CS230

Predicting Daily Fantasy Basketball Scores Using Sequential Deep Learning

Name: William Denton
SUNet ID: wdenton
Department of Computer Science
Stanford University
wdenton@stanford.edu

Name: Sri Jaladi
SUNet ID: sjaladi
Department of Computer Science
Stanford University
sjaladi@stanford.edu

Abstract

Due to its ability to capture previous data and effectively impact future predictions based on this past data, recurrent neural networks (RNNs) have gained a lot of traction. Similarly, the gated recurrent unit (GRU) and long-short-term-memory (LSTM), versions of RNNs, have become increasingly common in forecasting situations. In this project, we research and explore utilizing sequential deep learning methods (and general deep learning methods) to help forecast fantasy basketball scores for players on a given game. Our project displays the potential that lies in general RNNs for fantasy basketball score forecasting.

1 Introduction

Our project is predicting basketball players' future performance (in terms of fantasy points). As NBA fans, we sought to apply advances in deep learning to this highly variable and high data task to see if computers could identify trends where humans are incredibly inconsistent and often perform quite poorly. Companies such as ESPN or DraftKings often rely on very basic formulas, betting trends, or by-hand evaluations for their projections. In this project, we sought a different approach, namely to use sequence based deep learning methods (RNN, GRU, LSTM) to predict a basketball player's fantasy score in their next game. Specifically, using the player's last 9 games, each of which had 20 features (statistics captured each game), we predict the stats and/or the fantasy points of that same player on the 10th game (the next sequential game following that 9 game input). With this approach, we create models which seek to use pure statistical performance data to derive statistical knowledge about how a player will perform in the future. The end goal is for this model is to be deployed to predict the over/under on a player's fantasy points for a single game and to be accurate enough to beat the gambling odds.

2 Related work

There exists some related work in this field. Some AI/ML projects either predict good players for the ENTIRE season (Hermann, Ntoso), don't utilize deep learning techniques (Skandan), or use data on the cumulative level and DON'T include game by game data (Young, Koo, Gandhi). The referenced papers essentially solve one of the tasks we tackle but not all simultaneously. In addition, all of the listed related works do NOT use sequential learning models. While cumulative data (Young et al.) proved to be effective, we believe we can improve upon these results with microscopic, sequential data. ESPN Forecast, considered the best in this field, utilizes 200 experts with extensive NBA backgrounds (ESPN). Using ESPN Forecast as an estimation for Bayes error shows that many existing models fall short in efficacy despite Bayes error being large (ESPN forecast isn't overly accurate). In addition, there has been some work in optimizing fantasy basketball lineups, but not necessarily predicting a specific fantasy score (Earl). Our project, however, pursues a different path as it operates under the goal of trying to accurately compute over/under fantasy point values for individual players.

3 Dataset and Features

The data we used was collected from web-scrapers and NBA stat sites, specifically from basketball-reference.com (Sports Reference LLC). We elected to focus on the regular season performance of every player every season. To collect the data, we used two scrapers (Agartha, Bradley). We compiled a list of every active NBA player from the year 2000 to the year 2022, and downloaded their performance into a CSV which stored that player's games and

CS230: Deep Learning, Spring 2022, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

performance (see figure 2 in Appendix 9.1 for an example player season). We transformed each season into data to input into our model by grabbing 10 game sequences from every season (making each example a 20x10 array due to 20 features, see transposed version in figure 3 Appendix 9.1). For stats-evaluating models, the 10th game's stats (see transposed version in figure 4 Appendix 9.1) is the true output, and for end-to-end models the 10th game's fantasy score is the true output. In total, our dataset has around 400,000 9 game series WITH a single true output. That dataset was split into a 20,000 size test set and a 380,000 size training set (95/5 split). We focused on just a training and test set because exploring different models was a higher priority than hyper-parameter tuning, and thus we sought to explore which types of sequential models worked best using a simple train and test set.

4 Methods

Due to minimal previous work in utilizing neural networks for fantasy score predictions, we implement many different models to identify an optimally suited approach. We experiment with many different models to gauge what would be successful and to identify the reasoning why. Resulting from our hypothesis that a player's last few games would have an impact on their performance in the next game, a majority of the models that we utilize are forms of recurrent neural networks (RNN).

A recurrent neural network works by processing a single full input (in our case full inputs consisted of 9 consecutive games) and producing an output (or outputs). This process consists of processing a single game, using a hidden layer, and a fully connected end layer to get the final output. The output of the hidden layer is then passed on to the next layer to be used in combination with the next game in its hidden layer. This process continues until the entire chain (of 9 past game inputs) is complete. The reason that we chose to focus primarily on sequential models like this was because we wanted to create a model that would take into account the previous games of the player and have future predictions be impacted by the results over those games. An RNN model (of different sorts) is perfect for this as each hidden layer continues to pass down information so that previous games continue to play an impact into future predictions. Furthermore, with sequential models we hoped to identify "hot" (a player doing well) and "cold" (a player doing badly) streaks.

When discussing end-to-end versus stats-evaluating models, end-to-end means that the model was built to take in the players past games and output a single fantasy score prediction for the player's next game. Stats-evaluating means the model was built to take in the player's past games and output a prediction for all stats that will impact fantasy score (points, assists, blocks, shots made, etc.) for the 10th game. In both cases, the 10th game is used as the "truth" value, or the target output (whether it's the fantasy score or the exact stats).

There were many different combinations of models and hyper-parameter strategies that we tried. In order to try and get as much of our results, realizations, and conclusions into the paper, we have elected to elaborate and display 6 of these models that were the more representative versions over testing. Note that justification for these models can be found in the results section. The 6 models we utilized were:
Stats-Evaluating RNN Model (from scratch), End-to-End RNN Model, End-to-End GRU Model, Stats-Evaluating LSTM Model, End-to-End LSTM Model, Stats-Evaluating Deep Neural Network

It is important to note that we utilized different loss functions depending on whether the model was an end-to-end model or whether it was a stats-evaluating model. We utilized mean squared loss (L2Loss) for stats-evaluating models and utilized average L1Loss (absolute error) for end-to-end models.

L1Loss for a single batch was defined as the following:
$L1Loss = \frac{1}{n}\sum_{i=1}^{n} |\hat{y}^{(i)} - y^{(i)}|$
Note n represents the number of games in a single batch (usually we utilized stochastic gradient descent, meaning $n = 1$), $\hat{y}^{(i)}$ represents the predicted fantasy scores for the ith example in batch and each $y^{(i)}$ represents the true fantasy scores from the ith example. The graphed L1Loss during training graphs the average loss for all 9 processed outputs that occur during a single training example. During testing, only the final prediction (using past 9 games to predict the 10th game) is used for consistency in testing and to match the goal of the project. This leads to some end-to-end models to have training loss higher than their respective testing loss, but it is very expected.

