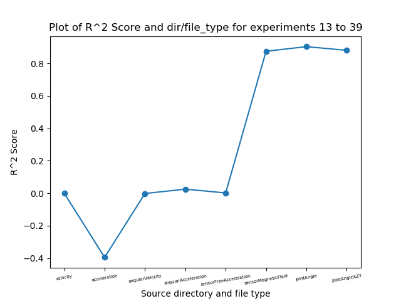
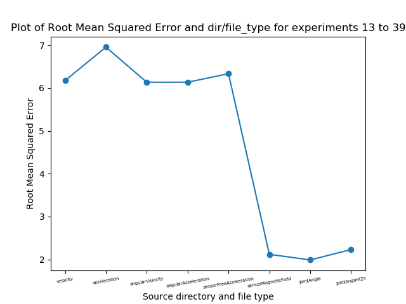
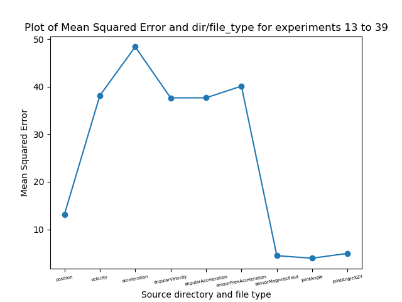
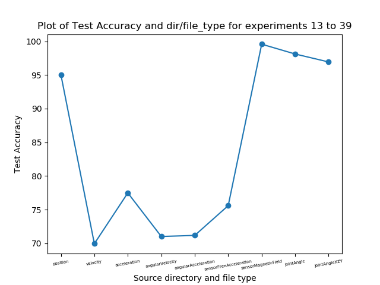
**Results Discussion and Conclusions**

In line with the results from experiment set 1, we can see here that raw joint angle data is much more useful in predicting the single act scores for test data of sequences than stat values from ‘AD’ files. This is most likely down to the same reasons as for the other output types; namely, comparatively small about of ‘AD’ data, neural networks better utilizing raw measurements, etc. Given that it is consistent with the previous experiment set results in determining the comparative performance for different source data types, the results of the experiment set is therefore more about whether it is possible for these data types to train a RNN to predict sequences of values that correspond to activities that the sequence being tested might not be from (i.e. the sequence will be from part of a file that, at most, corresponds to one activity the subject is performing, and therefore can only try to assess what its likely other single activity scores are). The result is that it performs well, especially with joint angle data, predicting approximately **99%** of the individual activity scores and **98%** of the all activity sequence scores.

From this, we can draw several conclusions:

1. **The different types of source data can be used to train a model to accurately predict single act values**
2. **Like the previous two output types, raw joint angle measurements do this better than ‘AD’ stat values**

**Experiment Set 3: Raw Measurements for All Output Types**



With this experiment set, we now turn our attention to just looking exclusively at raw measurements. This is primarily due to the much better performance of models trained on raw measurements compared with ones trained on stat values from ‘AD’ files, as seen in experiment sets 1 and 2. The question we thus ask is: can we show similarly high performance when we train models on types of raw measurements from the suit data other than just joint angles? To this end, we look at 9 raw measurements in total that are contained within the ‘AD’ files and that are recorded by the suit during use: position, velocity, acceleration, angularAcceleration, angularVelocity, sensorFreeAcceleration, sensorMagneticField, jointAngle, and jointAngleXZY. Unlike the joint angle measurements, we don’t have the other raw measurements in separate, unique files. Therefore, we make extensive use of the ‘ext\_raw\_measures.py’ script in order to extract, for every file and for every raw measurement, the measurement data and store them in separate files as ‘.csv’ files. From here, these are then able to train and test an RNN model in the same way.

For this experiment set, we are exclusively concerning ourselves now with NSAA assessment files rather than 6-minute walk files. This is more a choice based on the intended direction of the project to be more concerned with assessing and making predictions concerning the NSAA assessments and how that is connected to natural movement data (more on this later) than the 6-minute walk data. Thus, each of the entries along the x-axis of the graphs represents a single model trained on that particular raw measurement data for every NSAA file we have available. Currently, we are just looking at how the models performs on testing data from subjects it has already seen before, rather than complete subjects being left out of the training set, though this shall be explored later on with ‘left out’ subjects in trained models. Furthermore, though each of the models are trained with the same number of files and each with the same sequence length (due to there at every time step the suit records all raw measurements), the feature size will vary; in other words, if the training shape to the RNN models are of shape (x, y, z), ‘x’ and ‘y’ will always be the same between raw measurements but ‘z’ will vary: for raw measurements based on segment measurements of the suit ‘z’ will be 69, while for angle-based measurements it will be 66 and for sensor readings it will be 51.

**Results Discussion and Conclusions**

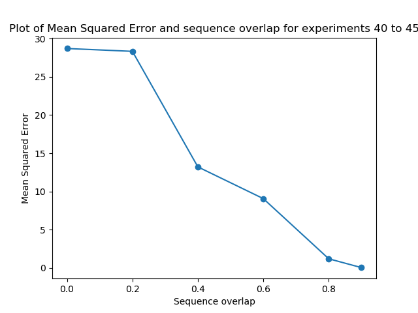
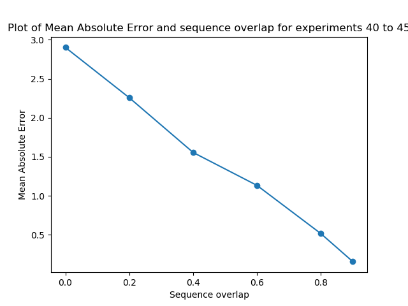
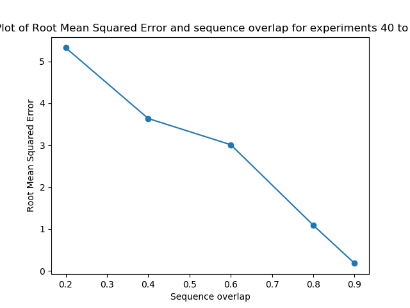
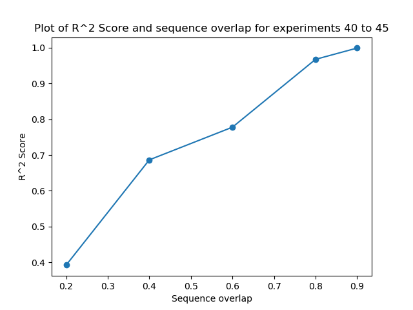
A consistent finding in all of the above graphs is that there are three raw measurements that are far and away better than the others: jointAngle, jointAngleXZY, and sensorMagneticField. These measurements can be seen to perform better for all three output types we are training towards: D/HC classification (graph 1), overall NSAA score (graphs 2 – 5), and individual activities classification(graphs 6 – 7). Both jointAngle and jointAngleXZY raw measurements were to be expected to perform this well, as this is consistent with results obtained in experiment sets 1 and 2 (note that this isn’t using exactly the same files, as experiment sets 1 and 2 use joint angle exclusive files that were pre-extracted before this project’s inception, while the models trained on ‘jointAngle’ here were instead measurements extracted from the ‘AD’ files and represent somewhat different subjects than the ‘JA’ or ‘DC’ files).

Position is another measurement that performs particularly well compared with the other 5 measurements, though for some metrics (e.g. MAE for overall NSAA score) it performs noticeably worse than jointAngle, jointAngleXZY, and sensorMagneticField. We can observe that these 4 ‘useful’ measurements predict **~97%** of test sequences accurately for D/HC classification, compared with only **~73%** for the other 5. Furthermore, when we look at models trained for overall NSAA score regression, we note an average MAE of **~1.5** for the useful 4 measurements, while we see an average MAE of **~4.3** for the other 5; and for single-activity classification for individual activity accuracy, we see **~97%** for the useful 4 and **~74%** for the other 5. With respect to the performance of the useful 4 measurements, while we expect to see performance for jointAngle and jointAngleXZY, the usefulness of sensorMagneticField and position is slightly more surprising. For position, we can speculate this to be as a result of subjects with more severe Duchenne being more inclined to have their limbs and torso in positions which are very indicative of their condition, compared with the healthy-control patients. The sensorMagneticField measurement, however, is slightly more mystifying and does not have an obvious explanation at this point. What’s more, our initial speculation was that the velocity and acceleration measurements would be fairly useful to distinguish between D and HC subjects and, moreover, help predict their overall NSAA scores, as we believed that, as these are key measurements of a subject’s potential for movement, they would be useful for training an RNN model. We found these, however, to perform comparatively badly on test sequences, which leads us to believe that either velocities and acceleration of movement between subjects don’t vary that much or that they aren’t as indicative of movement ability as we thought.

