Final Project Report: Softball Swing Classifier

By: Riya Nandakumar

DGMD S-14 Summer 2020

Table of Contents

1: Introduction

The Softball Swing Classifier strives to be a solution to the daunting task of learning to play baseball/softball. One of the first needed skills when beginning to play the sport is to know how to swing a bat; this is easier said than done, as it can take decades for a player to perfect the technical motion needed to optimize one's batting. Even certain professional players tweak components of their swing regularly, in order to achieve peak performance. This machine learning classifier helps to flatten the steep learning curve by analyzing one component of a player's swing: their back leg's rotation. In baseball/softball, a majority of a swing's power comes from the lower body. This can be broken down into two components, of the step into the swing and the back leg's pivot, which together help to optimize rotational power, allowing the player to hit the ball farther than before. The back leg's pivot, in particular, is the main source of rotation in a swing; many coaches teach this to young children through the phrase, "squash the bug," as they imagine a bug beneath their back foot, and aim to squash it into the ground.

The image above of Bryce Harper, one of the best baseball players in the MLB, illustrates the proper rotation in his back leg of his swing; his swing has more rotation than the average, yet it still has great technique. This motion does not come without its hindrances, however, since many beginners can develop habits which inhibit proper rotation. Three common issues with the back leg's pivot are under-rotating the leg and thus having little power, over-rotating the leg and thus having a wild swing, and losing balance and picking up the leg from its pivot point. The aim of this classifier is to identify these three common issues, as well as good swing technique, in order to facilitate greater swing improvement in beginner baseball/softball players. My hope with this project is that it will allow absolute beginners in the sport to dive head first into the game I and millions of others love, with no fear of initial failure at the skills needed. Additionally, this project aims to help coaches identify players in need of extra help in their swing, allowing teams to progress at a greater rate.

2: Project Description

This project, in its current state, utilizes the app SensorLog off the Apple App Store (costs $4.99).

Here, the trained and tested CreateML neural network model is uploaded, connected to the proper features, and used to classify motions in a player's swing; SensorLog has the ability to record many different sensors, such as core location and audio, but for current purposes, the sensors of interest are the accelerometer, gyroscope, and device motion (which contains its own accelerometer and gyroscope). Using the data recording methods provided by the app, one can record the outcomes of the model and utilize this for further practicing and self-analysis. However, the goal for the future of this model is to move to a separate app, of mine own

creation, in order to have a simple interface which any person can use on their own or with a team; this app is not completed, as I have no prior experience with Swift and iOS app development, and this will be covered further in the "Moving Forward" section of this report. Additionally, further details about the methods of data collection and methods of recording swings will be discussed in the next section, "Development Process." However, for the time being, this is the current state of the classification system.

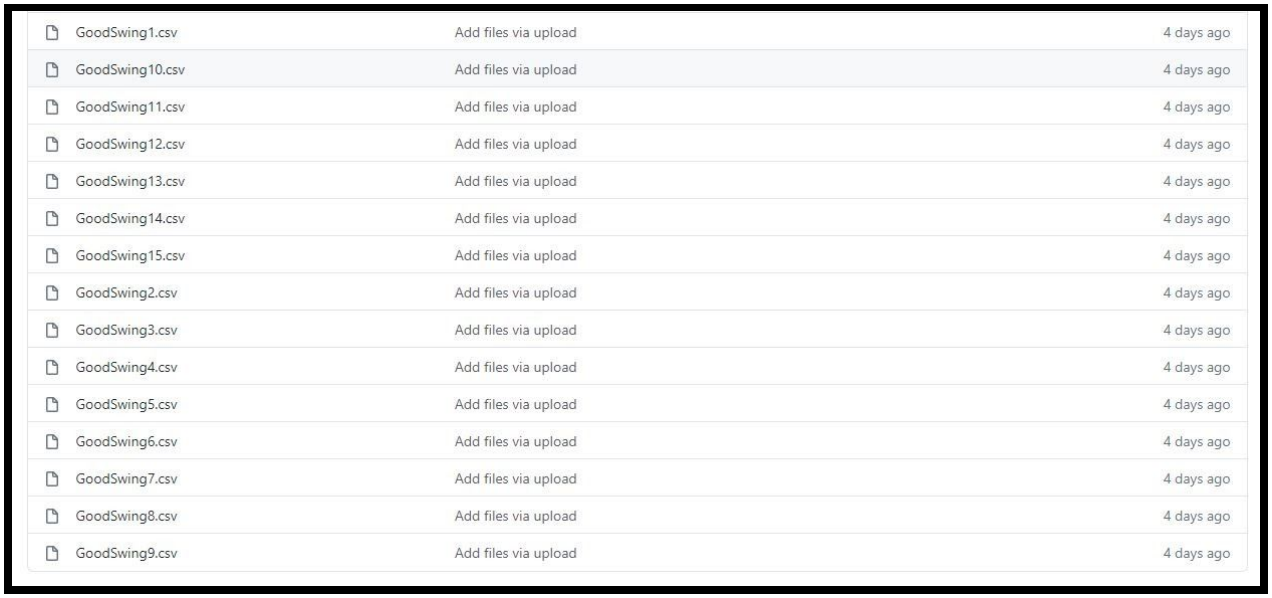
3: Development Process -- Data Collection

