Ranking Super Smash Characters

Trayson Keli'i
Department of Computer Science
Brigham Young University

Abstract

This paper seeks to explore the ability of machine learning models to predict the efficacy of different *Super Smash Bros.* characters across the five different generations of games in the franchise. The newest generation of this game, Smash Ultimate, is new enough that players have not settled upon consistent rankings for the different characters; as such, our goal is to predict those rankings by evaluating the attributes which were found to be significant features in previous generations. Data for all five games was manually collected from various websites and then consolidated into the most important attributes. The different models which were evaluated where decision tree, k-nearest neighbors, k-means clustering, and a neural network trained with backpropagation. Of these models, the neural network was found to be the most effective, achieving an accuracy of 80.1%.

1 Introduction

1.1 Motivation

Historically, tier lists have been created by competitive players practicing with different characters and choosing their favorites. By applying machine learning to this problem, we hope to be able to take that historical data and apply it to the newest game which has not yet had solidified tier lists. The older games have had years of competitive play to stabilize the popular strategies(aka meta). We hope to leverage this stability to help speed up the development of the newest game's meta. This is important for competitive play because it helps professional players to be able to make wise choices in which characters they spend time practicing.

1.2 Problem Description

The Super Smash Brothers series maintains a strong presence in the esports community and is praised for its large, diverse roster of characters. Professional players and amateurs alike are constantly creating lists of character rankings to determine which character would perform the best in a tournament setting. A characters rank represents the relative advantage of using one character over any other in a fight of two equally matched players, and a character tier is a grouping of characters who are ranked similarly, usually listed in letter grades. We aim to run different machine learning algorithms on the rankings and tiers of the 133 characters in previous Super Smash Brothers games, then use the best learned model on the 72 characters in the sequel, Super Smash Brothers Ultimate to predict what their rankings should be. In the future, we will compare our machine-learned rankings with the official rankings when the game stops receiving updates.

2 Methods

In this section we first explain the methods used in collecting and preparing our data. Following that is an explanation of the models and experiments performed in this work.

2.1 Data Sources

The bulk of our data was retrieved from SmashWiki[2]. This is a community driven database for all of the Super Smash Brothers games. Another source of information that we were hoping to consider comes from SmashBoard user Kurogane-Hammer, the curator of a fan-made database of more in-depth information for the characters of certain entries in the series [4]. However, in the end this data was not included because it proved to be difficult to unify across characters, and the information was limited to only 3 of the 5 games, with large chunks of data missing for the newest game.

We wanted to incorporate some attributes that were not purely measurable in-game but observable through gameplay, such as character archetype. For this we had to rely on our own knowledge of the games and experiments conducted by our team members.

Finally, we collected individual character rankings and tier lists from several different locations. The main source was again SmashWiki, but other sources included Smashboards [3], RankedBoost [5], HTC [6], and tier lists from professional players collected by IGN [7].

2.2 The Dataset

The training dataset has 133 training instances, comprised of the rosters from games 1-4, with 43 attributes per instance, and one class label–the characters tier. The test set, comprised of the roster of Super Smash Brothers Ultimate, contains 72 instances, for a combined total of 205 instances across all games. There is a combination of continuous and nominal attributes.

Because of the sheer number of attributes for each instance, several versions of the dataset with reduced attributes were compiled to be used throughout our experiments. The full attribute list and attribute lists for each of the reduced datasets are listed in the appendix, and the complete dataset can be found at [1].

The class labels for our dataset were created by standardizing individual tier and ranking lists from the online sources listed in **Section 2.1**, averaging together the standardized scores for each instance, re-standardizing the averaged values to retain a standard deviation of 1. Finally, those standardized scores were discretized into 6 even width bins to represent a universal standard tier list: S (Superior), A, B, C, D, and F.

The attributes collected for each instance are primarily comprised of the raw damage output and type information for the moveset of each character. Each of the characters has roughly the same set of moves. This allows us to compare categories of attacks across the different characters. However, not every move is exactly alike across characters, as some characters perform multiple successive attacks for the same player input, or have a chance of dealing higher damage in certain situations. To deal with this, We chose to sum multiple successive attacks and average together damage ranges to create an overall damage score for the attack. For instance, one attack may deal 4 damage on frame 3, 1 damage each on frames 4-6, and 2 damage on frame 7 for a *total* of 9 damage, and another attack may deal 14 damage if it the attack hits in a "sweet spot," or 4 damage otherwise, for an *average* damage of 9.