Therefore, the conclusions we draw are as follows:

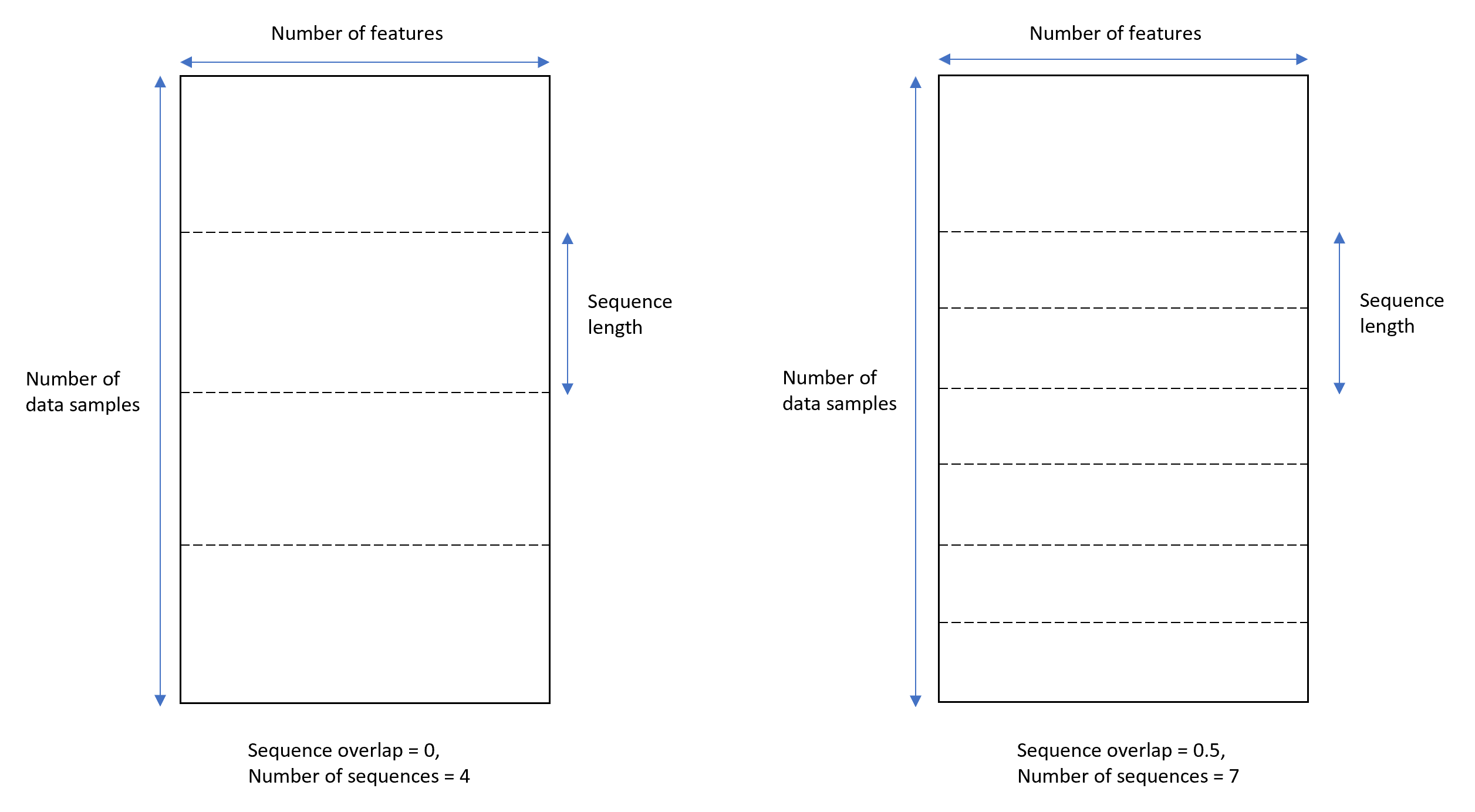
1. **jointAngle, jointAngleXZY, position, and sensorMagneticField are useful raw measurements for building RNN models on, while velocity, acceleration, angularVelocity, angularAcceleration, and sensorFreeeAcceleration are not**
2. **sensorMagneticField is surprisingly well performing (more or less just as good as the joint angle measurements), while position is still strong but slightly worse, and velocity and acceleration aren’t as useful as initially thought**

**Experiment Set 4: Sequence Overlap for Stat Values from AD Files**

One of the problems with stat values extracted from ‘AD’ files via the ‘comp\_stat\_vals.py’ script was that it dramatically reduced the amount of available data we have for training: when we are computing the stat values over intervals of 1 second (the standard measurement used for this project thus far), we are essentially doing calculations over 60 rows of raw measurement data (due to the sampling rate of the suit being 60Hz and we chose 1 second’s worth of data) and producing as an output 1 row of data. While this theoretically contains much of the useful information of the 60 rows simply condensed into 1 row, it doesn’t change the fact that we have now reduced the amount of raw data that we feed into the RNN 60-fold. This is most likely a large factor in the comparatively-weak results of stat values from ‘AD’ files seen in experiment sets 1 and 2. However, a way we chosen to get around this is by using a sequence overlap.

Sequence overlap is essentially used here as follows: consider a 2D block of data that we have available that is produced by the ‘comp\_stat\_vals.py’ script and ‘ft\_sel\_red.py’ to reduce to dimensionality of the data. This block has a number of rows equal to the number of produced rows of stat value data over all files that we are using (e.g. all NSAA files available in the directory) and a number of columns equal to the reduced dimensionality produced by ‘ft\_sel\_red.py’ (e.g. from ~4000 from ‘comp\_stat\_vals.py’ to ~30 to be read in by ‘rnn.py’). Normally, we will take ‘slices’ of this block to produce smaller blocks with a number of rows now equal to ‘sequence length’ before moving on to the next block below it until all data is consumed (ignoring leftover data at the end of the block that won’t fit into a smaller block). This is the case with the diagram below on the left.



However, if we consider a sequence overlap of 50% (i.e. overlap proportion of 0.5), we are instead able to capture 7 blocks of data rather than 4. Scaled up to smaller sequence lengths relative to the number of data samples, this is approximately 50% more data; if we scale up to an overlap of 0.9, we have 900% more data. Though we end up with a fair bit of redundancy with this approach (as the same vector of a data sample is used in numerous sequences), there are two primary benefits of doing this:

1. Much more available data, which is useful to train the models to become much more accurate.
2. With sequences that have no overlap, there is a high chance that one or more activities that occur in the NSAA assessment might be ‘cut’ along these lines (see the dashed lines in the above left image); this would mean the activity isn’t included in an entire sequence, while with a sufficient sequence length and sequence overlap, it’s more likely to be captured in its complete form in at least one of the sequences.

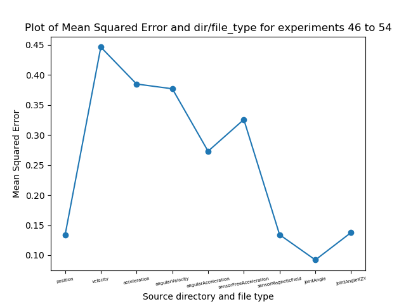
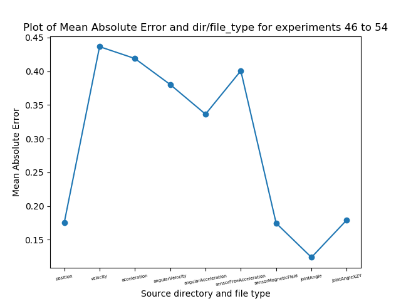
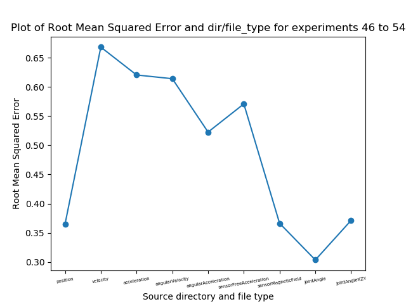
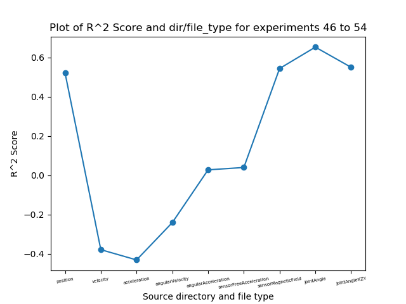
**Results Discussion and Conclusions**

Note that for this experiment set, we are only concerned with the overall NSAA score regression output type, as previous results show the performance of models on this output type are generally very consistent with results on other output types, hence there isn’t expected to be any need to repeat these experiments with the other output types here. Unsurprisingly, we see a massive increase in the performance with a larger sequence overlap. This will be in part due to the massive increase in available data: while the total data for NSAA files from stat values produced by ‘comp\_stat\_vals.py’ for a sequence over of 0 is (742, 10, 30) and shows a **MAE = ~2.9**, when the sequence overlap increases to 0.9, we have total data of shape (7420, 10, 30), ten times as many available sequences, which results in **MAE = ~0.1**. Note that these results are still just on test data, so it shows that the models can generalize much better to new, unseen sequences the overlap is much higher.

