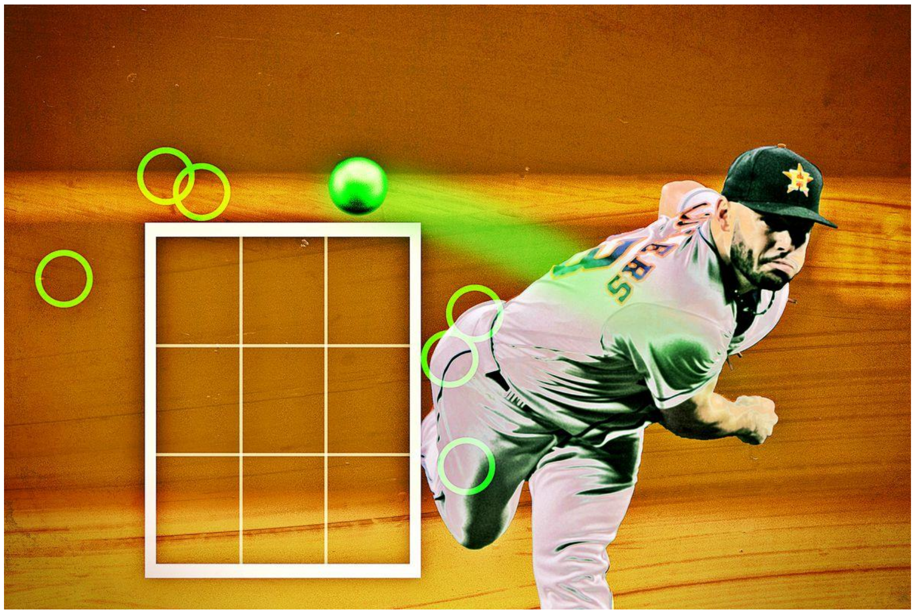
Understanding the Strike Zone:
A Deep Learning Approach to Pitch Prediction



INFO-518 Deep Learning Final Project

By Connor Bryson, Kael Ecord, Chin-yu Wu

Introduction and Problem Description:

Baseball is a sport of precision and anticipation. Every game is a battle between the pitcher at the mound and the batter at the plate. The goal of a pitcher in baseball is to throw the ball into a “strike zone” in a way that does not allow the batter at the plate to hit the ball. Major League Baseball (MLB) defines the “strike zone” as “the area over home plate from the midpoint between a batter's shoulders and the top of the uniform pants -- when the batter is in his stance and prepared to swing at a pitched ball -- and a point just below the kneecap.” 1 The strike zone is rectangular shaped with 9 zones within the rectangle and 4 zones outside the rectangle resulting in 14 different parts depending on where the ball was thrown. For a pitch to be considered strike either:

1. Part of the ball must cross over part of home plate while in the strike zone. (Called Strike)
2. The ball made contact with the batter's bat.
3. The batter started swinging the bat, decided to stop, but the bat broke the plane at the bottom of the plate according to the 1st or 3rd base umpire. (Check Swing)

Every game there is an umpire behind home plate opposite the pitcher that determines whether a given pitch is determined to be a strike. With batters changing every at bat, the umpire must change the strike zone to match the height of the batter at the plate. However, this constant shift in the strike zone has resulted in inconsistent pitch calls which can favor the batter or the pitcher depending on the umpire. Not only does the strike zone shift during the game, but the strike zone also shifts depending on the umpire behind the plate.

With Pitchers and Batters frustrated by inconsistent pitch calls, Major League Baseball decided to introduce a new pitch system called the “Automatic Balls and Strikes System” (ABS). This system created a new “electronic strike zone” that can be overlayed on live camera footage for pitch confirmation. “In 2019, the independent Atlantic League used the electronic strike zone in an all-star game, and that same year, the Arizona Fall League was played with the ABS. In 2021, the ABS was deployed in some Class A parks." (Olney)

In 2023, it was announced that the electronic strike zone will be used in all 30 Triple AAA (One level below the Major Leagues) parks in 2 ways. “Half of the Class AAA games will be played with all of the calls determined by an electronic strike zone, and the other half will be played with an ABS challenge system similar to that used in professional tennis. Each team will be allowed three challenges per game, with teams retaining challenges in cases when they are proved correct. MLB's intention is to use the data and feedback from both systems, over the full slate of games, to inform future choices.” (Olney)

Goal of this Project:

Before this new ABS system, it was impossible to have a standard strike zone that could be compared to each other. However, that is no longer the case. This electronic system now allows teams to challenge the pitch in a way that can correct umpire mistakes and make the game fair for batters and pitchers alike.