It is worth noting that, for most classification problems, having only 133 training instances with so many attributes for each instance would be problematic. However, the problem being considered in this work is unique in that the universal set of instances is described almost in full with the 205 characters across all games (including Ultimate), barring any updates to characters made post game launch. And, with how unique each character is, the more detail we can capture in each instance, the better. Thus there is a need for a somewhat larger set of attributes and possible attribute values.

2.3 KNN

The K Nearest Neighbors algorithm works by collecting a group of data points as a kernel and then comparing new data points against those points to find its closest neighbors. Each new point's nearest K neighbors are found and then each of the output classes for those neighbors is given a vote that is scaled based on their distance from the new point. The largest vote wins.

2.4 Decision Tree

The Decision Tree algorithm trains by splitting the initial data set into subsets by those categorical attributes which are most informative. These subsets become the children nodes of the root (initial set). While we can still gain insight by splitting on informative attributes, we take these subsets and split them further growing our tree larger and larger. We can predict the output class for some new instance by matching our new instance's attributes with the attributes which were split on and traversing the tree as such. Once we arrive at some leaf node, we output the most common class among the instances contained in this node.

2.5 K-Means

The K-Means algorithm is an unsupervised clustering algorithm that groups similar instances together into clusters. This algorithm uses Euclidean distance as its primary distance metric, and uses a binary distance for nominal and unknown data. The crux of this algorithm are the *k* centroids, which represent the center point of each cluster to which each instance is measured against. The K-Means algorithm assigns all instances to their nearest centroid, then recalculates the centroid for each cluster, repeating until there is no significant change in the clustering. In terms of our data, this algorithm should work well with all dataset variations, as K-Means generally does not learn how the different features interact with one another. Our goal for using K-Means is to label each resulting cluster with most common label of each member, then combine similarly labeled clusters to model the character rankings.

2.6 Neural Network

The neural network was the most promising model among the various models we planned on testing. As such, we invested time into performing an exhaustive topology search on several of the dataset versions. We performed a search over the 9 different network topologies outlined below:

1. input-32-6
2. input-64-6
3. input-128-6
4. input-128-128-6
5. input-32-64-32-6
6. input-16-32-64-32-16-6
7. input-32-64-128-64-32-6
8. input-64-128-256-128-64-6
9. input-64-128-256-256-128-6

Each model utilized a learning rate of 1e-4, and a momentum of 0.99. On top of this, each model was initialized using xavier-normal initialization, with ReLU activations between hidden layers. Searches were not performed over these hyper parameters to save time, and were selected based on prior experience with neural networks. With each topology, we ran 3 tenfold cross validation trials on 3 different normalized versions of the training set and averaged the reported accuracy. For each training fold, early stopping was performed using an 80/20 validation split, stopping based on the best observed validation MSE. Training stopped when no improvements were observed after 20 epochs.

3 Initial Results

3.1 KNN

When KNN was run on the Super Smash Bros. dataset the accuracy was underwhelming with a best-achieved accuracy of

25%. Different values of k were tested, and k=5 was found to be the best performing. However, although the accuracy was rather low, when the confusion matrix was evaluated it was discovered that the model was usually close to identifying the correct classification. Below is a chart showing the confusion matrix for the KNN model run on the dataset.

	S	A	B	C	D	F
S	0	3	3	2	1	0
A	5	4	2	0	4	0
B	4	1	4	2	4	1
C	3	2	6	1	2	2
D	0	1	1	2	5	2
F	1	0	0	0	3	3

Figure 1: Confusion Matrix

As can be seen in figure 1, the highest numbers are along the diagonal axis. This suggests that although the prediction was wrong, it was close to the correct classification. In further investigation it would be interesting to switch out KNN for RBF and compare the Mean Squared Error with other regression models.

3.2 Decision Tree

In order to run the Decision Tree algorithm on our Smash Brothers data set, it was necessary to extend the Decision Tree algorithm to be able to handle continuous data. To do this, we normalized the continuous data and then binned them into 5 ranges.

Bin	Range
1	0 <= val <= 0.2
2	0.2 <val <= 0.4
3	0.4 <val <= 0.6
4	0.6 <val <= 0.8
5	0.8 <val <= 1.0

Running with these bins, however, gave an underwhelming accuracy of 18.91%. We considered it probable that this data set requires more nuanced connections between different attributes.