The process of data collection was also facilitated by the SensorLog application. In it, one can record data to a .csv or .json file, and choose which sensors to record. For the purposes of this project, I initially utilized the accelerometer and gyroscope, with it recording to a .csv; later on in the development process, I experimented with recording device motion, as this would be beneficial for the development of the iOS app and syncing those sensors, yet it didn't become fully realized in the timespan given. Each dataset recorded consisted of 80 swings, with 20 of each swing type; these 80 swings would be broken down into 40 right-handed and 40 left-handed. These swings would also be broken down into 60 for the training set, and 20 for the testing set. The manner of recording these swings involved the iPhone, with SensorLog up and running, attached to the side of the back leg's calf with a hairband.

Before each swing, a button in the SensorLog app would be pressed to begin recording data, and after the swing, this button would be pressed again to stop recording data. A demonstration of this process can be found in the video in the repository.

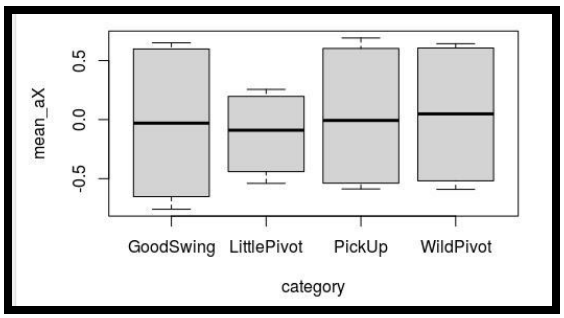
-- Data Examination and Feature Extraction

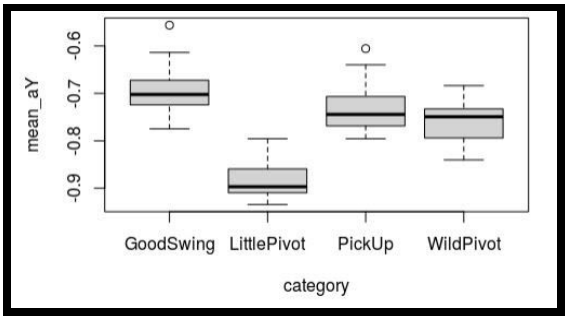
After data collection finished (although there were numerous rerecorded datasets made in the middle of project construction), each data file would be sent via email to the MacBook facilitating the model training and testing. For the purposes of data examination, each file had to be renamed to contain the type of swing it is; these data files can be found in the repository, but an example of this is in the image below.

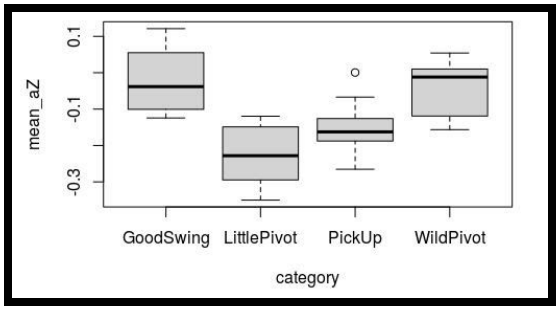


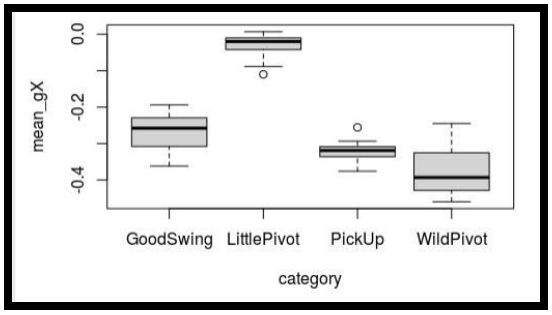
GoodSwing1.csv	Add files via upload	4 days ago
GoodSwing10.csv	Add files via upload	4 days ago
GoodSwing11.csv	Add files via upload	4 days ago
GoodSwing12.csv	Add files via upload	4 days ago
GoodSwing13.csv	Add files via upload	4 days ago
GoodSwing14.csv	Add files via upload	4 days ago
GoodSwing15.csv	Add files via upload	4 days ago
GoodSwing2.csv	Add files via upload	4 days ago
GoodSwing3.csv	Add files via upload	4 days ago
GoodSwing4.csv	Add files via upload	4 days ago
GoodSwing5.csv	Add files via upload	4 days ago
GoodSwing6.csv	Add files via upload	4 days ago
GoodSwing7.csv	Add files via upload	4 days ago
GoodSwing8.csv	Add files via upload	4 days ago
GoodSwing9.csv	Add files via upload	4 days ago