The goal of this project is to utilize Deep Learning Methods in a way that helps predict pitch call (Ball or Strike), pitch type, and strike zone area based on the metrics of the pitch thrown. By understanding the pitch metrics and whether that pitch is considered a strike or a ball, pitchers will be able to be more precise in their pitches, so they get the strike call that they want. Thus, our goal is to create a basic framework for future Deep Learning models related to this issue.

Dataset:

The data that we will be using comes from the MLB's advanced tracking technology that collects and analyses a massive amount of baseball data. From 2015-2019, Statcast consisted of a combination of cameras and radar systems to collect the data. However, in 2020 this changed with the introduction of the Hawk-Eye. (Same camera used for instant replay at all major tennis tournaments) This new camera increased the tracking ability, and now every team in the league has 12 Hawk-Eye cameras set up strategically around the stadium. For our study's purposes, there are 5 key cameras, which have higher frames per second, focusing on pitch tracking.

The specific data that we will be using has been downloaded using pybaseball. This is a python package built for baseball data analysis as it allows for easy scraping of Statcast data and a few other key baseball stat-keeping sources. Using this package, we set a date range for all the pitches we wanted to include in our dataset. We selected March 30, 2023, to October 1, 2023. This date range encapsulates the entire 2023 MLB regular season. After downloading this data, we ended up with 92 columns of pitch-by-pitch data for 717,504 individual pitches. Many of the columns were unrelated to the goal of this project (described above) so they were deleted from the dataset, but a comprehensive list of the columns in original dataset can be found here: Statcast Search CSV Documentation | baseballsavant.com (mlb.com).

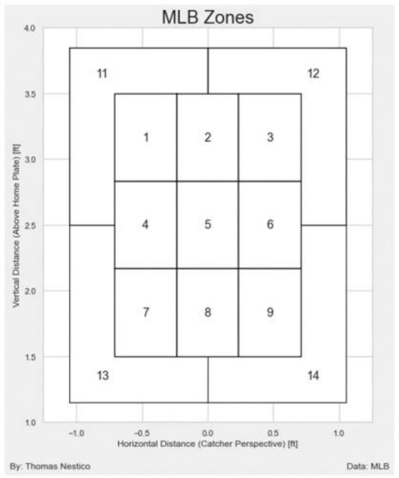
For our research, we will have 4 separate datasets.

- Dataset 1: Unbalanced – rows with type = 'X' left in
  - This dataset will be used by Chin-yu to predict "pitch_type" and by Kael to predict "zone". It will contain an unbalanced number of rows for each pitch type, for example there can be 100 fastballs and 90 curveballs (not real numbers). Rows where type = 'X' will be left in the dataset.
- Dataset 2: Unbalanced – rows with type = 'X' removed
  - This dataset will be also used by Connor to predict the variable "type". Before anything else is done all rows from the entire dataset where type = 'X' will be removed. The justification for this is that the goal is to predict whether the pitch was called a ball or strike by the umpire. The rows where type = 'X' contain pitches where the result of the pitch is the ball in play, i.e. the batter hit the ball, which results in the umpire not having to make a call. Finally, this dataset will contain an unbalanced number of rows for each pitch type, for example there can be 100 fastballs and 90 curveballs (not real numbers).
- Dataset 3: Balanced– rows with type = 'X' left in
  - This dataset will be also used by Chin-yu to predict "pitch_type" and by Kael to predict "zone". It will contain a balanced number of rows for each pitch type. This will be accomplished by finding the pitch type with the fewest number of pitches in the dataset. All other pitch types will be randomly sampled to end up with a dataset that has the same number of pitches for each pitch type. For example, if the knuckleball was only thrown 75 times in 2023, this dataset will contain 75 random rows for every other pitch type. Rows where type = 'X' will be left in the dataset.
- Dataset 4: Balanced– rows with type = 'X' removed
  - This dataset will be also used by Connor to predict the variable "type". Like dataset 2 all rows where type = 'X' will be removed. Finally, like dataset 3 this dataset will contain a balanced number of rows for each pitch type.