We can therefore conclude two things:

1. **A large sequence overlap leads to better performance for stat values from ‘AD’ files for NSAA files.**
2. **This is most likely due to the larger amount of available data and the increased window to capture complete activities.**

**Experiment Set 5: Sequence Overlap for Stat Values from AD Files**

This is the first experiment set that makes use of ‘mat\_act\_div.py’ to use the source ‘.mat’ files and the annotated Google sheet to experiment with what we call ‘single-act files’. These are files that contain only a single NSAA assessment within it and no other activities. For example, if we take the ‘D4’ source ‘.mat’ file, we then create 17 new files containing the 17 NSAA assessments, with each new file being a name like ‘D4\_act1.mat’, ‘D4\_act2.mat’, etc. This process obviously contains files that are a lot smaller than the source file, as well as cutting out a lot of the bits in between the assessments in the original file. For example, the original file will contain a complete recording of the suit data for a subject; this includes the data where a subject isn’t doing anything particularly relevant (e.g. standing around before beginning the next activity of the assessment or trying but failing to do one of the assessments). For more details on how this is done, see the section within the script ecosystem overview on ‘mat\_act\_div.py’.

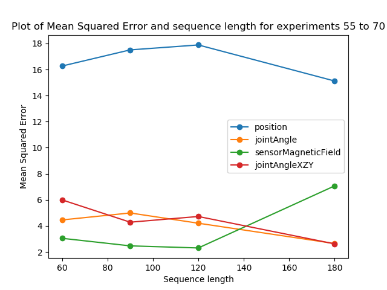
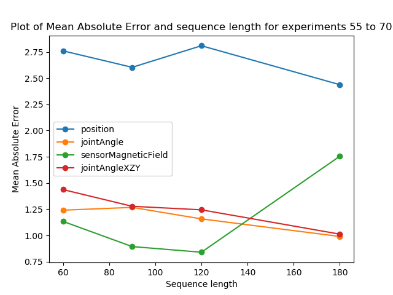
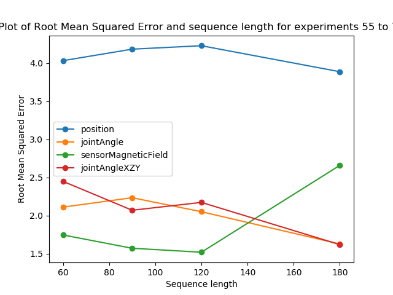
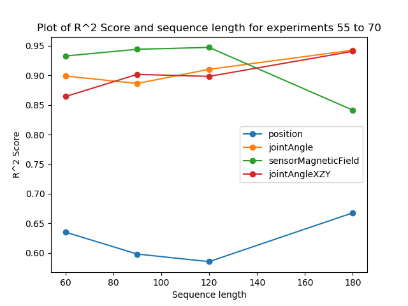
The aim of this experiment was to build models that are trained on one type of raw measurement; this is similar to experiment set 3 but with two differences: we’re now assessing on the single activity score for the file in question (i.e. a value of 0, 1, or 2), and we’re now using files that contain a single activity from a subject rather than all of the assessments (in most cases, though some smaller source files exist that contain only a handful of activities). We wanted to see whether or not the same raw measurements were as comparatively useful for single act source files as they were for the original files that contained all of the data; in a sense, we wanted to see if the measurements were as consistent when we removed a lot of the ‘non-assessment’ data from the source files (that is, data within the source files that don’t have an activity being undertaken). If this was the case, then we can conclude fairly strongly that these raw measurements are informative for NSAA assessments. Also, it’s worth noting that the ‘y labels’ assigned to each sequence coming from a single act file is now based on not only the file name but also what activity it represents; these two features are used to lookup the relevant value from the annotated Google sheet to get the assessed value of the activity.

**Results Discussion and Conclusions**

As expected, the raw measurements that proved to be the ‘best’ in experiment set 3 (jointAngle, jointAngleXZY, sensorMagneticField, and position) proved to be the best here as well, with the other 5 raw measurements heavily underperformed in comparison. This can be seen by the ‘useful’ 4 raw measurements obtaining a **MAE = ~0.15** while the others obtaining a **MAE = ~0.37**. Hence, this performs as we would expect, and the removal of a lot of the data between assessments from the source files does not affect the relative importance of some of the measurements. We can therefore conclude the following:

1. **The relative importance of jointAngle, jointAngleXZY, sensorMagneticField, and position as raw measurements compared to the other 5 is consistent with using just single act files rather than the complete source files.**

**Experiment Set 6: Different sequence lengths w/ overlaps for raw measurements**

For raw measurements, we have been (up until now) using a sequence length of ‘60’ as default for all of the experiments sets thus far. This represents a time window for the sequence of 1 second, as the suit is sampling at 60Hz, which therefore means that there are 60 rows of data stored in the source ‘.mat’ files every 1 second. Therefore, we wish to know whether or not this is a good time window to create each sequence with; however, with a given amount of source data, if we increase the sequence length, then we decrease the number of overall samples we can taken from this finite block. Therefore, it was decided to use a scaling sequence overlap; that is, we increase the sequence overlap in proportion to the sequence length to keep the number of samples more-or-less constant (accounting for differing numbers of left-over data at the ends of every file that can’t be made into a sequence). This is done by the following formula:

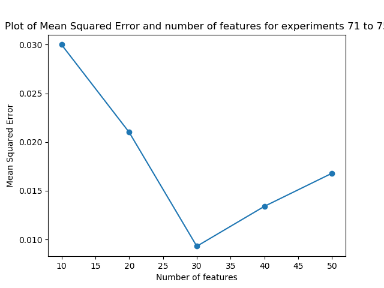
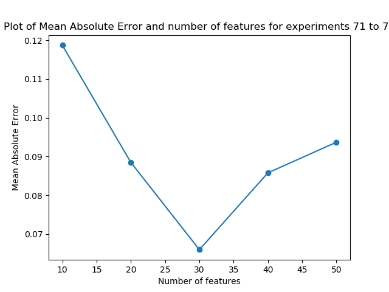
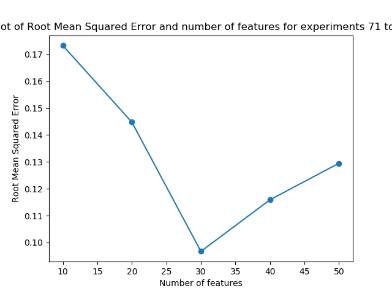
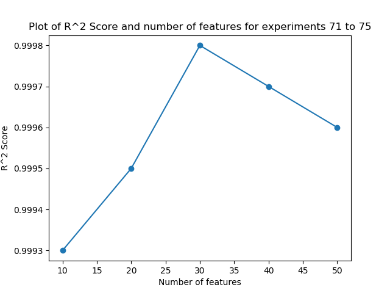
…where ‘’ and ’. For example, given our original sequence length of ‘60’, if we wish to experiment with ‘’, then (as ) and therefore (that is to say, we must set the sequence overlap proportion to 0.6667 to maintain a constant amount of samples. The reason we do this is to hold the dataset set size as a constant, and therefore any changes in performance of differing sequence lengths would be due to sequence length alone. Also note that we again only assess the differing sequence length on the overall NSAA score output type (for reasons previously mentioned in the experiment set 5 discussion. Finally, we wish to explore differing sequence lengths on multiple different types of raw measurements and compared them side by side (hence why we plot them all together on the same graph); however, we exclude 5 of the less ‘useful’ measurements, as they’ve been determined in experiment sets 3 and 5 to be not as useful as the other 4; hence to same time and graph readability, we don’t consider them here.