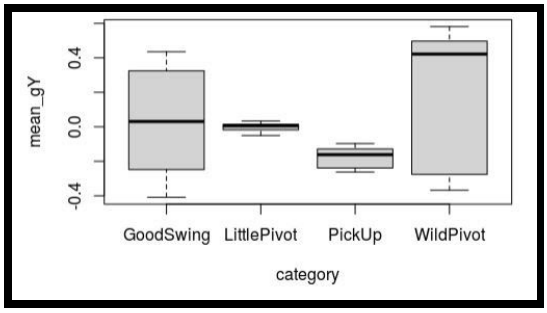
For the second (main) dataset and the fourth dataset (for device motion), projects in RStudioCloud were created to access the differences in mean accelerometer and gyroscope values across the four swing types; the actual R code can be found in the repository. For each accelerometer and gyroscope axis, a boxplot would be generated showing that specific swing type's distribution of mean values. Standard deviation was not utilized, as due to the black-box nature of CreateML, I felt it was better to focus purely on the differences in values, rather than variation; for certain features, standard deviation could have proved a better differentiating factor, but I tried to steer clear of this path. Below are the boxplots generated for the second (main) dataset, and the features I decided to use from these results are the accelerometer's Z axis and the gyroscope's X axis.

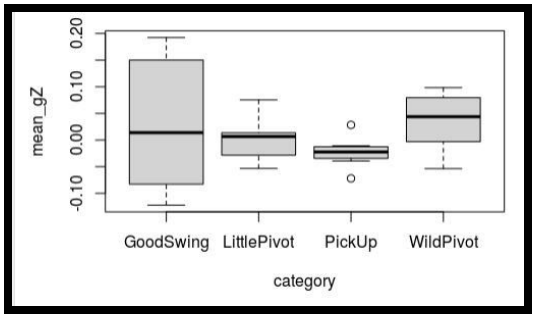




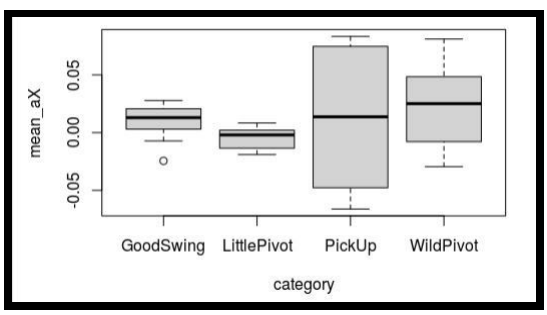


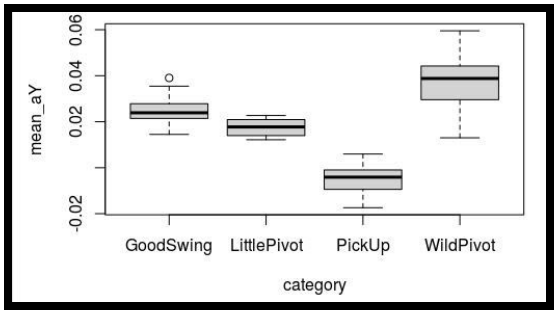


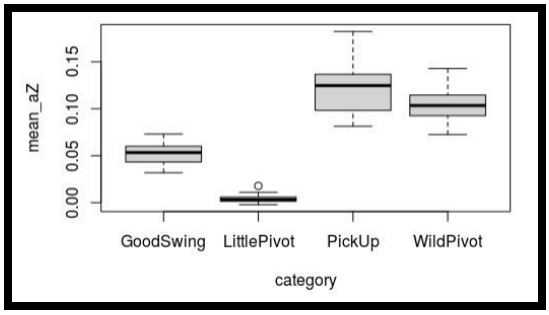


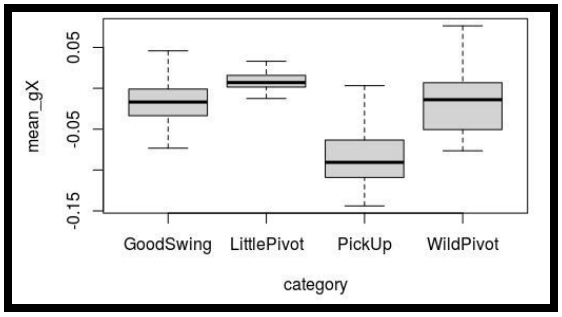


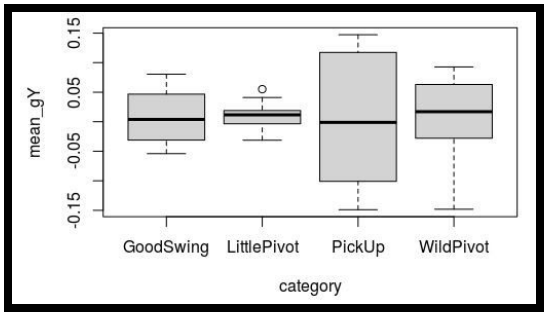
Below are the boxplots generated for the fourth dataset utilizing device motion, and the features I decided to use from these results are the accelerometer's Y and Z axes, and the gyroscope's X axis.

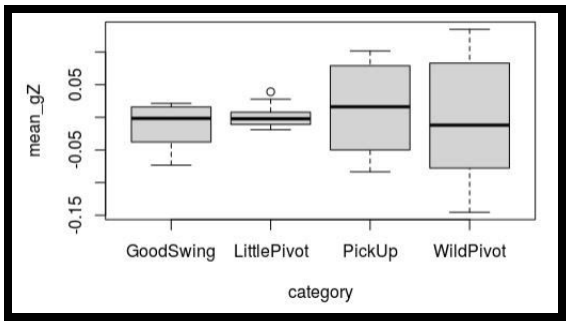








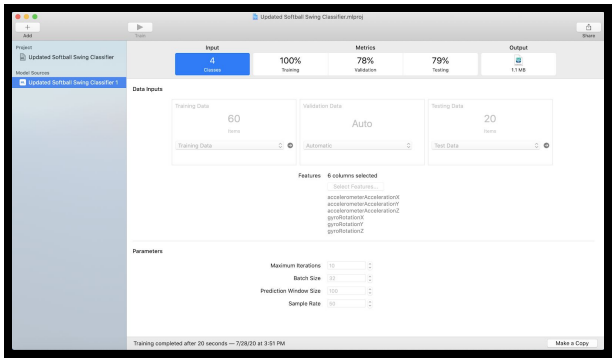


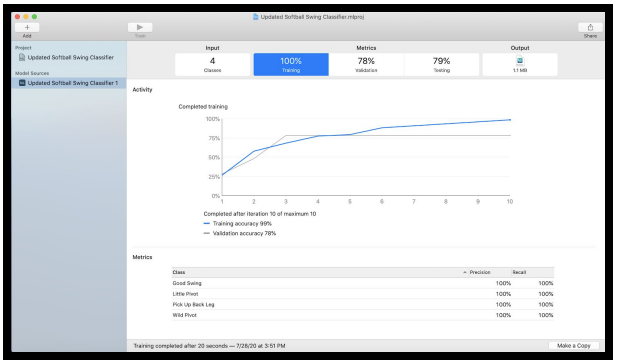


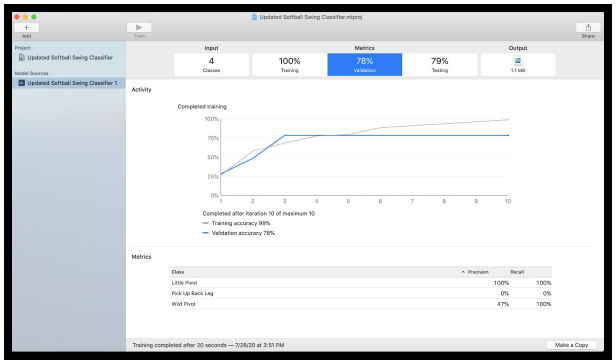
One caveat to this data analysis is that, although I picked out features to utilize in generating the models, I could rely on CreateML's neural network to process all six features, if the model could be significantly improved in doing so; this improvement in using all six features was present, so I did turn to models utilizing all axes of the accelerometer and gyroscope for the final products.