In looking for ways to improve accuracy for the Decision Tree algorithm, recall that while KNN didn't have the best accuracy, it did come close to grouping the characters in the correct tier. Furthermore, among top players there will often be debate as to whether or not one character is actually an "S" or "A" tier character. Given this, we tried to create a modified version of the data set where we narrowed the 6 tiers into 3, grouping them with their closest tier. Running on this altered data set, we were able to achieve an accuracy of 35.14% with no pruning and 36.50% with pruning. While this is a significant increase in % accuracy, it still leaves much to be desired. In order to achieve better results, we then ran the data set on our neural network algorithm.

3.3 Neural Network

The results of our neural network topology search are outlined below. The early results, starting with the smallest topology, were extremely poor, sitting just below random choice accuracy at around 14%. However, continuing the search resulted in much better performance.

10-Fold Cross Validation Accuracy

model no.	micro	optimized	full
1	0.1564	0.1559	0.1319
2	0.1494	0.1469	0.1594
3	0.2163	0.2615	0.2095
4	0.4357	0.7425	0.7548
5	0.5875	0.5712	0.5335
6	0.5705	0.5298	0.5520
7	0.6908	0.7298	0.7094
8	0.7500	0.7734	0.7502
9	**0.7216**	**0.7556**	**0.7809**

Figure 2: Results from the model topology and dataset grid search.

The trends show that the best topologies proved to be ones that had more layers and hidden nodes, with topology 9 performing best overall when trained on the full 44 attribute dataset, though performance between the optimized and full dataset were comparable.

3.4 K-Means

K-Means seemed like the perfect model for character rankings, since similar characters tend to be ranked the same. Before using this model, we decided upon the *k* value, that is to say the number of clusters the model will produce. Most Super Smash Brothers tier lists contain six tiers, which act as clusters of characters that perform the same in tournament settings. As such, we decided to use *k = 6* as a baseline of our clustering, as this would emulate a novel tier list. We also tested *k = 10* and *k = 15*, but did not test any higher values because the model would begin to create clusters with only one character, which is not statistically significant for this problem.

While there are many performance metrics for K-Means, we used Silhouette scoring as our primary accuracy metric. Because Silhouette scoring measures the ratio of dissimilarity between an instance and members of the nearest cluster over the dissimilarity of the instance compared to members of its same cluster, a Silhouette score close to one is desired. The highest average Silhouette score we could achieve with any of our dataset variations was 0.10935, with *k = 15.* While testing higher *k* values would improve the Silhouette score until it reaches 1.0, where every cluster only has one instance, the resulting clusterings would be statistically insignificant given our problem. Because we aim to predict a novel instance into one of six discrete labels, having more tight, yet low capacity clusters would prove ineffective in mapping the resulting clusters back to these six discrete labels. As such, we conclude that K-Means is not the optimal learning model for our problem.

4 Final Results

As mentioned in **Section 2.6**, the best observed performance was achieved with a neural network model trained with back-propagation. Therefore, final results were measured using this model.

To achieve test set results, the input-64-128-256-128-6 neural network was trained on the full 43 attribute training split of the dataset (All character data from games 1-4), then applied to the test split (the latest game, *Smash Ultimate*). The final accuracy was averaged across 10 trials to get a clearer sense of performance before revealing individual character classifications made by the neural network. The results of this are shown below.

Test Accuracy

Run	Accuracy
1	0.7703
2	0.8244
3	0.8378
4	0.7838
5	0.8378
6	0.8243
7	0.8378
8	0.7703
9	0.8378
10	0.8378

Despite the complexity of this model and the relatively small dataset, the model obtained good performance on both initial cross-validation and the final test set, which proves the quality of generalization achieved by the model.

On the following page is a table comparing the predicted classifications with the crowd source tier, in which the model agreed with the still-developing tier list of *Smash Ultimate* about 80% of the time.