The following variables will be used in our analysis:

Response Variables:

- pitch_type - Two letter abbreviation classifying the type of pitch derived from statcast
- Zone - The zone location of the ball when it crosses the plate from the catcher's perspective. 1-9 represents the strike zone 11-14 represent balls.



- Type - Shorthand of pitch result. B = ball, S = strike, and X = in play.

Input Variables:

- release_speed - Pitch velocity reported out-of-hand
- release_pos_x - Horizontal release position of the ball measured in feet from the catcher's perspective
- release_pos_z - Vertical release position of the ball measured in feet from the catcher's perspective
- release_spin_rate - spin rate of the pitch measured at the release point
- Stand - Side of the plate batter is standing
- p_throws - Hand pitcher throws with
- pfx_x - Horizontal movement in feet from the catcher's perspective.
- pfx_z - Vertical movement in feet from the catcher's perspective.
- plate_x - Horizontal position of the ball when it crosses home plate from the catcher's perspective.
    - NOTE: Will only be used when predicting pitch type
- plate_z - Vertical position of the ball when it crosses home plate from the catcher's perspective.
    - NOTE: Will only be used when predicting pitch type
- vx0 - The velocity of the pitch, in feet per second, in x-dimension, determined at y=50 feet (50 feet off the pitcher's mound towards the batter)
- vy0 - The velocity of the pitch, in feet per second, in y-dimension, determined at y=50 feet (50 feet off the pitcher's mound towards the batter)
- vz0 - The velocity of the pitch, in feet per second, in z-dimension, determined at y=50 feet (50 feet off the pitcher's mound towards the batter)
- ax - The acceleration of the pitch, in feet per second, in x-dimension, determined at y=50 feet (50 feet off the pitcher's mound towards the batter)

- ay - The acceleration of the pitch, in feet per second, in y-dimension, determined at y=50 feet (50 feet off the pitcher's mound towards the batter)
- az - The acceleration of the pitch, in feet per second, in z-dimension, determined at y=50 feet (50 feet off the pitcher's mound towards the batter)
- sz_top - Top of the batter's strike zone set by the operator when the ball is halfway to the plate
- sz_bot - Bottom of the batter's strike zone set by the operator when the ball is halfway to the plate
- effective_speed - Derived speed based on the extension of the pitcher's release
- release_extension - Release extension of the pitch in feet
- release_pos_y - Release position of pitch measured in feet from the catcher's perspective
- spin_axis - The spin axis in the 2D X-Z plane in degrees, from 0-360, such that 180 corresponds to a pure backspin fastball and 0 degrees represents a pure topspin (12-6) curveball

Methodology:

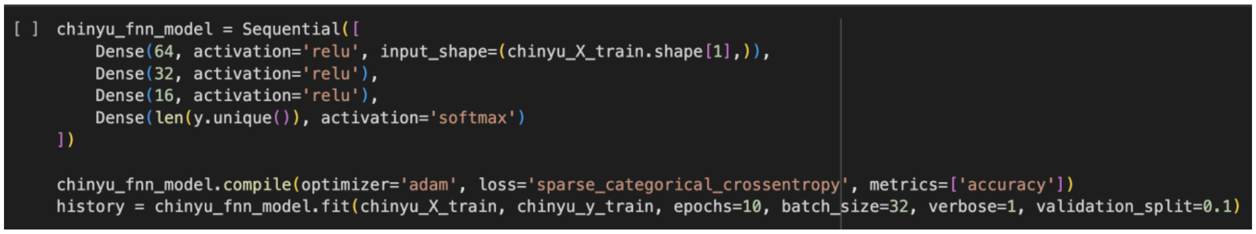
For this project we decided to create models to predict 3 different response variables:

1. Whether the Pitch was a Ball or a Strike (Pitch Call)
2. Pitch Type (Cutter, Slider, Fastball, etc)
3. Where in the Strike Zone (1, 2, 3,....) the pitch landed

For the pitch type and pitch call response variables model, we built a four-layer FFNN and 1D CNN model. For the strike zone prediction, we built 1D and 2D CNN models. Even though we have a very large dataset (700,000+ for Dataset 1 and 500,000+ for Dataset 2), there is still some minority class in the response variable. To account for these unbalanced classes, we used SMOTE sampling to compensate those minority class and then compare the test accuracy and test loss to the orthodox train-test split data. The SMOTE sampling dataset is the Balanced dataset that is mentioned previously.