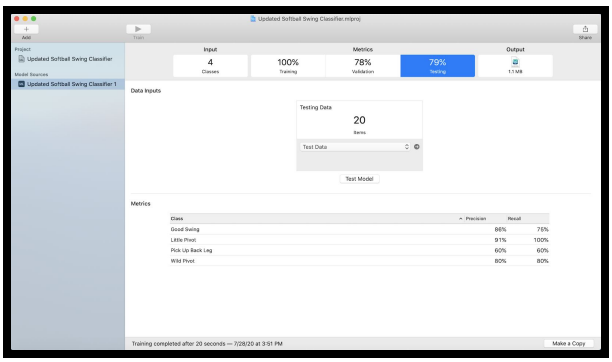
-- Model Creation

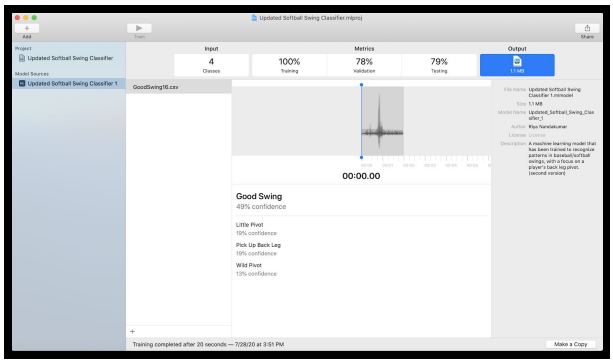
As previously stated, the CoreML model for classifying between swings was to be generated in Apple's CreateML, utilizing its newer "Activity Classifier" template. I properly organized my datasets into training and testing, with folders within those separating the different swing types. I then inputted the training and testing folders into CreateML, also selecting the desired features for the model. Screenshots of the CreateML interface can be seen below, with the results of the second (main) dataset's model.