Predicted Character Tiers

Character	Tier	Predicted	Character	Tier	Predicted
Bayonetta	B	A	Bowser	C	C
Bowser Jr.	F	F	Capt. Falcon	C	C
Charizard	B	B	Chrom	S	D
Cloud	A	A	Corrin	C	C
Daisy	S	S	Dark Pit	C	C
Dark Samus	D	D	Diddy Kong	B	C
Donkey Kong	B	C	Dr. Mario	D	D
Duck Hunt	D	A	Falco	B	B
Fox	A	A	Ganondorf	D	D
Greninja	C	C	Ice Climbers	F	F
Ike	A	A	Incineroar	C	A
Inkling	S	S	Isabelle	B	B
Ivysaur	A	A	Jigglypuff	D	F
Ken	B	B	King Dedede	F	F
King K. Rool	B	B	Kirby	F	F
Link	B	F	Little Mac	F	D
Lucario	B	B	Lucas	C	C
Lucina	S	S	Luigi	D	D
Mario	C	C	Marth	S	S
Mega Man	D	D	Meta Knight	A	A
Mewtwo	S	A	Mr. Game & Watch	D	A
Ness	C	C	Olimar	A	A
Pac-Man	F	F	Palutena	B	B
Peach	S	S	Pichu	A	A
Pikachu	S	S	Piranha Plant	C	D
Pit	C	C	R.O.B.	B	B
Richter	A	A	Ridley	D	D
Robin	D	D	Rosalina & Luma	C	A
Roy	S	S	Ryu	C	C
Samus	D	D	Sheik	B	B
Shulk	A	A	Simon	A	A
Snake	B	B	Sonic	A	A
Squirtle	A	C	Toon Link	B	B
Villager	C	C	Wario	C	B
Wii Fit Trainer	F	D	Wolf	B	B
Yoshi	A	A	Younk Link	A	A
Zelda	C	B	Zero Suit Samus	B	B

Figure 3: Our best model's predicted tiers for the game compared with the crowd-sourced tier list. Note how on some characters where the ranks differ, they differ by only one letter grade, where others differing classifications vary greatly.

5 Conclusions

Having a neural network outperform all other algorithms confirmed something interesting about the nature of the characters in this series. The data required a model capable of learning extremely high-order features in order to generalize well. In other words, what makes a character good or bad is very complex.

This falls in line with the consensus of the series competitive community: Besides the high degree of variation among each characters moves and raw damage output, there is an interplay between a characters play-style and other minor attributes which is hard to quantify. Perhaps the most accurate observations are made as one plays the game; one describes how a character feels rather than singling out an attribute or set of attributes that identify an overall best character. As such, the neural network model is better interpreted as a single opinion of the characters, rather than an improvement over the crowd-sourced classifications.

Fortunately, there are some inferences we can make about the neural network's differing predictions. For instance, our model in some cases seems to favor raw damage output over other considerations. This can be seen in it's predicted tier for characters like Rosalina & Luma, whose gimmick pairs two different characters together into one, potentially doubling damage output, and Mr. Game & Watch, who has attacks with a probability of dealing extreme high damage. However, other characters, like Chrom, have comparatively high damage output, but were ranked very low, suggesting that our model is looking for some combination of damage and another attribute, likely this character's recovery capabilities. Unfortunately, these observations are theories at best, due to the model's highly non-linear nature.

6 Future Work

Since our primary difficulty in solving this problem was working with a small, but wide dataset, it would be worthwhile to spend more time on refining. It is possible that including more detailed data, such as launch and frame by frame data for each attack, would produce more accurate results, and might make more interpretable models viable. Looking into patch updates that tweak character statistics would also add more potential instances.

Another possible improvement would be the use of different models. Ensembles of different learning models or of the same learning models with different hyperparameters might produce more accurate results. Employing residual connections and dropout in our neural network might further improve the accuracy of our model and lead to more decisive tier classifications for Smash Ultimate.

Looking at the results across all models, reformulating the problem as a regression problem would likely also prove beneficial to the results. Misclassified characters were in many cases only off by one rank, showing that the linear nature of the tier system holds importance to our problem. Formulating this as a classification problem essentially ignored this aspect, which implies that some models were more accurate than raw classification accuracy gives them credit for.

References

[1] Gormley, Z., Hilton, J., Keesling, P., Keli'i, T., & Wilhelm, G (2019). The Comprehensive Super Smash Brothers Character Dataset. [Spreadsheet]. Retrievable from https://docs.google.com/spreadsheets/d/103elr0mhpr14yhiLLib4_KinrFvMXrBGhfcc1fdAJjE/edit?usp=sharing

[2] SmashWiki (n.d). [Wiki]. Retrieved from https://www.ssbwiki.com/

[3] Shaya (2017). 4BR Smash for Wii U Tier List V4. [Forum Thread]. Retrieved from https://smashboards.com/threads/4br-smash-for-wii-u-tier-list-v4.452109/

[4] KuroganeHammer (n.d). [Website]. Retrieved from https://kuroganehammer.com/

[5] RankedBoost (n.d). [Website]. Retrieved from https://rankedboost.com/ssb4-tier-list/

[6] HTC eSports Tier List (n.d). [Web Article]. Retrieved from https://esports.htc.com/articles/esports-tier-list

[7] Super Smash Bros. Ultimate Wiki Guide (January 2019). Retrieved from https://www.ign.com/wikis/super-smash-bros-ultimate/Tier_Lists

A Dataset Versions

A.1 Full Dataset Attributes

The following table covers the full set of attributes collected for each character, excluding only the name of the character and which of the 5 games the instance comes from.