**Results Discussion and Conclusions**

What we were looking for in the above graphs is a consistent pattern for a given graph among the four plotting lines of an increase or decrease in performance when we change the sequence length; in other words, whether performance increased or decrease in the same way for all raw measurements for a given sequence length. This would give us a strong indication of how sequence length alters the performance of a certain metric. However, not only can we not any inter-raw-measurements patterns of change but we can’t see any discernible intra-raw-measurements pattern (i.e. can’t see single lines changing particularly strongly with respect to the output metric). For example, looking at the graph for the MAE metric, none of the lines for any of the raw measurements show any consistent improvement or worsening with increasing sequence length; each of them improves at some point on increasing the sequence length and also worsens at some point on increasing the sequence length. Additionally, the lines aren’t consistent with each other: the improving or worsening performance with increasing sequence length is often exclusive to one measurement and doesn’t necessarily translate to the others. From this, we can conclude:

1. **Increasing sequence length for data from raw measurements beyond 60 doesn’t show a consistent increase or decrease of performance amongst different useful raw measurement types; this is most likely due to the capturing window of 1 second being long enough to correctly assess for a given output type**
2. **To save on computational cost of training and testing on unnecessarily-longer sequences, we decide to keep the sequence length of 60**

**Experiment Set 7: Number of features needed for stat value data**

We now turn our attention back to stat value data from ‘AD’ files (i.e. the output of ‘comp\_stat\_vals.py’ as opposed to raw measurements from files obtained by ‘ext\_raw\_measures.py’). As a requirement of the pipeline, the data produced by ‘comp\_stat\_vals.py’ needs to have its dimensions dramatically reduced while keeping as much of the inherent variation of the data set still present. In the context of the data block diagram as seen in the section for experiment set 4, this is the ‘width’ of the block. As sequences are extracted from the source data ‘block’ by moving downwards, the number of features we chose to set is independent of the sequence length, sequence overlap, and discard proportion (more on this later), and so doesn’t affect the number of samples used in any of the experiment sets, including this one.

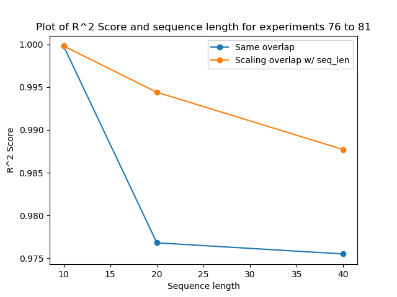
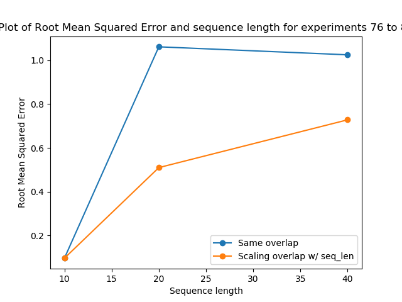
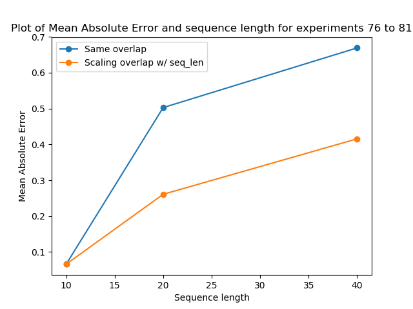
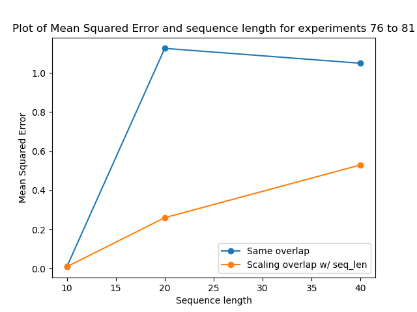
We have, up until now, been using ‘30’ as the default number of features. This is due to 30 features being the number of features that included over 99% of the variance when using the ‘variance threshold’; hence, this was a good default number of features to use for stat value files for any of the choices of feature reduction techniques. Also, it should be noted that the above results are for when the choice of feature reduction technique for ‘ft\_sel\_red.py’ was ‘pca’; that is, every result above (and also for every previous experiment conducted) has used PCA to reduce the dimensions of outputs of ‘comp\_stat\_vals.py’ to a smaller value. Further exploration of different feature selection/reduction techniques are explored at a later point, though PCA was chosen here as the default feature selection/reduction technique for other studies.

**Results Discussion and Conclusions**

Note that the above graphs are just for a single output type (the overall NSAA score) for reasons previously outlined and we are only interested in feature reduction for the statistical values from ‘comp\_stat\_vals.py’. From observing the above graphs, we can see a dramatic reduction in performance when we reduce the number of features from 30 to 10: we go from **MAE = ~0.1** to **MAE = ~0.17**, an increase in error of approximately 70%. This is most likely due to 10 features being not enough for PCA to capture the vast amount of the variance within the data it is given; hence, when we feed this data into RNN models, it is not as able to make accurate predictions of overall NSAA scores because it doesn’t have as complete a picture as if it’s using 30 features. Alternatively, if we increase the number of features from 30 to 50, we can see a notable worsening of performance (from **MAE = ~0.1** to **MAE = ~0.13**); hence, even though by using 50 features and therefore capturing more of the inherent variance and therefore ‘characteristics’ of the original data from ‘comp\_stat\_vals.py’, this is evidently not enough to overcome the consequences of using a higher-dimensional data for sequences to overcome the effects of the ‘curse of dimensionality’. Therefore, the following conclusions can be drawn:

1. **We shall continue with number of features to reduce to from data produced by ‘comp\_stat\_vals.py’ to 30.**
2. **This is the ideal ‘middle ground’ between sequences that don’t capture enough of the variance (e.g. number of features = 10) and sequences that have too many features to effectively train on (e.g. number of features = 50).**

**Experiment Set 8: Larger sequence lengths w/ overlaps for stat values from ‘AD’ files**



In the same way that we looked at increasing the sequence lengths used for raw measurements in experiment set 6, we now look at the same thing but from computed statistical values of source files by ‘comp\_stat\_vals.py’ and subsequently reduced to ‘30’ features. This time, however, we look at the effects of scaling the sequence overlap with sequence length (as done in experiment set 6) compared with not scaling the sequence overlap. We start with a sequence overlap of ‘0.9’ as standard (the value of which was determined by experiment set 4) for the initial default sequence length of 10. Note that the initial sequence length of 10 was chosen due to the initial lack of data available to use before the use sequence overlaps. Also, it’s worth pointing out that this length of 10 corresponds to a sequence capture window of 10 seconds (this is due to every ‘row’ of stat value data representing 1 seconds worth of data due to them being calculated over 60 rows of raw data sampled at 60Hz from the suit), unlike the raw data whose sequence length of 60 corresponds to a sequence capture window of 1 second. We also again assess only on the overall NSAA score output type for reasons previously discussed.

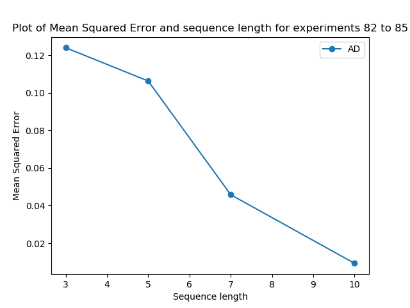
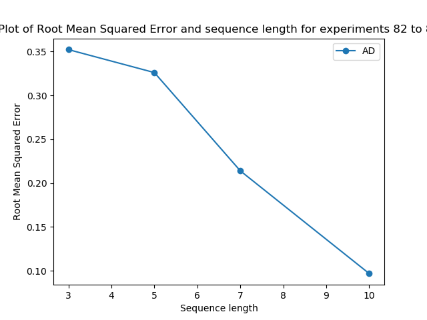
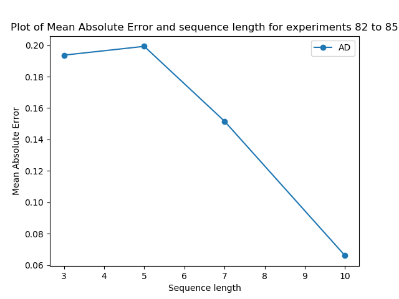
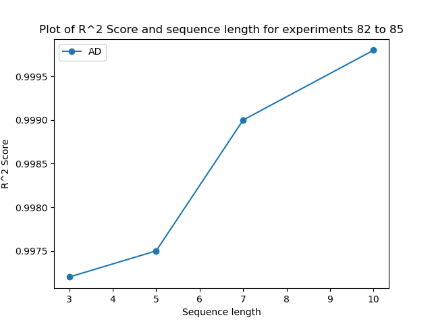
The primarily purpose of this experiment is to see if the performance of the models improved with a longer sequence length for computed statistical values of files. However, the secondary purpose is to determine if a scaling overlap improves the performance of the model, as was previously assumed for experiment set 6. This scaling overlap keeps the number of samples as a constant, as opposed to it decreasing due to the longer sequence lengths capturing more of the data per sequence. The experiments that have a scaling sequence overlap to keep the number of samples more-or-less constant are represented by the orange lines in the above graphs, while the ones with no scaling sequence overlap (and thus a reducing number of overall samples) are represented by the blue lines.