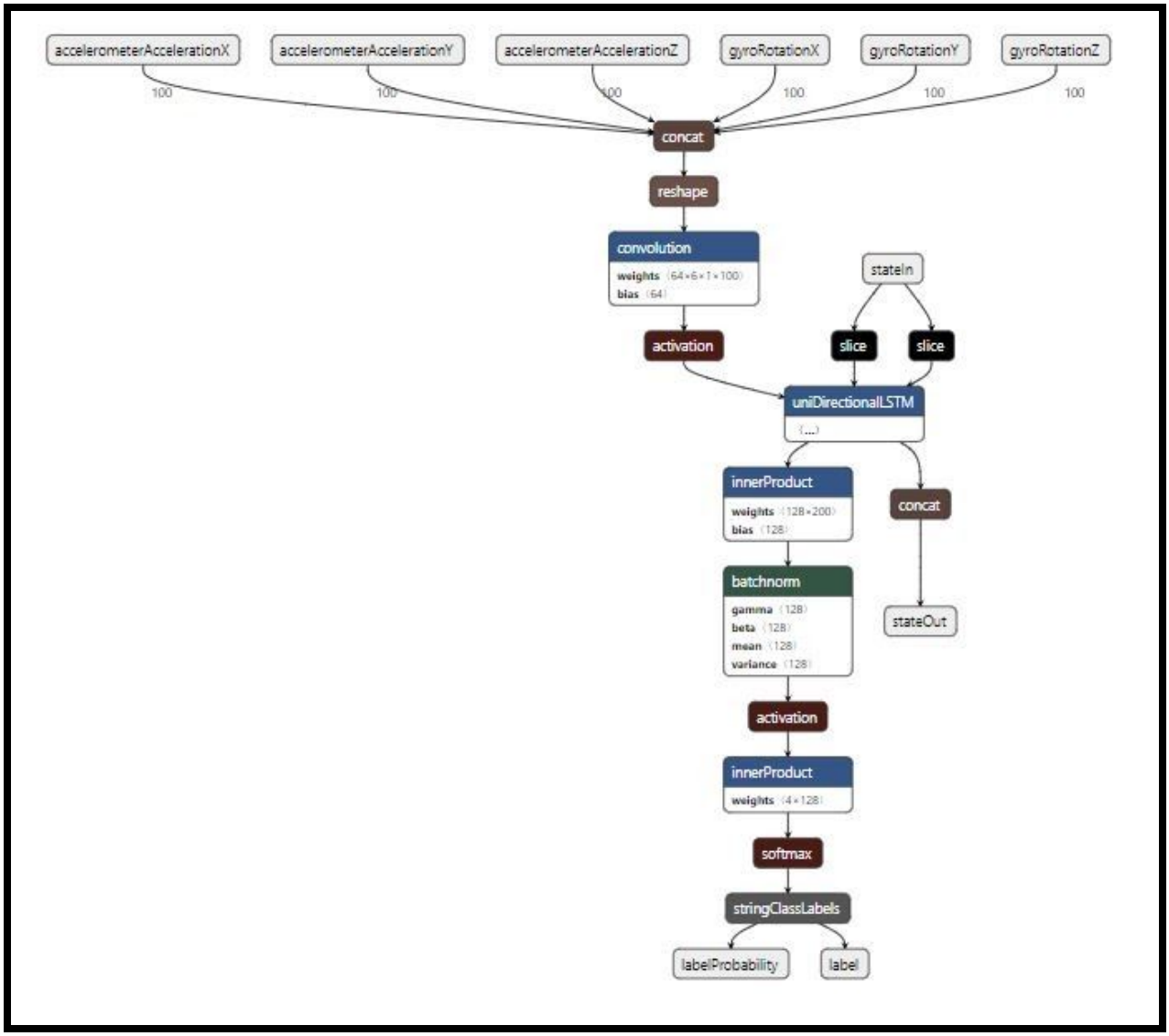








This generated model, utilizing all six features, has a testing accuracy of 79%, validation accuracy of 78%, and a training accuracy of 100%. Further understanding of this model comes in the form of Netron, a software that breaks down the components of otherwise black-box neural networks. The flow chart for the main model generated from this project can be seen below.