L1Loss was used for end-to-end models because this loss is utilized to get as many loss values as close to 0 as possible while having an equal impact on large errors/loss values. In other words, using L1Loss, larger loss values are penalized equally (relatively) to smaller values. While mean squared loss is far more common for regression tasks, we specifically chose L1Loss because it fit our project's end goals better. Since our model tries to predict a player's fantasy point value for a single game to make an over/under bet (i.e. try to bet whether or not a player would get more or less than the betting fantasy value), our general goal is to try and get as many games as close as possible to the true value. That way, we can maximize certainty on maximum number of bets. Errors of 100 and 50 are both giving a losing bet, and their result is the same. There is no need to try and have 100 have a significantly

stronger pull on weights as it is an outlier and incorrect bet (just like 50) anyway. Additionally, there are many players who earn small points per game so mean squared loss on fantasy points would mean a very small subset of games played by a very small subset of players (who earn lots of points on average creating larger errors) would heavily impact our overall loss, which is detrimental to our goal.

Mean squared error loss (MSELoss or L2Loss) for a single batch was defined as the following:
$MSELoss = \frac{1}{n}\sum_{i=1}^{n}\sum_{s=1}^{k}(\hat{y}_s^{(i)} - y_s^{(i)})^2$
Once again, n represents the number of examples in a batch. Furthermore, $\hat{y}_s^{(i)}$ represents the prediction for stat s from the ith example in the batch and $y_s^{(i)}$ represents the true value of stat s (out of a total k stats being predicted) in example i of the batch.

Note that L2Loss was used for stats-evaluating models because in this case, our model is better off avoiding large errors in stats predictions as opposed to maximizing very small errors. Because there are numerous stats to factor in, large errors in stats can heavily skew/hurt the entirety of fantasy predictions, also allowing for less generalizability.

Additionally, the activations used are part of methods. There is a very intentional reason that all presented models (except for two) utilize ReLU activation. Our primary goal was to try and test out different activation functions with the idea that some may work better than others. ReLU works in that it sets values lower than 0 to 0 and keeps the magnitude of values > 0 the same. Sigmoid and tanh DON'T do this, as they have small ranges (0 to 1 and -1 to 1 respectively) and don't perform well when computing larger stats. Further explanation on this is given in results.

Further, normalization was often used on the entire dataset during data pre-processing. When using normalization, we found the mean and stdDev of training data, then normalized stats with this mean and stdDev. A majority of models (aside from 2 for reasons specified later) employed data normalization as this essentially limited the magnitude/range/scope of the problem we chose to tackle.

5 Experiments/Results/Discussion

We are going to start with the overall results that were attained on the testing set by the 6 different large models that we trained. This is evaluated based on the average error (average L1 error) of fantasy point predictions versus the actual value for the testing set and is sorted from largest average error to smallest (with a single exception discussed further): The rest of the results section will start at the very top (worst model with highest error) and move

Model Type	Output/Prediction Type	Activation	Avg. Error (Avg. L1Loss)	Median Error (Median L1Loss)
RNN (From Scratch)	Stats-Evaluation	Sigmoid	18.497090854795672	N/A
GRU	End-to-End	Sigmoid	12.476802931127647	10.642890930175781
LSTM	End-to-End	ReLU	132468.80970362743	10.616925
Deep Neural Network	Stats-Evaluation	ReLU	10.27458460935271	7.7081577479839325
LSTM	Stats-Evaluation	ReLU	10.0295637536733	7.589242309331894
RNN (PyTorch)	End-to-End	ReLU	7.683919785579018	5.962127685546875

Figure 1: Testing set average/median errors of different models

down (best model with lowest error). At each step, each model will be discussed along with its results (qualitative and quantitative), justification, and overall conclusions that can be drawn from this model's results (alone and in comparison with other models). As general notes, we experimented with different batch sizes and found that a batch size of 1 was most successful most of the time because it gave the most insight into our models and their respective performance if we continued to optimize based on a single RNN example (10 game example). However, for deeper and more complex models, we found that a batch size of 64 was more robust against outlying player performances during convergence. Further, learning rates change for each different model, but were determined by identifying how long-run loss operated (lowering with higher oscillations and increasing with lack of convergence). A great amount of emphasis of our project, however, is on the models, activations, strategies, and insights as opposed to specific hyperparameters that work well.

5.1 RNN (Baseline) Model ⇒ (Avg. L1Loss = 18.497090854795672, All corresponding graphs/data in Appendix 9.2.1):

The RNN baseline model is an RNN model that we built from scratch using only Numpy libraries. This model is a simple RNN model that has 0 hidden layers and passes the output of the previous layer to the next layer. This model is a stats-evaluating model meaning that the exact stats are output. Further, optimization is done through gradient descent and the activation used throughout was sigmoid. The overall justification for this was that we wanted to create a baseline model. We explore a model that utilizes sequential deep learning without any improvements or optimizations (the most template/base possible version), which is why we did it from scratch.

As this model was used mainly as a comparison/baseline, the results for this model are just as we expected-this model had the worst error and worst performance on the test set. As a note, the utilization of sigmoid creates another issue which causes the next model to also perform very poorly (discussed during GRU model).

5.2 GRU Model ⇒ (Avg. L1Loss = 12.476802931127647, Median L1Loss = 10.642890930175781, All corresponding graphs/data in Appendix 9.2.3):

The GRU model was built using PyTorch and utilized a tanh and sigmoid activation for remembering/forgetting information along with the hidden layer. Further, the GRU model had a single hidden layer, normalized data, predicted on an end-to-end basis, and utilized the Adam Optimizer. The justification for this was that we wanted to utilize a GRU model (as a simplified version of LSTM) that would NOT use ReLU (like the rest of our models) to weight the impact of different activations.

The predominant reason we did not extensively tune this model was that the performance of the model was very poor, and the reasoning for this performance precluded improvement. While the average error of about 12.5 and median error of 10.6 don't seem bad, the problem with this model can be plainly seen in Figure 17 in Section 9.2.3 of the Appendix. From this picture (and the data), we noticed that the GRU model was only outputting a single constant output value, no matter what the inputs were. This problem (which is also present in the baseline RNN model), occurs due to the activation functions.