**Results Discussion and Conclusions**

We can see that in both scenarios where we either have a scaling sequence overlap or don’t, the performance of the model decreases with increasing sequence length. This is also significant because, for raw measurements, we can see that sequences are quite able to use longer sequence lengths and that ‘60’ seems to be an ideal point for these measurements. Therefore, from a smaller sequence length being more ideal for computed statistical values, we can conclude that this is a result of each row of data for computed statistical values containing more contextual data than a single row of raw measurement data, as opposed to it being down to the ideal data shape for an RNN to learn from (which is not the case as an RNN can learn quite well for a sequence length of ‘60’, as seen in experiment set 6). This is most likely due to the data contextual window growing too large. For example, when we increase the sequence length to 50 here, that means that each sequence takes in 50 seconds worth of source data, whch most likely isn’t able to account for the minutia of details that differentiates sequences of different classifications, overall NSAA scores, etc. Additionally, our reasoning of using a scaling sequence overlap in here and in experiment set 6 is justified, as for every metric above, the increased sequence length performs better with a scaling sequence overlap, as opposed to having a static sequence overlap of ‘0.9’, though in both cases it is still not as good as keeping the sequence length to 10. Therefore, we can conclude the following:

1. **Increasing the sequence length beyond 10 is not ideal for computed statistical values, which is shows that a contextual window of beyond 10 seconds makes learning for an RNN increasingly difficult**
2. **The use of a scaling sequence overlap shows better results than not using a scaling sequence overlap**

**Experiment Set 9: Smaller sequence lengths w/ overlaps for stat values from ‘AD’ files**

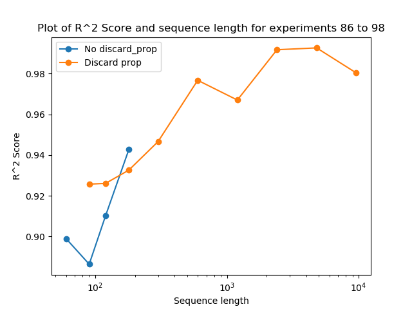
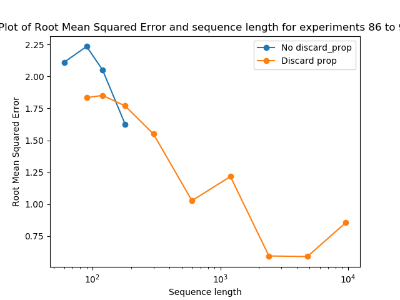
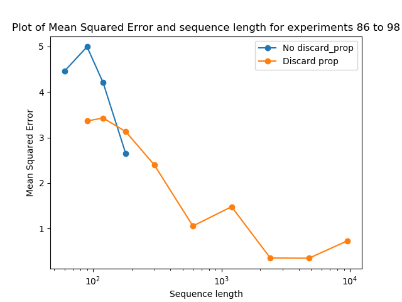
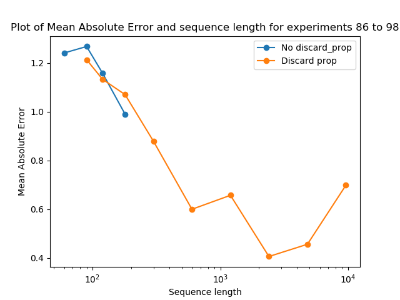
Before moving onto examining of larger sequence overlaps for raw measurements as opposed to just for extracted statistical values (as in experiment set 8), it’s necessary to ensure that a sequence length of 10 is indeed ideal for our setup, as the previous experiment set only concurred that a sequence length of 10 or smaller could be the ideal length. Here, we look at these smaller sequence lengths, going down to a sequence length of 3 (i.e. three sequences of vectors of 30 numbers are fed into the RNN which represents a context windows of 3 seconds). The expectation prior to undertaking these experiments was that a sequence length of 3 would be too short to draw time-contextual inferences from by the RNN when learning, while it was felt that 3 seconds might be too short to make accurate predictions for D/HC classifications, overall NSAA scores, etc., for many sequences. Note that everything else remains as it was in experiment set 8 (i.e. same source directory, same output type for the models, etc.), with the above graphs showing the results when we are using a scaling sequence overlap to keep the number of samples as a relative constant (i.e. decreasing the overlap as we decrease the sequence length), which corresponds to the orange lines in the graphs in experiment set 8.

**Results Discussion and Conclusions**

As we can see in the above graphs, the results evidently concur with our predictions: the performance for the overall NSAA score output type for models trained on extracted statistical values peak at a sequence length of around 10, with decreasing performance shown when we lower this value while keeping the number of samples relatively constant. As mentioned previously, this is most likely due to an RNN generally thriving on sequences of longer than 3 to get time-contextual information along with a context window of 3 seconds being not as useful as a 10 second window for NSAA scores. Furthermore, this knowledge that 10 seconds of context window for NSAA activity sequences is generally most useful will carry us over into the following experiment, where we shall attempt to use a similar context window to see if we get comparable results. Therefore, we can conclude the following:

1. **Smaller sequence lengths for models built on extracted statistical values showed worse performance, and therefore a sequence length of 10 for these sequences are ideal**

**Experiment Set 10: Very Large Sequence Lengths for Raw Measures w/ Discard Proportion**



Having established that a 10 second contextual window was ideal for extracted statistical values, we now wish to see if this is something that is exclusive to statistical values or whether it exists for raw measurements as well; if it does, then the 10 second context window is more likely to be an inherent characteristic of sequences made from NSAA assessment files, regardless of what measurement types are drawn from these files. For raw measurements, each row of data that is going into the RNN represents 1/60th of a second, as was previously established due to the sampling rate of the suit producing the data being 60Hz. Therefore, to get 10 seconds worth of data from raw measurements, we would need 600 rows of data (i.e. a sequence length of 600) while computed statistical values only need a sequence length of 10. However, setting sequence length to 60 for raw measurements presents us with two problems:

1. The increasing of sequence length for raw measurement data from 60 (as we have used as the default up until this point) to 600 will reduce the data 10-fold due to there being far more of the data needed for each of the sequences.
2. A sequence of several hundred has been previously established to be difficult for an RNN model to train on, along with much more computational demanding and containing a lot of possibly redundant data.

The first problem is solved by using a sequence overlap of 0.9 (for increasing from a sequence length of 60 to 600); this keeps the number of samples at a relative constant as we increase the sequence size. This ensures that any changes that occur in performance will be down to just sequence length and not the number of samples used; see experiment set 4 for a more in-depth discussion of this. However, this does not solve the other problem of too-long sequences to feasibly train the RNN model on. What we want is a way for an RNN model to gain a longer contextual window (e.g. of 10 seconds, corresponding traditionally to a sequence length of 600) without lengthening the sequence length itself. In this sense, any increase in performance while increasing the context window will be entirely down to increasing the RNN’s contextual information for a given sequence and not the actual sequence length. This is done by the ‘--discard\_prop’ optional argument supplied to ‘rnn.py’.

What this does is fairly simple: for each sequence, once it has been extracted from the source data block (also accounting for sequence overlap if appropriate), we keep only every ‘nth’ line of the sequence. For example, assume we have a sequence of length 600 and have set the ‘--discard\_prop’ argument to 0.9. This means that 90% of the data of the sequence is discarded, which results in keeping every 10th line in the sequence. This achieves two things:

1. Discards 540 rows and we are left with 60 rows of data which, as a result of taking the ‘kept’ rows at even increments along the sequence, means that we still have context covering the 600 rows.
2. By having even sampling of ‘kept’ rows of data, we also limit the possibility of missing any ‘important’ pieces of data that existing within the original 600 rows, as it’s likely that 1 or more of the kept lines would have captured some of this information.