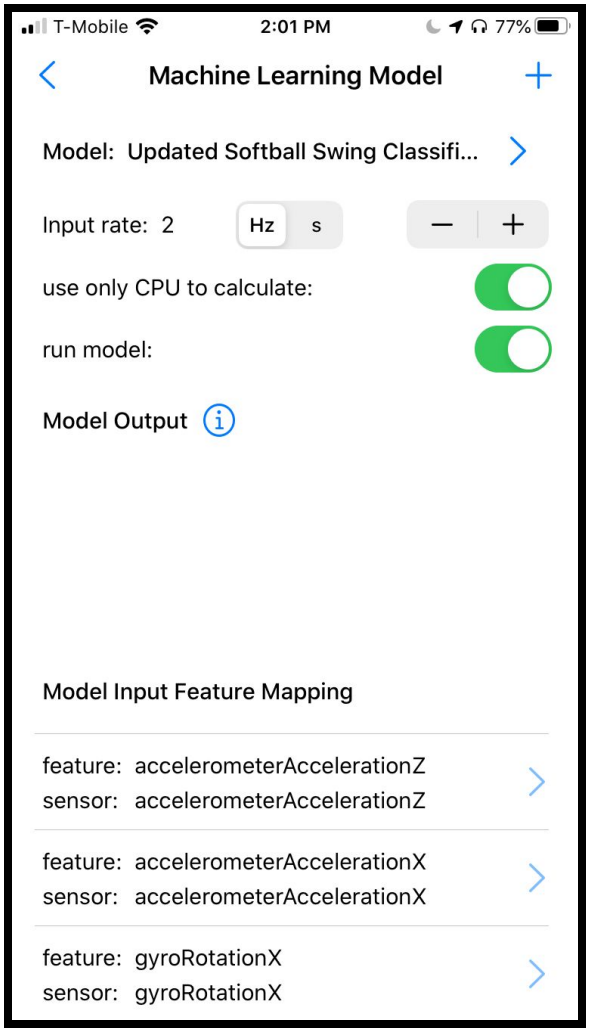


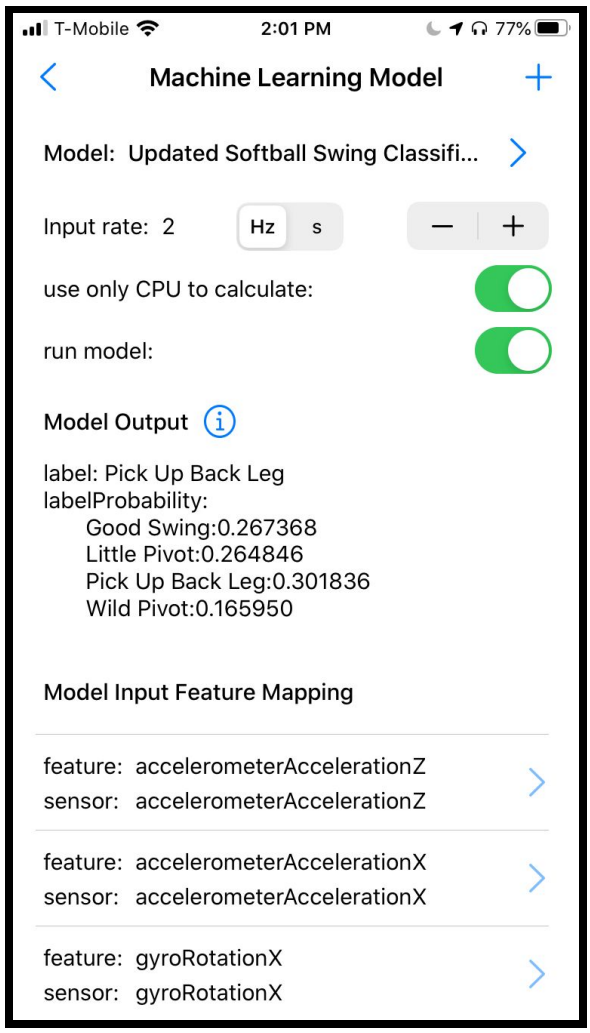
As observed in the network diagram, the model first concatenates the six features into a matrix that it can manipulate in further analysis. Next, a series of steps involving convolution and output manipulation produce a string of the label and the probability of such label. Further work on this diagram will be touched upon in the "Moving Forward" section.