Full Dataset

Attribute Name	Data Type
weight	continuous
archetype	nominal
recovery	nominal
run speed	continuous
jab damage	continuous
jab type	nominal
forward tilt damage	continuous
forward tilt type	nominal
upward tilt damage	continuous
upward tilt type	nominal
downward tilt damage	continuous
downward tilt type	nominal
dash-attack damage	continuous
dash-attack type	nominal
forward smash damage	continuous
forward smash type	nominal
upward smash damage	continuous
upward smash type	nominal
downward smash damage	continuous
downward smash type	nominal
neutral air damage	continuous
neutral air type	nominal
forward air damage	continuous
forward air type	nominal
backward air damage	continuous
backward air type	nominal
upward air damage	continuous
upward air type	nominal
downward air damage	continuous
downward air type	nominal
forward throw damage	continuous
backward throw damage	continuous
upward throw damage	continuous
downward throw damage	continuous
neutral special damage	continuous
neutral special type	nominal
side special damage	continuous
side special type	nominal
upward special damage	continuous
upward special type	nominal
downward special damage	continuous
downward special type	nominal
tier (class label)	nominal

A.2 KNN Forward Wrapper Optimized Dataset

This version of the dataset was compiled by running a KNN model through a forward wrapper algorithm in multiple runs to obtain the subset of attributes that were most often chosen by the wrapper.

Optimized Dataset

Attribute Name	Data Type
weight	continuous
archetype	nominal
recovery	nominal
run speed	continuous
jab damage	continuous
forward tilt damage	continuous
forward tilt type	nominal
downward tilt damage	continuous
dash-attack damage	continuous
dash-attack type	nominal
forward smash type	nominal
upward smash type	nominal
downward smash damage	continuous
downward smash type	nominal
neutral air damage	continuous
neutral air type	nominal
forward air damage	continuous
forward air type	nominal
backward air damage	continuous
backward air type	nominal
upward air damage	continuous
downward air damage	continuous
downward air type	nominal
forward throw damage	continuous
backward throw damage	continuous
upward throw damage	continuous
downward throw damage	continuous
neutral special damage	continuous
neutral special type	nominal
side special damage	continuous
upward special damage	continuous
upward special type	nominal
downward special damage	continuous
downward special type	nominal
tier (class label)	nominal

A.3 Limited Types Dataset Attributes

This version of the dataset tosses out the type information of most moves. We felt this would be a valid option, as most moves are melee attacks, providing relatively little information for a large number of attributes.

Limited Types Dataset

Attribute Name	Data Type
weight	continuous
archetype	nominal
recovery	nominal
run speed	continuous
jab damage	continuous
forward tilt damage	continuous
upward tilt damage	continuous
downward tilt damage	continuous
dash-attack damage	continuous
forward smash damage	continuous
upward smash damage	continuous
downward smash damage	continuous
neutral air damage	continuous
forward air damage	continuous
backward air damage	continuous
upward air damage	continuous
downward air damage	continuous
forward throw damage	continuous
backward throw damage	continuous
upward throw damage	continuous
downward throw damage	continuous
neutral special damage	continuous
neutral special type	nominal
side special damage	continuous
side special type	nominal
upward special damage	continuous
upward special type	nominal
downward special damage	continuous
downward special type	nominal
tier (class label)	nominal

A.4 Micro Dataset Attributes

This dataset limited the attributes to an extreme by tossing out all attack attributes aside from special attacks the characters can perform. This was to gauge how much information could be gleaned from the more highly varied attributes

Micro Dataset

Attribute Name	Data Type
weight	continuous
archetype	nominal
recovery	nominal
run speed	continuous
neutral special damage	continuous
neutral special type	nominal
side special damage	continuous
side special type	nominal
upward special damage	continuous
upward special type	nominal
downward special damage	continuous
downward special type	nominal
tier (class label)	nominal