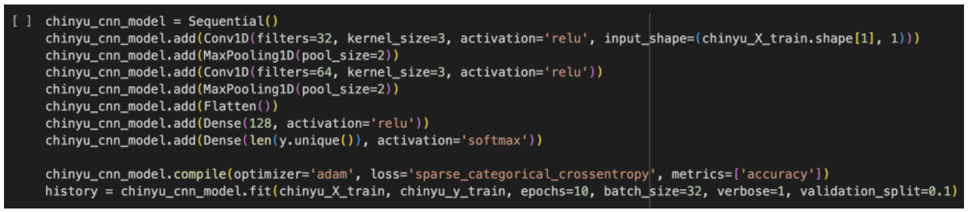
Comparing the test accuracy of using unbalanced and balanced dataset some of the test accuracy did improve some of it doesn't. But two out of the three prediction models that have the highest accuracy are using the balanced dataset. For the pitch call prediction, the highest accuracy is FFNN using the balanced dataset, highest accuracy for pitch type prediction is CNN using unbalanced data and the highest accuracy for zone prediction is CNN using balanced data. An example of what structure was used for the FFNN and the CNN can be seen below.

Feed Forward Neural Network Architecture:



```
[ ]  chinyu_fnn_model = Sequential([
         Dense(64, activation='relu', input_shape=(chinyu_X_train.shape[1],)),
         Dense(32, activation='relu'),
         Dense(16, activation='relu'),
         Dense(len(y.unique()), activation='softmax')
     ])

     chinyu_fnn_model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
     history = chinyu_fnn_model.fit(chinyu_X_train, chinyu_y_train, epochs=10, batch_size=32, verbose=1, validation_split=0.1)
```

CNN Architecture (1-Dimensional Input):



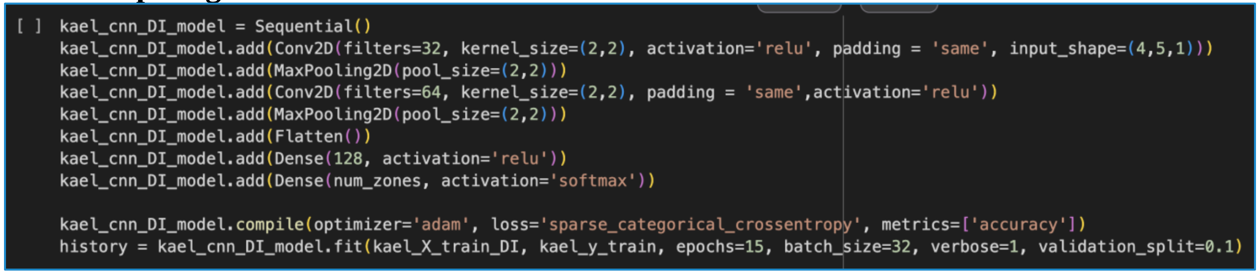
```
[ ]  chinyu_cnn_model = Sequential()
     chinyu_cnn_model.add(Conv1D(filters=32, kernel_size=3, activation='relu', input_shape=(chinyu_X_train.shape[1], 1)))
     chinyu_cnn_model.add(MaxPooling1D(pool_size=2))
     chinyu_cnn_model.add(Conv1D(filters=64, kernel_size=3, activation='relu'))
     chinyu_cnn_model.add(MaxPooling1D(pool_size=2))
     chinyu_cnn_model.add(Flatten())
     chinyu_cnn_model.add(Dense(128, activation='relu'))
     chinyu_cnn_model.add(Dense(len(y.unique()), activation='softmax'))

     chinyu_cnn_model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
     history = chinyu_cnn_model.fit(chinyu_X_train, chinyu_y_train, epochs=10, batch_size=32, verbose=1, validation_split=0.1)
```

The second CNN method that was used is called the DeepInsight method. “The concept [...] is to first transform a non-image sample to an image form and then supply it to the CNN architecture for the [...] classification purpose” (Sharma, et. al, 2019). The basic idea is to restructure your 1-dimensional input into a 2-dimensional matrix to allow the CNN architecture to extract features like it would from an image. This is done by configuring the matrix so that related variables are close together and unrelated variables as far apart as possible. This method was implemented when attempting to predict “zone”. Variables such as release_spin_rate and spin_axis were placed together while variables sz_top and sz_bot were placed together on the opposite side of the matrix. The authors of the paper that introduced this method showed significant improvement in classification accuracy over methods like Decision Trees and Random Forest. After implementing this method, the following CNN model structure was used.