We therefore use this technique to allow us to experiment with much larger sequence lengths (up to 9600) with corresponding discard proportions; these are shown by the orange lines in the graphs above. We also look at not using the discard proportions for increasing sequence lengths, though we can only go up to a certain sequence length before computational limits restrict us from continuing; for example, training models with data of a sequence length of 9600 with no discard proportion was expected to take between 3 and 4 days to train the model with the same number of epochs as the previous ones.

It’s worth noting that the sequence length is scaling for all experiments done here so the number of samples is always kept relatively constant. This is true for both the blue and orange lines above, as in both scenarios with a sequence length of 120, the discard proportion is set to 0.5. Finally, we only use one raw measurement here (joint angles) to experiment with longer sequence lengths. This is mainly due to previous experiment sets showing us that the performance of the useful 4 raw measurements are almost always completely in-line with each other with respect to increases or decreases in performances with changing independent variables. Therefore, to save on expensive experimentation time, we chose to evaluate the performance of models on only one raw measurement, safe in the knowledge that other raw measurements will most likely see the same change in performance.

**Results Discussion and Conclusions**

One thing we can see from the graphs above is the noticeable increase in performance (e.g. lowering of MAE on the test set) up to approximately a sequence length of 600. This is the point on the graph of the MAE metric where the discard proportion line shows an upwards inflection afterwards (i.e. when going up to a sequence length of 1200, the MAE gets higher); furthermore, past this point in the graph, the improvements are comparatively minimal for the extra time it takes to pre-processing the data. With very large sequence lengths and corresponding discard proportions, even though the amount of data that is input into the RNN models, it takes longer and longer to prepare the data. Therefore, a compromise was reached in the **sequence length of 600** being the best choice given the proportional discard proportion for how long it takes the model to compute.

This is also a good length to have as it means the ‘ideal’ (in the sense of model performance and time needed to compute) context window for raw measurements with a high discard proportion and scaling sequence overlap is 10 seconds, which is the same as for extracted statistical values. Another thing to note is that, for sequence lengths where we have results for both the blue and orange lines in the graphs, we see that the orange line (representing using a discard proportion) shows better results. This shows that, given the same sequence overlaps, when using a discard proportion to reduce the sequence length that is used to train the models, we see better model performance; this is most likely due to RNN models not performing for sequences of several hundred or longer. We can therefore conclude the following:

1. **Performance of models increases with higher sequence length and scaling discard proportion, with a good compromise of performance and computing cost being a sequence length of 600.**
2. **The ideal context window of 10 seconds for extracted statistical values is more-or-less replicated here, with the 10 second context window (sequence length of 600) being a high-performance parameter setting, and therefore 10 seconds is a good window for data from NSAA files in any measurement form.**
3. **Using discard proportion over not using it for identical sequence lengths shows better performance, which indicates that RNN models struggle to perform on sequences of too-high length.**

**Model Predictions Set 1: Natural movement files on models built on NSAA and walk files**

MAE between true and predicted NSAA scores over files (NSAA) = 4.66,

MAE between true and predicted NSAA scores over files (6minwalk-matfiles) = 5.03

Percentage of correct predicted file D/HC label (NSAA) = 75.92%

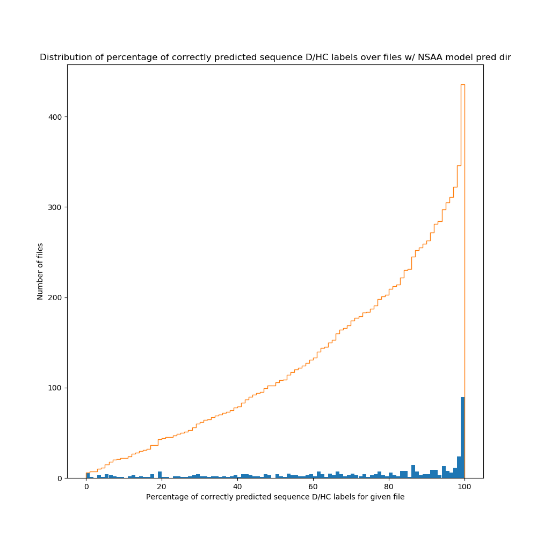
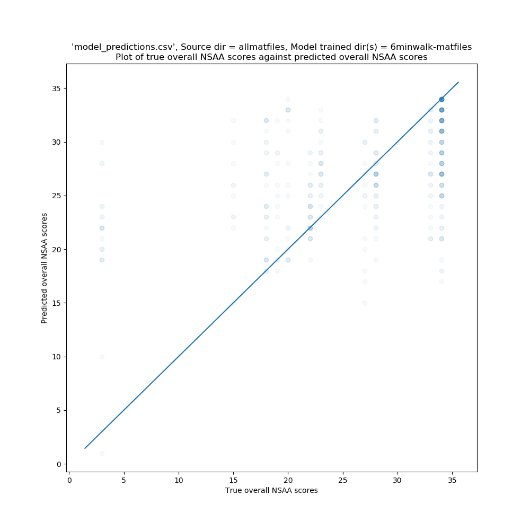
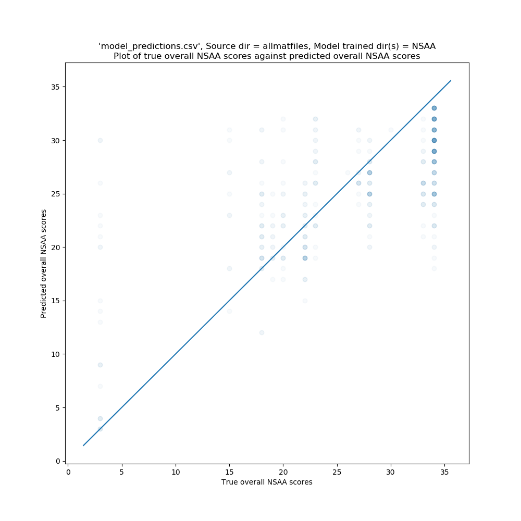
Percentage of correct predicted file D/HC label (6minwalk-matfiles) = 74.54%

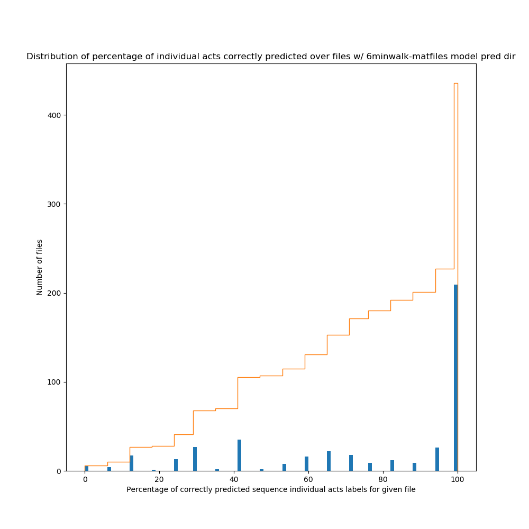
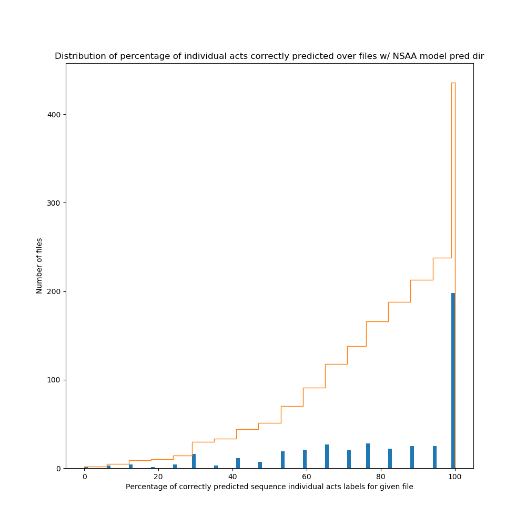
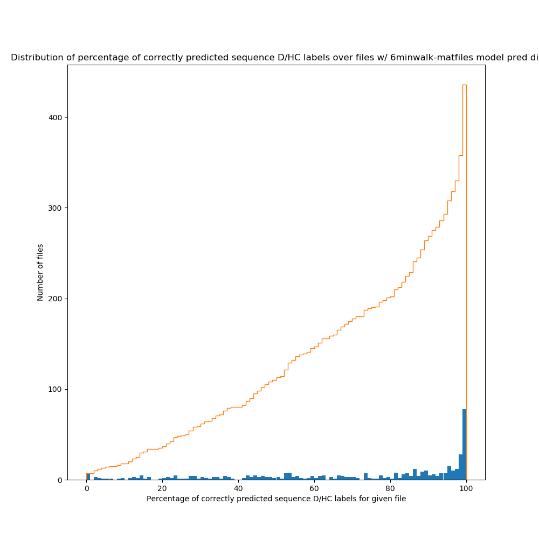
MAE of percentage predicted wrong sequence D/HC classification over files (NSAA) = 28.72