-- Deployment

Deployment of the generated CoreML model occurred back to the SensorLog application, as it has a feature to process CoreML models and generate their outputs. This was slightly too good to be true, as the system for outputting classifications and recording these classifications

into a malleable .csv file was extremely finicky, and tough to get to properly process a stream of data.





.

The screenshots above show the basic interface for operating a CoreML model in SensorLog, with it first requiring you to connect the features of the model to sensors in the app; this was simple since I created the model based on the same exact features as recorded in this app. The screenshot to the right portrays how the app shows the output of the model, with it predicting a swing where the back leg was picked up off the ground, and showing the individual probabilities of each swing type; as noticeable in the output above, the portrayed prediction has quite a low probability, with every swing's probability being somewhat similar. This form of deployment for the model is far from optimal, as it can be extremely difficult to work with; the main goal moving forward and generally with deployment is to create an iOS app to properly house the

model, record classifications, and easily distribute this data to players and coaches. However, due to my inexperience with Swift and app development, this component to the project has not come to life within the timespan given. This will be touched upon again in the "Moving Forward" section.

4: Challenges

This project as a whole proved challenging due to various reasons, including my age/experience (or lack thereof), technical difficulties, and lack of needed resources.

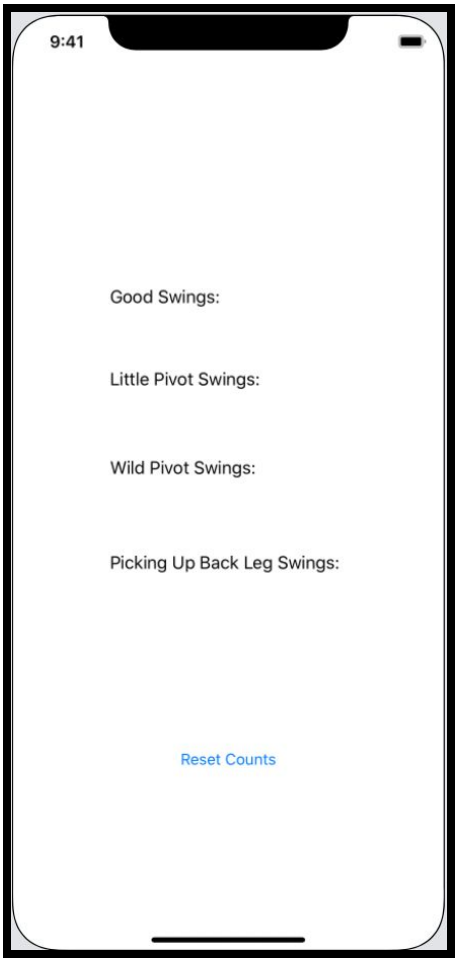
-- Timeline

I began the project with the intentions of utilizing the SensorTile and a Python notebook for recording the data, creating a model, and doing real-time classification. However, due to compatibility issues between my laptop and the SensorTile (due to applications I had downloaded to my laptop, most likely), I decided to switch gears towards an easier form of data collection and analysis. In my research after the SensorTile debacle, I found the SensorLog iOS app and shifted towards a CoreML model in order to save time and easily deploy the model to my phone. Utilizing CoreML and CreateML was easier said than done, as I have a Windows laptop, and no one in my family had a newer Macbook that I could borrow. I spent a weekend sorting through numerous ways of bringing a MacOS environment to my laptop, including building a virtual environment, downloading an old edition of iTunes, and trialing a cloud-based MacOS system. No workaround had the ability to complete my needed tasks, so my family as a whole decided to upgrade my sister's old Macbook for a Macbook capable of creating my needed model. This Macbook, as expected, worked great for my purposes; however, the route to this last option proved to be practically landmine-filled. After I had a path properly laid out for creating the model, I moved onto data collection, where I recorded two datasets quickly within the first two days of this process; the first dataset worked fine, yet I noticed slight issues with classifying good swings, so I decided to re-record these twenty swings into the second dataset. Although this dataset and the model generated from this dataset would prove to be the best created in the time given, I continued to experiment with different components to add or remove from the models and data collection. In my third dataset, I tried to add a "No Activity" class, as