Each statistic between players deviated only slightly from the average, but each fantasy score could be significantly further from the average. The goal of our project was to take more tightly distributed player stats and predict a more widely distributed fantasy score. The issue with sigmoid and tanh is that their range is bounded heavily (0 to 1 and -1 to 1 respectively). Thus, very large values are heavily reduced. For example, a sigmoid/tanh activation on a fantasy prediction of 100 is going to be seen nearly the same as a prediction of 30 even though these values are very different. In other words, sigmoid and tanh restrict and tighten the distribution of outputs when we need to expand it or constant that distribution. Trying to compensate, we normalized the data to try and tighten values, but this did not change the end result. Thus, to overcome this hurdle and avoid this issue, we used ReLU whose bounds go from 0 to ∞. Since ReLU (when positive) has no bound, it allows for negative values to become 0 while maintaining larger positive values' magnitude. Using ReLU in the other models fixed this issue of predicting a single value.

5.3 LSTM Model (End-to-End) ⇒ (Avg. L1Loss = 132468.80970362743, Median L1Loss = 10.616925, All corresponding graphs/data in Appendix 9.2.6):

This End-to-End LSTM model was built using Keras and works in a similar manner to the GRU model in that it also utilizes activations to remember, forget, and modify its internal hidden information that gets passed on. However, this model uses ReLU activation between layers, uses the Adam Optimizer, and has 4 hidden layers. This was one of the two LSTM models that was built and was implemented with the goal of identifying when players were on hot or cold streaks.

Looking alone at the average error for this model, it is easy to conclude that this model performs very poorly. However, on further inspection, there was a single test case that was outputting a very high magnitude value (likely caused from over/under flow during the model's propagation). This caused the entire error to be extremely high, when in reality this occurred very rarely. Further, as previously mentioned, for our project's goals, we need as many errors close to 0 while errors greater than a certain large value are all treated as equal. In that sense, this outrageous example is just one bad example and does NOT reflect this overall model. In order to account for this, we utilized median error (because it gives a metric of all errors, more robust to outliers) on this model along with others. When using median error as the comparison metric, this model performs the 4th best.

5.4 Deep Neural Network Model ⇒ (Avg. L1Loss = 10.27458460935271, Median L1Loss = 7.7081577479839325, All corresponding graphs/data in Appendix 9.2.4):

The stats-evaluating deep neural network we utilized took in the past 9 games and outputted the stats prediction for the 10th game. Further, this model had 5 hidden layers (dropping by a factor of $\frac{1}{2}$ in nodes), used the Adam Optimizer, a batch norm layer, and ReLU activation. The main reasoning for this model was to get a control or baseline for our entire task. While the RNN baseline is meant to be a baseline for sequential models, this model serves as a baseline for deep learning solutions to predicting fantasy scores. Further, we had 5 hidden layers decreasing from 512 nodes to eventually reach 12 (number of stats that impact fantasy score) while also having a batch norm layer.

Quantitatively, this model performs the third best. This indicates that only well-applied sequential learning models outperform basic deep learning and that sequential learning alone is not the answer to our task. Of the models discussed so far, this is the first with reasonable avg and median errors and varying outputs based on the input.

Finally, of the next 3 models, this model's outputs were the most condensed (smallest output range), as can be seen in Figure 21 of Appendix 9.2.4.

5.5 LSTM Model (Stats-evaluating) ⇒ (Avg. L1Loss = 10.0295637536733, Median L1Loss = 7.589242309331894, All corresponding graphs/data in Appendix 9.2.5):

The stats-evaluating LSTM we used was implemented using Keras and would take in past 9 games, run through an LSTM model and output stats predictions for the 10th game. This model utilized the Adam Optimizer, 4 hidden layers (same as the other LSTM model for consistency), normalized data, and ReLU activation. These parameters were primarily chosen to remain as consistent as possible with the previous LSTM model and the deep neural network to compare with these two models. This model was meant to be a more improved version of each of these and so comparing to each was important.

The loss and performance of this model on the testing set was only marginally better than the deep neural network, again indicating that sequential learning models alone cannot solve this problem. In fact, enlarging the model or making it more complex does not necessarily create better performance either (explored in next section). However, this model had the second best performance quantitatively. Further, the training loss of this model can be seen to be much less than the deep neural network but the overall results are the same. This points to this model overfitting on less varying stats (like blocks) because its L2Loss on stats (during training) is low, but its end fantasy prediction loss is similar to the deep neural network.

5.6 RNN (PyTorch/improved) Model ⇒ (Avg. L1Loss = 7.683919785579018, Median L1Loss = 5.962127685546875, All corresponding graphs/data in Appendix 9.2.2):

The final, and best performing model, is a general end-to-end RNN model with a single hidden layer, ReLU activation, unnormalized data, and Adam Optimizer. The hyper-parameters were chosen in order to provide benefits over the baseline RNN model that was used. After testing with normalized and un-normalized data (just like other models), and doing tuning on the learning rate, we achieved this model.

Clearly, this model has by far the best performance out of the previous models, despite being rather simple in comparison. When compared to the GRU or LSTM models, this model does not have any gates or probabilities to keep track of or forget information. Further, this model only uses a single hidden layer. The large-scale reasoning for this model being good is its generalizability. The pitfall that larger models fall into (for this task) is that they are making connections and identifying patterns which are not actually helpful. In the case of the LSTM models, it is likely that they are keeping track of information from several games ago which likely has no impact on the next game. Another reason for this performance is that more layers (through activation and weights) often means the distribution becomes tighter and tighter. As discussed earlier, trying to predict fantasy scores is means expanding the distribution, not tightening it, giving a potential justification as to why a model with 1 hidden layer performed much better than models with 4 or more.

As our best found/tested model, we can explore some interesting resulting observations. To start, we note that the distribution of the scatterplot from figure 11 in Appendix 9.2.2 is much more condensed and tight (especially around the red "truth" line), but also has a wider range on the x-axis in comparison to other models. This is a general indication that the model is more successful in its predictions. In addition, this model had a larger prediction range than any other model. Finally, about 80% of all errors were under 10 points, which is the same as the best average error for other models (can be seen in figure 9 in Appendix 9.2.2).

6 Conclusion/Future Work

Overall, the best model that we identified for our task was a simple end-to-end RNN model with a single hidden layer. The results display that creating a successful fantasy score predicting model requires the model to be very generalizable and requires the model to be able to spread rather than squeeze through forward propagation. In addition, the fact that larger and more complex models were handily beaten indicates that this task, to be solved extremely well, requires significantly more complexity and/or a very strong background in fantasy scores and game trends.
In addition, median error was a better gauge for our model's real-world performance as it removed the skewed nature of average loss (similar to the L1Loss vs. L2Loss discussion earlier). Finally, our project and research displayed that, for this and similar tasks, it is vital to use ReLU activation due to the importance of the output range of the chosen activation. Models without ReLU could not discern any differences in inputs and performed by far the worst.