MAE of percentage predicted wrong sequence D/HC classification over files (6minwalk-matfiles) = 29.28

Average percentage of single acts correctly predicted over files (NSAA) = 80.84%

Average percentage of single acts correctly predicted over files (6minwalk-matfiles) = 75.28%





With the models now built that we will most likely be sticking with for the duration of the project (i.e. with an ideal sequence length, overlap, discard proportion, raw measurements used, etc.) ascertained by the previous experiment sets, we now move onto using these along with ‘model\_predictor.py’ to predict on whole files. Previously, in all experiment sets, the data that was reported was testing sequences from a mixture of the source files used to build the data set. This meant that assessment was done on a sequence-by-sequence basis, where the assessment of each sequence was to be independent of each other. With ‘model\_predictor.py’, this is quite different: we instead provide a specific name for the file, which will use all of the data to do an assessment on (the difference being that for the previous experiment sets, there will be a mixture of target D/HC classifications, overall NSAA scores, etc. for the test sequences, whereas with ‘model\_predictor.py’ there is a D/HC classification, overall NSAA score, etc. that is common among all of the sequences within the source file.

For making predictions on whole files via ‘model\_predictor.py’, the broad sequence of steps is as follows:

1. Split the source file into sequences, each with the same ‘y’ labels.
2. Assess each of these sequences on each of the models that we wish to be using, with a model for each measurement type and each output type (e.g. if we are assessing on 4 measurement types and 3 output types, there will be 12 models that each sequence is assessed on).
3. For each of the output types, average the response over all sequences for a given measurement and then average these over all measurements to get a single prediction for the given output type.

Each assessment of a file made by ‘model\_predictor.py’ is then stored as a single row of results within ‘model\_predictions.csv’. This is what’s used for the following model predictions sets.

For the first set, we are concerned with every file that we have of subjects involved in what we call ‘natural movement behaviour’; that is, data captured by the suit of the subject doing activities that aren’t NSAA or 6-minute walk assessments, such as playing or eating. This amounts to over 400 files, with as much as 30 files for a single subject captured. Hence, for each of these files, we run it through ‘model\_predictor.py’ to make various predictions for different output types. This is helped by the use of ‘test\_altdirs.py’, which automates a lot of the process and doesn’t require the user to manually run ‘model\_predictor.py’ >400 times.

While we shall get to assessing NSAA files on models built on NSAA files in the next section, here we wish to look at how well models that are built and tuned on NSAA files perform when presented with natural movement behaviour files. Obviously, the models have never seen any part of the files before, and the data it is now assessing on does not necessarily have characteristics of the original NSAA assessment files; for instance, the models will have been trained on sequences from files that do specific activities (i.e. one of the 17 NSAA assessment activities) or contain the subject walking. However, the natural movement behaviour will most likely be of movement characteristics that it’s never seen, and so the models are required to generalize their knowledge to other types of movement.

**Results Discussion and Conclusions**

**Console Output**

The following covers the model predictions of rows 1 to 436 of ‘model\_predictions.csv; for information on a prediction-by-prediction basis, see these rows. Instead of showing the results contained in these rows individually, however, we instead computed several statistics from these rows for certain columns with the aim to provide an insight into how well models build on NSAA and 6-minute walk files predict on natural behaviour files. It’s also worth noting that, at present, the natural movement behaviour files only exist as raw joint angle files. Hence, we the only raw measurement type of model we can use is the ones built on joint angle data (unlike predictions on NSAA files we shall discuss later, which uses 5 total data types for models).

The first two console outputs as seen above takes the average of the difference between each file’s true value for overall NSAA score and predicted value. This is done by, for each relevant row of ‘model\_predictions.csv’, finding the absolute difference between the true and predicted overall NSAA value columns; this gives a measure of how well the model predicted the overall NSAA score for that file, with this difference being 0 if it predicted the correct score; this is then averaged over all the relevant rows. What we can see is that models build on NSAA files that predict natural behaviour files predict a file on average within **4.66** of its true value, whereas models built on 6-minute walk files predict within **5.03** of its true value. These are reasonably strong initial results, though are possibly slightly skewed given the prevalence of files within the 25 to 34 score range. This is also including every single natural behaviour file available, and so the scores might be negatively impacted by natural movement files that are more-or-less not possible to infer an exact score (e.g. a file of a subject sitting very still for long periods).

The second two console outputs are the percentage of files that have the correct D/HC predicted classification. For each of the files, ‘model\_predictor.py’ takes the most common ‘D’ or ‘HC’ label for the sequences for the D/HC output type and makes that the prediction for the file; for example, for a file that contains 100 sequences, if 60 of them are predicted as being from a ‘D’ file and 40 are predicted as being from an ‘HC’ file, then the overall prediction for the file is a ‘D’ label. We also have a ‘true’ D/HC classification for the file; this is just based on which of the two the file’s name begins with. Hence, the percentage of correctly predicted D/HC labels is the percentage of rows with a predicted D/HC label that matches the true D/HC label. The results of **75.92%** and **74.54%** shown that in a majority of cases, natural movement files assessed on models built on either NSAA or 6-minute walk files get a correct classification; this is a notable good result that indicates, with further refinement and tuning, that these models would identify quite accurately what type of classification the subject is by simply observing natural movement data.

With the third set of console outputs, we are still concerned with analysing the D/HC classification predictions. Unlike the second set, for each file we aren’t concerned with the single estimated classification of the file by the models; rather, we look at it on a sequence by sequence basis in that we want to see the percentage of correctly classified sequences for the file being tested on. We then compute the ‘percentage of predicted wrong sequences’; this is simply the difference of the percentage of correctly classified sequences and 100%. For example, for ‘allmatfiles\D10-001’, we have 77.81% sequences predicted as being of ‘D’ label and 22.19% predicted as being of a ‘HC’ label. Since they are all supposed to be ‘D’ sequences (since they all came from a ‘D’ file), it therefore got 22.19% of the sequences wrong: this is the ‘percentage predicted wrong sequences’. We then repeat this over all other files from the natural movement behaviour data set and find the MAE of this set. This results in an error percentage of **28.72%** on models built on NSAA files and **29.28%** on models built on 6-minute walk files. This is fairly similar to the second set of console outputs in that it shows the models predict correct classes the majority of the time, though ideally this will approach 0% for both model types (NSAA and 6-minute walk) as we continue to refine and improve the system.

The final set of console outputs are more straightforward. For every predicted file (i.e. row we are concerned with in ‘model\_predictions.csv’), we get the value contained within the ‘Percent of acts correctly predicted’. This looks at the single act predictions for that file (i.e. a list of 17 values between 0 and 2 that the model believes are the correct single act scores for that file) and sees how many of those it got correct with respect to their true values; this then manifests as this ‘percent of acts correctly predicted’ score. We then repeat this over all the natural movement behaviour files and average this value to find the scores that appear on the fourth set of console outputs. The values of **80.84%** and **75.28%** are quite impressive: this means that on average the models that are trained on NSAA files predict ~13.7 of the 17 activities with the correct score, while the models trained on 6-minute walk files predict ~12.8 of the 17 activities. It’s also noted that particularly accurate scores for certain rows generally correspond with an overall NSAA prediction that is quite close to it’s true value, which makes sense as they are both involved with ascertaining NSAA scores; it’s just that the latter is predicting cumulative scores.

**Graph Output**

Along with the console output that is produced by ‘graph\_creator.py’, as discussed above, we also have several graphs that have been produced. However, in comparison to previous experiment sets, it is somewhat harder to ascertain specific results from these graphs, as these contain ~400 points and therefore must rely on the more evident ‘trends’ shown within the graphs. The first two show the true overall NSAA scores for each natural movement behaviour file along the x-axis and the files’ corresponding predicted overall NSAA scores along the y-axis by the NSAA models and 6-minute walk models. The closer these points are to the ‘y = x’ line projected through the middle of the plot the better, as this means they are closer to their true predicted values. From these graphs, we can see a tendency for points to hover around this line; however, this is still a great deal of variation around these areas. We can also see the model particularly struggles when presented with files that have an overall score of ‘3’; this is most likely due to these files being from an ‘outlier’ subject (due to the subject not completing many of the activities, which resulted in a lower score than they most likely should have). Additionally, we can see that files that have an overall score of ‘34’ (i.e. being from an ‘HC’ subject) are often assigned a score a fair bit lower of between 25 and 30. These observations from the two graphs for both types of models therefore give us good guidance of where to focus on improving the models.