was recommended in the 2019 WWDC video demonstrating the CreateML Activity Classifier template; I utilized all six features, yet the accuracy of this model simply could not keep up with the second model. Next, I began my attempts at creating an iOS app, and I decided to record a completely new dataset using the device motion sensor in SensorLog in order to transition to an iPhone application easier. The two models generated from this dataset (one with three features, one with all six) were extremely far from the accuracy I had aimed for, as the testing accuracy of both models fell around 50%; I continued utilizing these models for the purposes of app development, yet I knew I could not actually demo these models. Additionally, after recording my fourth dataset, I completed the data analysis in R to pick out proper features; this comes slightly out of order compared to a normal workflow for creating a machine learning model, but I added this as another form of experimentation, since the neural network training would operate the same no matter the number of features. By the time I had scraped my fourth model ideas, it was time to wrap up the project and begin working on deliverables.

5: Moving Forward -- Current Improvements

This project in its current state has great room for improvement. As stated previously, my main goal for this project moving forward is to shift the generated CoreML model into its own iOS app, where classification and tallying of different types of swings would be much smoother.



The screenshot above is what the basic interface of the app would look like, with tallies of each swing type and a button to reset the counts; components of the interface I would like to add are a screen showing the latest swing classified, and a manner of downloading the tallies along with the accelerometer and gyroscope data. Additional desires I have with this project are to broaden the data, as although in its current form variation in the data comes from the right-handed and left-handed split in swings, all swings utilized in datasets are my own. I did have a friend of mine record several swings early on, but the process of recording swings was not completely polished back then, so the data gathered was essentially unusable. Goals for broadening the data would be to recruit others to record swings, but to show the proper manner of data collection, in order to create the cleanest dataset possible. I would also like to try different types of models, potentially using a Python notebook and then converting the generated model to CoreML; by the time I was made aware of packages to convert Python notebooks into CoreML, I had already generated my

own CoreML model using CreateML. However, I do generally seek to switch this project over to the SensorTile and Python in order to get a deeper understanding of the actual processes that go into the model and data collection, so trials of different model types would be interesting to do in order to find the best fit. One small caveat before I would work on this switch is that I would need to fix the current compatibility issues between my laptop and the SensorTile.

-- Related Ideas

One idea I have which is similar to this project, yet slightly different, is to create a video classifier for classifying swings. This would be easier for coaches to utilize, than straight accelerometer and gyroscope data, as recordings of swings would be simple to look back on and observe mistakes in technique. A video classifier could also encompass more components of a swing, rather than just the back leg's rotation, so the model can be more generalizable. This is an idea I find very interesting and will work at in the future; however, for the time being, I have stuck to the back leg pivot classifier.

6: Repository and Resources

Repo:: https://github.com/riyanandakumar/Softball-Swing-Master

Images::

"Squash the Bug":
https://www.google.com/search?q=bryce+harper+squash+the+bug&rlz=1C1CHBF_enUS753US753&sxsrf=ALeKk00lw91MlQfezO8RLXGQPE1reYbzTg:1596854561679&tbm=isch&source=iu&ictx=1&fir=krqzO7G5jmQfXM%252C4WzuB0H8NZB6BM%252C_&vet=1&usg=AI4_-kQ5cko1lClNa0k-ry_DD5nzoaBzVw&sa=X&ved=2ahUKEwienqDnyorrAhVLj3IEHS3xD3wQ9QEwAnoECAIQBw&biw=1366&bih=657#imgrc=krqzO7G5jmQfXM

SensorLog:
https://www.google.com/search?q=sensorlog&rlz=1C1CHBF_enUS753US753&sxsrf=ALeKk03BDRuE15JaS7NPbuaZJ7MlE1W6MA:1596854566202&source=lnms&tbm=isch&sa=X&ved=2ahUKEwifp7TpyorrAhWVknIEHU69BWcQ_AUoAHoECA0QCA&biw=1366&bih=657#imgrc=ChZ5zfx_zktA7M

Resources::

CreateML Activity Classifier: https://developer.apple.com/videos/play/wwdc2019/426/

Accelerometer in XCode: https://www.youtube.com/watch?v=XDuchXYiWuE&t=132s

Gyroscope in XCode: https://www.youtube.com/watch?v=nsChcl2fvps&t=217s