The foremost manner to build upon this project for future work would be to develop a model that is trained to give an optimal gambling strategy based on player predictions. Additionally, more research can and should be done in trying to utilize more classic machine learning methods and unsupervised learning mechanisms due to the discovered noisy nature of neural networks on this task.

7 Contributions

We both contributed an equal amount to the project as a whole. As freshmen, this was our first time taking on a project of this scope and so a majority of the balancing was done by taking on each other's roles when the other had other commitments. In that sense, all the data scraping, processing, and organization was done by Will. In addition, Will developed the LSTM models and the Deep Neural Network model that was discussed. Sri developed the baseline RNN model, the improved RNN model, the GRU model, and the testing mechanism. In terms of earlier reports (for the milestone and proposal), we both would simultaneously work on the document to maximize our productivity. During the final project writeup, we split up drafting sections, then both reviewed, revised and edited the entire report. Will drafted the introduction, data, contributions, and references section and Sri drafted the other portions of the writeup. In addition, Will managed the preparation, recording, setup, and processing of the final video.

8 References

Abadi, Martx27;in, Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., ... others. (2016). Tensorflow: A system for large-scale machine learning. In 12th $USENIX$ Symposium on Operating Systems Design and Implementation ($OSDI$ 16) (pp. 265–283).

Agartha, V. (n.d.). Basketball Reference Scraper: A python module for scraping static and dynamic content from basketball reference. GitHub. Retrieved May 30, 2022, from https://github.com/vishaalagartha/basketball_reference_scraper

Bradley, J. (n.d.). Basketball Reference Web Scraper: NBA stats API via basketball reference. GitHub. Retrieved May 30, 2022, from https://github.com/jaebradley/basketball_reference_web_scraper

Chollet, F., others. (2015). Keras. GitHub. Retrieved from https://github.com/fchollet/keras

Earl, James, "Optimaztion of Fantasy Basketball Lineups via Machine Learning" (2019). Senior Honors Theses. 836. https://digitalcommons.liberty.edu/honors/836

ESPN Internet Ventures. (2013, April 15). The gold standard for predictions. ESPN. Retrieved May 27, 2022, from https://www.espn.com/nba/story/_/page/ESPN-Forecast-track-record-130415/nba-espn-forecast-proved-most-accurate-prediction-system

Harikrishnan, V.K., Deore, H., Raju, P., Agrawal, A., Sharma, M.M. (2021). Predictive Analysis Using Machine Learning Techniques for Fantasy Games. In: Manik, G., Kalia, S., Sahoo, S.K., Sharma, T.K., Verma, O.P. (eds) Advances in Mechanical Engineering. Lecture Notes in Mechanical Engineering. Springer, Singapore. https://doi.org/10.1007/978-981-16-0942-8_65

Hermann, Eric, and Ntoso, Adebia. "Machine Learning Applications in ... - cs229.Stanford.edu." CS229.Stanford, Stanford, https://cs229.stanford.edu/proj2015/104_report.pdf.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., ... Chintala, S. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. In Advances in Neural Information Processing Systems 32 (pp. 8024–8035). Curran Associates, Inc. Retrieved from http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

Skandan, Shreya S. Learning to Turn Fantasy Basketball Into Real Money Introduction to Machine Learning. https://shreyasskandan.github.io/Old_Website/files/report-ChanHuShivakumar.pdf.

Sports Reference LLC. Basketball-Reference.com - Basketball Statistics and History. https://www.basketball-reference.com/.(dateofyourvisit)

Young, Connor, and Koo, Andrew, and Gandhi, Saloni et al. Final Project: NBA Fantasy Score Prediction - Connoryoung.com. Cornell Tech, https://connoryoung.com/resources/AML_FinalProject_Report.pdf.