The next two graphs show the distribution of the percentage of correctly predicted sequence D/HC labels for every natural movement behaviour file we tested on. This is essentially looking at either the ‘Percentage of predicted ‘D’ sequences’ or ‘Percentage of predicted ‘HC’ sequences’ for each file (depending on whether the file is a ‘D’ or ‘HC’ file). We then plot the distribution of these percentages for each of the files for both types of models, along with plotting the cumulative distribution lines. We can see from these graphs that there is high number of files where 100% of its sequences were predicted with the correct label, which is doubly impressive given the models these files are testing on have never seen natural movement behaviour before. It also shows us that, in both model types cases, there is a wide distribution of scores, with a not insignificant number of files having fewer than 50% of their sequences correctly classified (~100 files in both cases out of ~400 total). These graphs highlight that, while we have seen fairly positive results for the classification as shown on the second set of console outputs, there is still a lot of room for improvement.

The final two graphs again show a distribution of files. This time, however, we are looking at the distribution of percentages of correctly predicted sequence individual acts labels (which corresponds to the distribution of scores used to compute the fourth set of console outputs). Here, we can see a noticeable difference between the two types of models: while the models built on NSAA files have ~50 files that have <50% of the correctly predicted sequence individual acts labels, models build on 6-minute walk files have ~100 files that have <50% of the correctly predicted sequence individual acts labels. This is particularly impressive with regards to the NSAA models and shows the models proficiency to learn these individual scores. The disparity between the two model types also makes sense: the 6-minute walks don’t contain any NSAA activities within them (only the walk) and so we are asking the model to make predictions for the individual activities for a subject when the model itself never sees any of these files; it’s only presented with sequences of subjects with corresponding individual activities. Because it can’t as easily draw any inferences about walk data to correspond to other activity scores, it makes sense that it doesn’t perform as well compared with a model that does see these activities.

**Model Predictions Set 2: Model performance on left-out vs non-left-out files**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **File Name** | **Measurements tested** | **Percent of acts corrected predicted** | **Predicted D/HC Label** | **Percent of correct predicted sequences** | **True Overall Score** | **Predicted Overall Score** |
| D3 (already seen) | AD, Joint Angle, Joint Angle XZY, Position, Sensor Magnetic Field | 100% | D | 100% | 15 | 18 |
| D3 | AD, Joint Angle, Joint Angle XZY, Position, Sensor Magnetic Field | 23.53% | D | 80% | 15 | 26 |
| D3 | AD | 5.88% | D | 54.69% | 15 | 28 |
| D3 | Joint Angle | 29.41% | D | 97.1% | 15 | 27 |
| D3 | Joint Angle XZY | 23.53% | D | 81.7% | 15 | 26 |
| D3 | Position | 41.18% | D | 85.94% | 15 | 25 |
| D3 | Sensor Magnetic Field | 47.06% | HC | 35.49% | 15 | 22 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **File Name** | **Measurements tested** | **Percent of acts corrected predicted** | **Predicted D/HC Label** | **Percent of correct predicted sequences** | **True Overall Score** | **Predicted Overall Score** |
| D11 (already seen) | AD, Joint Angle, Joint Angle XZY, Position, Sensor Magnetic Field | 100% | D | 100% | 27 | 27 |
| D11 | AD, Joint Angle, Joint Angle XZY, Position, Sensor Magnetic Field | 76.47% | D | 100% | 27 | 27 |
| D11 | AD | 70.59% | D | 71.88% | 27 | 28 |
| D11 | Joint Angle | 82.35% | D | 65.78% | 27 | 26 |
| D11 | Joint Angle XZY | 82.35% | D | 74.53% | 27 | 25 |
| D11 | Position | 52.94% | D | 84.84% | 27 | 28 |
| D11 | Sensor Magnetic Field | 47.06% | D | 99.69% | 27 | 26 |

In this model predictions set, we look at 2 subjects (‘D3’ and ‘D11’) and look at how well models perform on predicting upon them when they have seen the files in training vs when they haven’t. These two subjects were chosen as their overall NSAA scores were mid-range (i.e. weren’t close to a perfect ‘34’ and weren’t outliers like ‘3’), though we intend to repeat this for numerous other subjects in the near-future, and each table represents the results concerning a single one of these two subjects. It should be noted that the models these files are tested upon are models built on NSAA files, along with the testing files being NSAA files themselves, so the models should be familiar with this sort of data.

We’ll first examine the results of the ‘D3’ subject. The first row is based on models that have already seen the ‘D3’ file in training and not only is it familiar with the subject, but it is also familiar in the particular data it is being tested on. This is only useful to us as a ‘baseline’ of how well the model could theoretically perform, as this is using some of the data that has been used for training for testing purposes. This therefore holds no practical use as a metric, as files that we would present to the models in a real-world setting would obviously not have been seen by the models before. Note that this uses all the measurements that we have decided upon using (extracted statistical values and the 4 ‘useful’ raw measurements) to make its estimations upon. Unsurprisingly, it predicts all of its sequences with the correct D/HC classification, along with all of all of the single act labels. However, it is still out on its estimation of the overall NSAA score by 3, which is somewhat surprising as not only should it be closer because the models are familiar with the file’s data, but the single-act scores are completely accurate; if these single act scores are correct, then should also have a overall score of a correct value as this is simply the accumulation of these values. This discrepancy of performance of different output types is possibly due to the single-act models being better trained to deal with sequences from this file; the cause of this is prompt for further investigation.

For the second line of the table, we then observe how well models predict on files that have been left-out completely of the training and testing process. This more accurately simulates what it would be like for the model in ‘production’; that is, assessing on new files in its intended capacity. We can see a noticeable steep drop in performance of the models: it only gets 23.53% of the single act predictions correctly, the number of correct sequence classifications drops 20% (though it still correctly determines the file to be a ‘D’ file) and the predicted overall NSAA score drops to being 11 away from the real score, down from 3. This is particularly poor generalization performance for this newly-seen file for the models and is heavy motivation to continue with generalization techniques to generalise the models’ learning inferences to new files. It’s worth noting that this was a particularly low score for ‘D’ files (with most falling in the +20 range), so this might contribute somewhat to its poor performance, though further investigation into other ‘left out’ files will be conducted later to confirm this.

The next 5 lines of the table again look at models that have had the ‘D3’ subject ‘left out’ of the model training (done by setting the ‘rnn.py’ optional argument to ‘--leave\_out=D3’). However, instead of using all 5 measurements for the three output types and aggregating their predictions, we instead look at individual measurements in turn; that is, a measurement is extracted from ‘D3’ (e.g. joint angle) and is tested on the three that correspond to this measurement (with ‘D3’ being left out of all of them) for each output type. The aim with this is to determine whether or not any particular measurements are more useful for generalizing to new, unseen files. At this point, the results from these rows are fairly inconsistent: while models built from the position and sensor magnetic field measurements perform better with respect to the single act scores output, the joint angle and joint angle XZY measurements perform better with respect to the D/HC classification output type. Additionally, there is not a noticeable disparity of improvement for the overall NSAA score among the measurement (with the exception of sensor magnetic field). Hence, at this stage, this is not enough information to draw any conclusions from and more of these ‘left out’ models for other subjects are needed.

Much of the same conclusions can be drawn for the ‘D11’ subject table. For the models that are trained on all measurements when seeing ‘D11’ as a left-out file (i.e. the 2nd row of the table), however, the models are a lot better at predicting more accurate outputs: there is a much higher percent of acts correctly predicted, higher percent of correct predicted sequences, and a closer true overall NSAA score compared with that of ‘D3’; here, the model predicts the ‘D11’ overall NSAA score correctly, rather than in the case of ‘D3’ where it is off by 11. This is particularly impressive for ‘D11’, as these models obviously have never seen this subject before. A likely cause for this increase in accuracy, however, is that ‘D11’ is closer to the mean overall NSAA score amongst the subject groups: ‘D11’ is more representative of the ‘average’ subject with Duchenne. Again, however, when we look at the left-out models but only single-measurements (rows 3 to 7 of the table), we again cannot drawn any particular conclusions with respect to one measurement or another being more useful to model generalization of unseen data; again, more of these left out subjects (rather than just ‘D3’ and ‘D11’) are most likely needed to draw any conclusions in this regard.