CNN-DeepInsight Input Matrix Construction:

Release_pos_x	Release_pos_y	throw	Pfx_x	Sz_top
Release_pos_z	Release_speed	Release_extension	Pfx_z	Sz_bot
Release_spin_rate	Effective_speed	vz	az	Stand
Spin_axis	vx	vy	ax	ay

CNN-DeepInsight Architecture:



```
[ ]  kael_cnn_DI_model = Sequential()
     kael_cnn_DI_model.add(Conv2D(filters=32, kernel_size=(2,2), activation='relu', padding = 'same', input_shape=(4,5,1)))
     kael_cnn_DI_model.add(MaxPooling2D(pool_size=(2,2)))
     kael_cnn_DI_model.add(Conv2D(filters=64, kernel_size=(2,2), padding = 'same',activation='relu'))
     kael_cnn_DI_model.add(MaxPooling2D(pool_size=(2,2)))
     kael_cnn_DI_model.add(Flatten())
     kael_cnn_DI_model.add(Dense(128, activation='relu'))
     kael_cnn_DI_model.add(Dense(num_zones, activation='softmax'))

     kael_cnn_DI_model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
     history = kael_cnn_DI_model.fit(kael_X_train_DI, kael_y_train, epochs=15, batch_size=32, verbose=1, validation_split=0.1)
```

Results:

Model	Response Variable	Data	Test Accuracy	Test Loss
Feed Forward	Pitch Call	Unbalanced	0.8037	0.4186
Feed Forward	**Pitch Call**	**Balanced**	**0.8214**	**0.3899**
CNN	Pitch Call	Unbalanced	0.7668	0.4987
CNN	Pitch Call	Balanced	0.6736	0.6155
Feed Forward	Pitch Type	Unbalanced	0.8243	0.4534
Feed Forward	Pitch Type	Balanced	0.7888	0.5298
CNN	**Pitch Type**	**Unbalanced**	**0.8337**	**0.4239**
CNN	Pitch Type	Balanced	0.8127	0.4772
CNN	Zone	Unbalanced	0.8412	0.3767

CNN	**Zone**	**Balanced**	**0.8476**	**0.3868**
CNN- Deep Insight	Zone	Unbalanced	0.6436	0.8622
CNN- Deep Insight	Zone	Balanced	0.3956	1.6646

Future Work:

With the ABS system currently testing in the minor leagues, it is a matter of time until this technology is introduced to the major leagues. Once the automatic strike is introduced, the MLB will be able to have a standard strike zone no matter the game, batter, or umpire. Since every strike zone will be consistent, future questions like "How does the height of a batter affect the size of the strike zone?" and "What is the best way to use pitch challenges given that too many lost challenges results in losing the ability to challenge?" will become relevant to teams around the league as will the data that supports those analyses.

References:

Olney, Buster. "Sources: All AAA Parks to Use Electronic Strike Zone in '23." ESPN, ESPN Internet Ventures, www.espn.com/mlb/story/_/id/35434317/sources-all-aaa-parks-use-electronic-strike-zone-23. Accessed 13 Apr. 2024.

Sharma, A., Vans, E., Shigemizu, D. *et al.* DeepInsight: A methodology to transform a non-image data to an image for convolution neural network architecture. *Sci Rep* **9**, 11399 (2019). https://doi.org/10.1038/s41598-019-47765-6

"Strike Zone: Glossary." mlb.com, www.mlb.com/glossary/rules/strike-zone. Accessed 13 Apr. 2024.

Statcast Search CSV Documentation | baseballsavant.com (mlb.com)

Statcast | Glossary | MLB.com

Appendix:

Connor Bryson- Wrote the sections "Introduction and Problem Description", "Goal of this Project", "Future Work" and contributed to the sections "Methodology" and "Results". Created Multi-Label Classification Neural Network and CNN Model to predict pitch call (Ball or Strike)

Kael Ecord - Wrote the "Dataset" section and contributed to "Methods" and "Results" sections and worked on data cleaning. Created CNN model and CNN model using Deep Insight method to predict strike zone area (1,2, etc.).

Chin-yu Wu- Wrote the sections "Methodology" and "Result" and worked on data cleaning. Created Multi-Label Classification Neural Network and CNN Model to predict pitch call (Fastball, Slider, etc.)