9 Appendix/Graphs/Data

9.1 Pictures of Input/Output Data

active	date	points_scored	plus_minus	team	location	opponent	outcome	seconds_played	made_field_goals	attempted_field_goals	made_three_point_field_goals	attempted_three_point_field_goals	made_free_throws	attempted_free_throws	offensive_rebounds	defensive_rebounds	assists	steals	blocks	turnovers	personal_fouls	game_score
TRUE	2017-10-17	22	9	GOLDEN STATE WARRIORS	HOME	HOUSTON ROCKETS	LOSS	1773	8	18	3	9	3	3	0	5	4	1	0	2	4	14.9
TRUE	2017-10-20	28	7	GOLDEN STATE WARRIORS	AWAY	NEW ORLEANS PELICANS	WIN	2107	7	16	4	11	10	10	0	3	8	0	0	1	1	24.7
TRUE	2017-10-21	27	-6	GOLDEN STATE WARRIORS	AWAY	MEMPHIS GRIZZLIES	LOSS	1805	9	17	6	11	13	13	0	6	3	0	0	2	5	28.6
TRUE	2017-10-23	29	22	GOLDEN STATE WARRIORS	AWAY	DALLAS MAVERICKS	WIN	1869	7	15	2	10	13	13	2	0	8	4	0	6	1	25.9
TRUE	2017-10-25	30	14	GOLDEN STATE WARRIORS	HOME	TORONTO RAPTORS	WIN	2251	9	20	4	11	8	8	0	4	5	3	1	2	2	25.2
TRUE	2017-10-27	20	-8	GOLDEN STATE WARRIORS	HOME	WASHINGTON WIZARDS	WIN	2172	7	19	3	13	3	3	0	5	8	2	0	2	2	15.8
TRUE	2017-10-29	27	12	GOLDEN STATE WARRIORS	HOME	DETROIT PISTONS	LOSS	2128	11	17	3	7	2	3	1	5	8	1	0	5	3	21.7
TRUE	2017-10-30	31	21	GOLDEN STATE WARRIORS	AWAY	LOS ANGELES CLIPPERS	WIN	1754	9	14	7	11	6	8	2	3	6	2	1	1	3	31.0
TRUE	2017-11-02	21	16	GOLDEN STATE WARRIORS	AWAY	SAN ANTONIO SPURS	WIN	1911	7	13	3	4	4	4	1	7	5	3	0	2	2	21.2
TRUE	2017-11-04	22	44	GOLDEN STATE WARRIORS	AWAY	DENVER NUGGETS	WIN	1778	7	14	5	10	3	4	0	0	11	2	0	3	0	21.3
TRUE	2017-11-06	16	25	GOLDEN STATE WARRIORS	HOME	MIAMI HEAT	WIN	2033	5	19	2	9	4	5	0	5	4	2	0	1	3	9.4
TRUE	2017-11-08	22	16	GOLDEN STATE WARRIORS	HOME	MINNESOTA TIMBERWOLVES	WIN	1786	7	16	2	7	8	8	0	8	8	2	1	5	2	18.5
TRUE	2017-11-11	22	39	GOLDEN STATE WARRIORS	HOME	PHILADELPHIA 76ERS	WIN	1794	5	12	3	8	9	9	0	4	9	1	0	2	0	22.1
TRUE	2017-11-16	9	-14	GOLDEN STATE WARRIORS	AWAY	BOSTON CELTICS	LOSS	1864	3	14	2	9	1	2	2	4	5	4	0	4	5	4.1
TRUE	2017-11-18	35	3	GOLDEN STATE WARRIORS	AWAY	PHILADELPHIA 76ERS	WIN	2079	11	22	4	9	9	9	0	5	5	0	1	2	1	27.3
TRUE	2017-11-19	36	4	GOLDEN STATE WARRIORS	AWAY	BROOKLYN NETS	WIN	1853	14	24	4	10	7	8	3	8	7	3	0	7	6	30.4
TRUE	2017-11-22	24	-17	GOLDEN STATE WARRIORS	AWAY	OKLAHOMA CITY THUNDER	LOSS	1849	9	18	3	8	3	3	1	4	6	1	0	6	1	15.7
TRUE	2017-11-24	33	39	GOLDEN STATE WARRIORS	HOME	CHICAGO BULLS	WIN	1811	10	18	4	11	9	9	0	7	4	2	0	2	2	28.6
TRUE	2017-11-25	27	5	GOLDEN STATE WARRIORS	HOME	NEW ORLEANS PELICANS	WIN	2158	9	25	3	13	6	6	0	4	6	1	0	1	2	17.7
TRUE	2017-11-29	28	4	GOLDEN STATE WARRIORS	AWAY	LOS ANGELES LAKERS	WIN	2364	9	26	3	9	7	8	0	5	7	0	0	5	4	17.0
TRUE	2017-12-01	23	19	GOLDEN STATE WARRIORS	AWAY	ORLANDO MAGIC	WIN	1863	9	20	3	8	2	2	1	5	10	3	0	4	2	20.0
TRUE	2017-12-03	30	20	GOLDEN STATE WARRIORS	HOME	MIAMI HEAT	WIN	1819	11	18	5	9	3	4	0	2	3	2	0	6	1	27.1
TRUE	2017-12-04	31	-1	GOLDEN STATE WARRIORS	AWAY	NEW ORLEANS PELICANS	WIN	2067	9	19	5	11	8	9	2	3	11	1	0	2	6	27.9
TRUE	2017-12-30	38	21	GOLDEN STATE WARRIORS	HOME	MEMPHIS GRIZZLIES	WIN	1535	13	17	10	13	2	2	0	4	3	0	0	0	2	33.8
TRUE	2018-01-02	32	7	GOLDEN STATE WARRIORS	AWAY	DALLAS MAVERICKS	WIN	2369	11	19	6	13	4	5	0	5	8	1	0	3	1	27.4
TRUE	2018-01-04	29	3	GOLDEN STATE WARRIORS	AWAY	HOUSTON ROCKETS	WIN	2211	10	20	4	10	5	6	0	9	5	1	0	3	3	27.6
TRUE	2018-01-06	45	28	GOLDEN STATE WARRIORS	AWAY	LOS ANGELES CLIPPERS	WIN	1751	11	21	8	16	15	16	2	4	3	3	0	2	2	39.2
TRUE	2018-01-08	32	3	GOLDEN STATE WARRIORS	HOME	DENVER NUGGETS	WIN	2050	9	17	5	10	9	10	1	3	9	2	0	4	3	28.0
TRUE	2018-01-10	24	-5	GOLDEN STATE WARRIORS	AWAY	TORONTO RAPTORS	WIN	2368	8	12	2	7	10	12	0	6	9	1	0	1	3	24.1
TRUE	2018-01-15	23	13	GOLDEN STATE WARRIORS	AWAY	CLEVELAND CAVALIERS	WIN	2004	8	15	4	8	3	4	1	3	8	2	0	3	3	20.3
TRUE	2018-01-17	30	21	GOLDEN STATE WARRIORS	AWAY	CHICAGO BULLS	WIN	2355	10	18	6	11	4	4	0	9	4	1	0	6	2	27.1
TRUE	2018-01-20	19	-11	GOLDEN STATE WARRIORS	AWAY	HOUSTON ROCKETS	LOSS	2171	8	20	5	15	2	4	0	1	8	2	0	6	2	7.7
TRUE	2018-01-23	32	4	GOLDEN STATE WARRIORS	HOME	NEW YORK KNICKS	WIN	1899	9	19	8	15	8	8	0	6	7	0	0	1	2	27.2
TRUE	2018-01-25	25	20	GOLDEN STATE WARRIORS	HOME	MINNESOTA TIMBERWOLVES	WIN	2245	9	14	5	9	2	3	2	3	8	3	0	2	2	27.2
TRUE	2018-01-27	49	6	GOLDEN STATE WARRIORS	HOME	BOSTON CELTICS	WIN	2208	10	24	8	13	9	10	0	4	5	2	0	1	2	43.1
TRUE	2018-01-30	14	-17	GOLDEN STATE WARRIORS	AWAY	UTAH JAZZ	LOSS	1719	4	13	1	7	5	5	1	1	5	0	0	3	1	7.9
TRUE	2018-02-02	22	14	GOLDEN STATE WARRIORS	AWAY	SACRAMENTO KINGS	WIN	2125	7	17	3	10	8	8	1	4	9	2	0	4	3	14.8
TRUE	2018-02-05	24	4	GOLDEN STATE WARRIORS	AWAY	DENVER NUGGETS	LOSS	2137	8	16	5	12	3	4	1	5	4	2	0	5	2	16.8
TRUE	2018-02-08	21	-12	GOLDEN STATE WARRIORS	HOME	OKLAHOMA CITY THUNDER	LOSS	1869	6	14	2	9	7	7	1	4	5	1	1	3	3	16.0
TRUE	2018-02-09	30	3	GOLDEN STATE WARRIORS	HOME	DALLAS MAVERICKS	WIN	1761	7	12	4	9	2	2	2	5	6	0	0	4	1	18.5
TRUE	2018-02-10	17	6	GOLDEN STATE WARRIORS	HOME	SAN ANTONIO SPURS	WIN	1732	7	13	3	7	0	0	1	3	8	2	0	2	0	17.9
TRUE	2018-02-12	33	21	GOLDEN STATE WARRIORS	HOME	PHOENIX SUNS	WIN	1793	8	17	3	9	3	3	2	7	7	0	1	3	0	18.2
TRUE	2018-02-14	17	-7	GOLDEN STATE WARRIORS	AWAY	PORTLAND TRAIL BLAZERS	LOSS	2566	8	17	3	10	2	2	2	2	6	3	0	7	4	8.1
TRUE	2018-02-22	44	8	GOLDEN STATE WARRIORS	HOME	LOS ANGELES CLIPPERS	WIN	2229	14	19	8	11	8	8	1	5	10	2	0	4	4	41.9
TRUE	2018-02-24	21	24	GOLDEN STATE WARRIORS	HOME	OKLAHOMA CITY THUNDER	WIN	1759	8	18	5	12	0	0	1	8	6	3	0	2	2	18.4
TRUE	2018-02-26	21	25	GOLDEN STATE WARRIORS	AWAY	NEW YORK KNICKS	WIN	1817	8	16	4	9	1	1	0	5	5	0	0	3	1	14.9
TRUE	2018-02-28	25	20	GOLDEN STATE WARRIORS	AWAY	WASHINGTON WIZARDS	WIN	2107	5	15	4	9	11	12	0	5	3	2	0	6	3	14.5
TRUE	2018-03-01	35	2	GOLDEN STATE WARRIORS	AWAY	ATLANTA HAWKS	WIN	1429	8	15	4	9	8	8	0	1	3	2	0	0	0	35.1
TRUE	2018-03-08	34	11	GOLDEN STATE WARRIORS	HOME	BROOKLYN NETS	WIN	2038	12	20	6	12	4	5	1	5	4	3	1	3	2	29.3
TRUE	2018-03-08	2	-1	GOLDEN STATE WARRIORS	HOME	SAN ANTONIO SPURS	WIN	143	0	1	0	0	2	2	0	0	0	0	0	0	0	1.3
TRUE	2018-03-23	29	9	GOLDEN STATE WARRIORS	HOME	ATLANTA HAWKS	WIN	1522	10	18	3	8	5	5	1	6	1	0	1	1	1	22.9

Figure 2: Stats per game data utilized

active	date	points_scored	plus_minus	team	location	opponent	outcome	seconds_played	made_field_goals	attempted_field_goals	made_three_point_field_goals	attempted_three_point_field_goals	made_free_throws	attempted_free_throws	offensive_rebounds	defensive_rebounds	assists	steals	blocks	turnovers	personal_fouls	game_score
TRUE	2017-10-17	22	9	GOLDEN STATE WARRIORS	HOME	HOUSTON ROCKETS	LOSS	1773	8	18	3	9	3	3	0	5	4	1	0	2	4	14.9
TRUE	2017-10-20	28	7	GOLDEN STATE WARRIORS	AWAY	NEW ORLEANS PELICANS	WIN	2107	7	16	4	11	10	10	0	3	8	0	0	1	1	24.7
TRUE	2017-10-21	27	-6	GOLDEN STATE WARRIORS	AWAY	MEMPHIS GRIZZLIES	LOSS	1805	9	17	6	11	13	13	0	6	3	0	0	2	5	28.6
TRUE	2017-10-23	29	22	GOLDEN STATE WARRIORS	AWAY	DALLAS MAVERICKS	WIN	1869	7	15	2	10	13	13	2	0	8	4	0	6	1	25.9
TRUE	2017-10-25	30	14	GOLDEN STATE WARRIORS	HOME	TORONTO RAPTORS	WIN	2251	9	20	4	11	8	8	0	4	5	3	1	2	2	25.2
TRUE	2017-10-27	20	-8	GOLDEN STATE WARRIORS	HOME	WASHINGTON WIZARDS	WIN	2172	7	19	3	13	3	3	0	5	8	2	0	2	2	15.8
TRUE	2017-10-29	27	12	GOLDEN STATE WARRIORS	HOME	DETROIT PISTONS	LOSS	2128	11	17	3	7	2	3	1	5	8	1	0	5	3	21.7
TRUE	2017-10-30	31	21	GOLDEN STATE WARRIORS	AWAY	LOS ANGELES CLIPPERS	WIN	1754	9	14	7	11	6	8	2	3	6	2	1	1	3	31.0
TRUE	2017-11-02	21	16	GOLDEN STATE WARRIORS	AWAY	SAN ANTONIO SPURS	WIN	1911	7	13	3	4	4	4	1	7	5	3	0	2	2	21.2
TRUE	2017-11-04	22	44	GOLDEN STATE WARRIORS	AWAY	DENVER NUGGETS	WIN	1778	7	14	5	10	3	4	0	0	11	2	0	3	0	21.3

Figure 3: One full example (used in train AND test)

TRUE	2017-10-17	22	9	GOLDEN STATE WARRIORS	HOME	HOUSTON ROCKETS	LOSS	1773	8	18	3	9	3	3	0	5	4	1	0	2	4	14.9

Figure 4: Single Game Data (inputted at each recurring level)

9.2 FULL RESULTS DATA:

9.2.1 Stats-Evaluating Baseline RNN Model:

Training data:

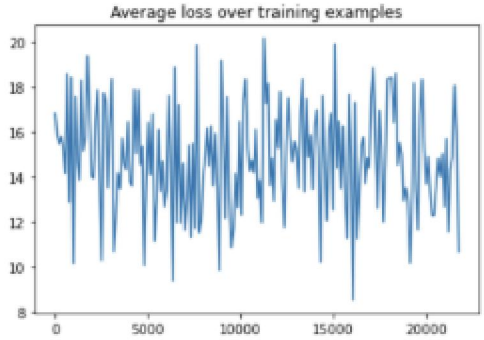

Figure 5: Avg L1Loss over training

Testing data: Average L2loss over all test examples (error on all stats combined): 18.497090854795672

9.2.2 End-to-End RNN Model:

Training data:

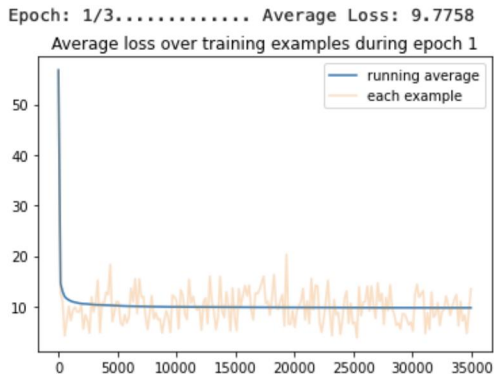

Figure 6: Avg L1Loss over epoch

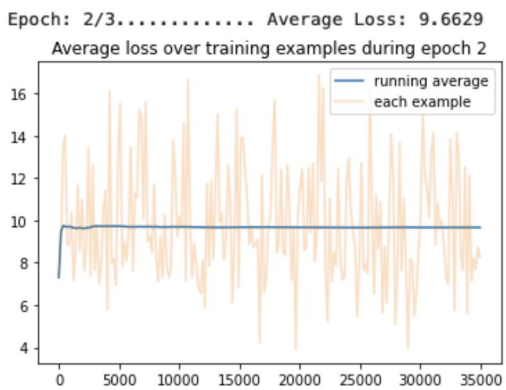

Figure 7: Avg L1Loss over epoch

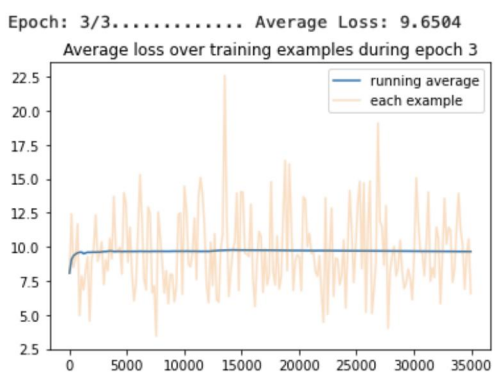

Figure 8: Avg L1Loss over epoch

Testing data:

Average l1 loss: 7.683919785579018
Median l1 loss: 5.962127685546875
Average l2 loss: 103.70863571214198
Median l2 loss: 35.546966538764536

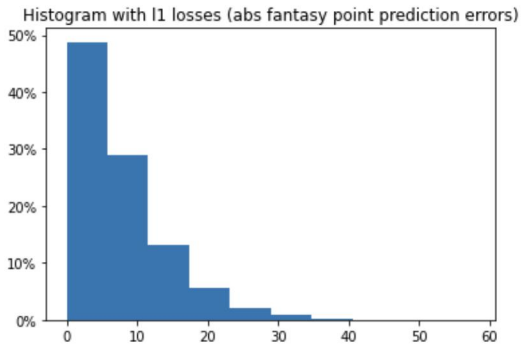

Figure 9: Average L1Loss Histogram

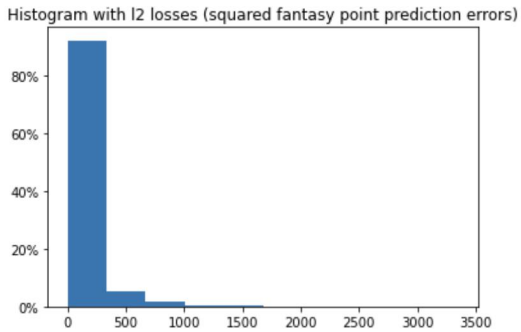

Figure 10: Average L2Loss Histogram

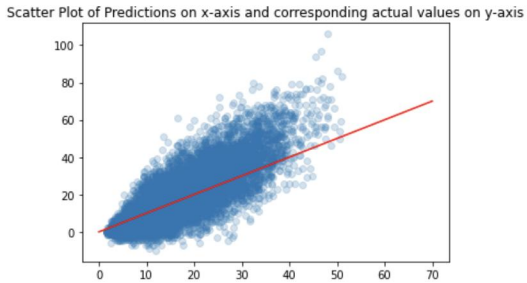

Figure 11: Scatterplot of predictions vs actual values

9.2.3 End-to-End GRU Model:

Training data:

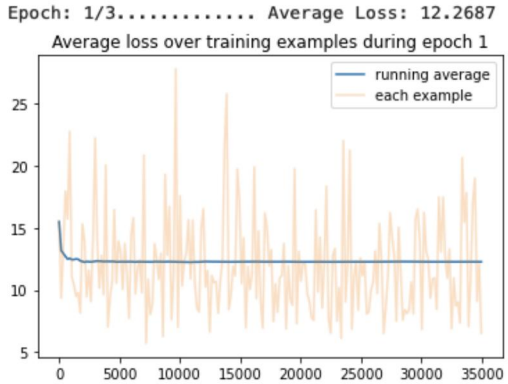

Figure 12: Avg L1Loss over epoch

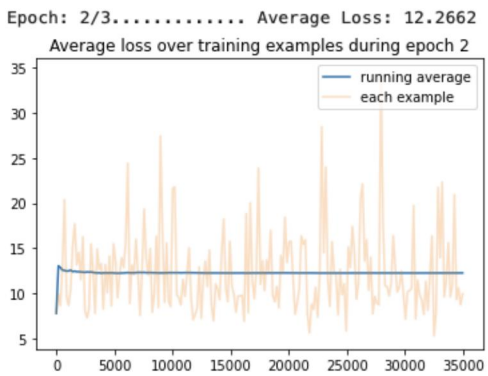

Figure 13: Avg L1Loss over epoch

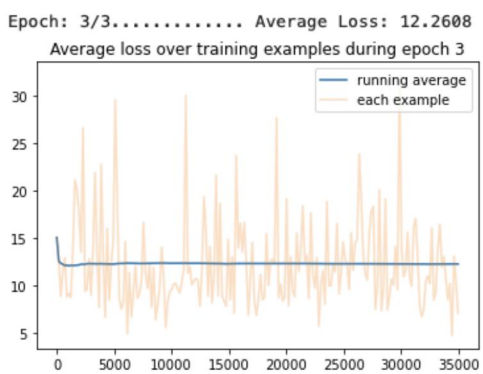

Figure 14: Avg L1Loss over epoch

Testing data:

Average l1 loss: 12.476802931127647
Median l1 loss: 10.642890930175781
Average l2 loss: 251.06187881570733
Median l2 loss: 113.2711273516179

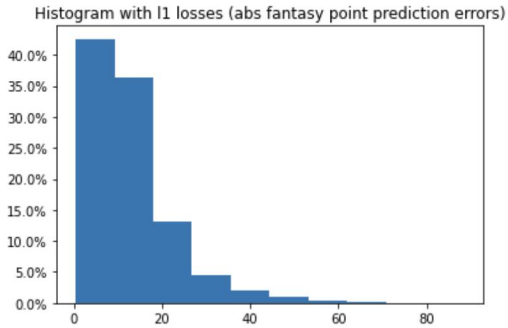

Figure 15: Average L1Loss Histogram

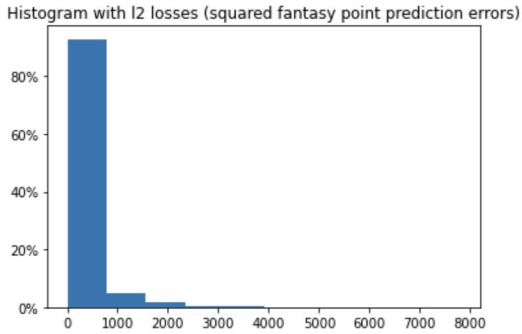

Figure 16: Average L2Loss Histogram

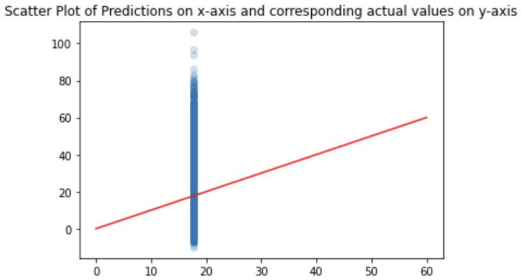

Figure 17: Scatterplot of predictions vs actual values

9.2.4 Stats-Evaluating Fully Connected Neural Network:

Training data:

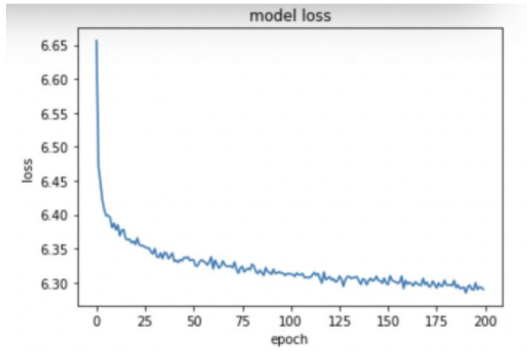

Figure 18: Avg L2Loss over epoch

Testing data:

Average l1 loss: 10.27458460935271
Median l1 loss: 7.7081577479839325
Average l2 loss: 187.05158908281052
Median l2 loss: 59.41569586780473

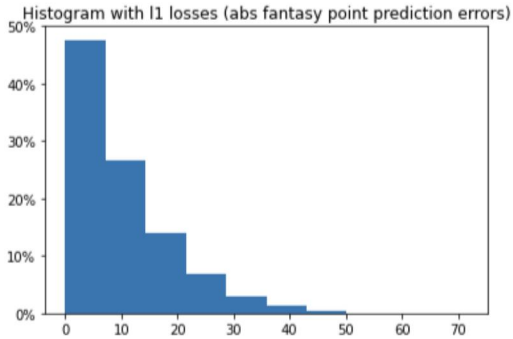

Figure 19: Average L1Loss Histogram

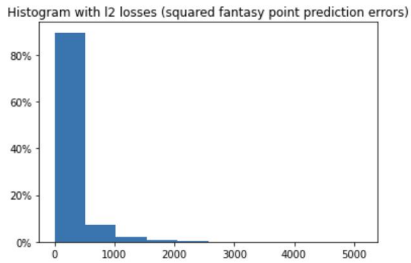

Figure 20: Average L2Loss Histogram

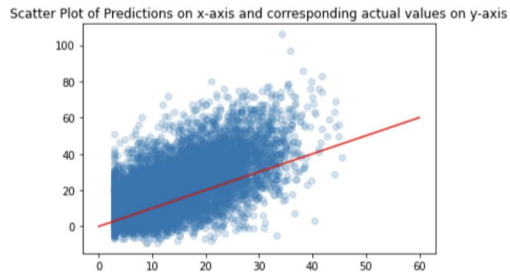

Figure 21: Scatterplot of predictions vs actual values

9.2.5 Stats-Evaluating LSTM Model:

Training data:

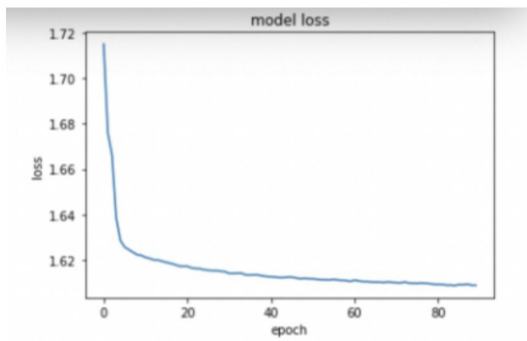

Figure 22: Avg L2Loss over epoch

Testing data:

Average l1 loss: 10.0295637536733
Median l1 loss: 7.589242309331894
Average l2 loss: 176.89190820961775
Median l2 loss: 57.5965988297533

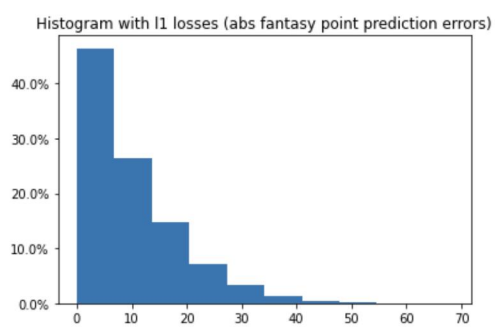

Figure 23: Average L1Loss Histogram

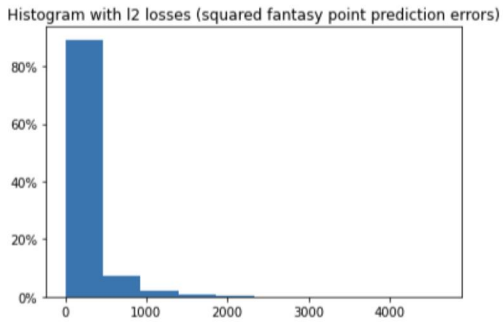

Figure 24: Average L2Loss Histogram

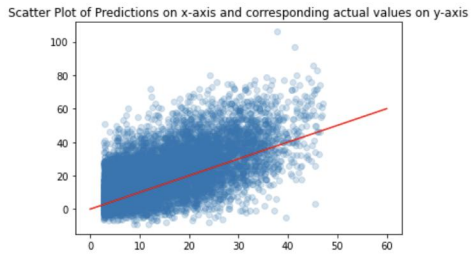

Figure 25: Scatterplot of predictions vs actual values

9.2.6 End-to-End LSTM Model:

Training data:

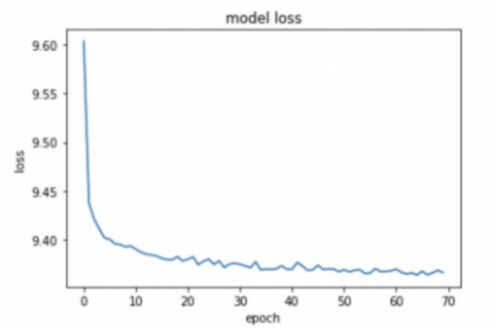

Figure 26: Avg L1Loss over epoch

Testing data:

Average l1 loss: 132468.80970362743
Median l1 loss: 12.616925
Average l2 loss: 3213672.667353244
Median l2 loss: 159.1868

Due to some errors being too large, graphs were not able to be made