SABR Analytics Conference, 2019, Phoenix, AZ

DeepBall: Modeling Expectation and Uncertainty with Recurrent Neural Networks

Daniel Calzada, daniel@deepball.net

Abstract
Making reliable preseason batter projections for baseball players is an issue of utmost importance to both teams and fans who seek to infer a player's underlying talent or predict future performance. However, this has proven to be a difficult task due to the high-variance nature of baseball and the lack of abundant, clean data. For this reason, current leading models rely mostly upon expert knowledge. We propose DeepBall, which combines a recurrent neural network with novel regularization and ensemble aggregation. We compare this to Marcel, the industry-standard open-source baseline, and other traditional machine learning techniques, and DeepBall outperforms all. DeepBall is also easily extended to predict multiple years in the future. In addition to predicting expected performances, we apply standard machine learning techniques to extend DeepBall to model uncertainty in these predictions by estimating the maximum-likelihood distribution over potential outcomes for each player. Due to the positive results, we believe that in the future, DeepBall can be beneficial to both teams and fans in modeling expectation and uncertainty. Finally, we discuss potential extensions to the model and directions of future research.

This paper is an overview of the DeepBall algorithm, with excerpts taken from Daniel Calzada's MS thesis. For a more rigorous and technical discussion of the model architecture, learning process, and learning procedure, we refer the reader to the original DeepBall paper [1].

Contents

1. Introduction

Since the inception of the sport, many have sought to describe baseball quantitatively with statistics. In the mid-20th century, the usefulness of a traditional statistic, batting average, was criticized for weighing all hits equally and not accounting for other contributions, such as drawing walks, thus not measuring a batter's true contribution to his team.

Sabermetricians developed new offensive statistics such as OBP, OPS, and later WAR [2, 3] and wOBA [4] to account for this gap. While these are useful descriptive statistics, they have limited predictive quality, and so anyone who wishes to project future performance must find it elsewhere. To this end, sabermetricians have developed multiple prediction systems based on domain knowledge.

We seek to explore the answers to two major questions: can a machine learning model isolate a batter's true talent from raw, noisy statistics, and can it use this talent to generate a prediction for the next season? We claim the latter objective is sufficient: if a model can generate an accurate prediction, it must have an internalized representation of the batter's true talent. Answering this question is of utmost importance to many people and groups, including MLB team front offices, who seek to make informed trade and playing time decisions optimal for short-term and long-term team performance; and fantasy baseball players, who seek to build an optimal roster for their league. For this reason, we propose a new projection system, DeepBall, to generate these predictions.

1.1 Related Work

Many researchers have studied the problem of preseason predictions. One popular system is Marcel, proposed by Tom Tango [5], which is designed to serve as a baseline for player season projection tasks. In generating a prediction for one statistic, it uses a 5/4/3 weighted combination of the player's totals from the last 3 seasons, taking into account the league averages over that time. Since it is open-source and thus can be retroactively evaluated, we will use this as the primary industry-standard baseline. Interestingly, more complex systems struggle to significantly improve upon Marcel despite its simple design [6, 7].

PECOTA was developed by Nate Silver, then with Baseball Prospectus. To create a projection for a player, it explicitly searches for similar players [8] and uses their subsequent seasons to predict how the new player will perform. With simple alterations, this algorithm can be used to predict other useful things such as breakout rate. While PECOTA is a popular projection system, most of its results are held behind a paywall, so we did not examine it in depth.

Steamer [9] is one of the primary public projection systems. The system resembles Marcel by using a linear combination of five previous years including a regression towards the league mean [8, 10]. The linear weights, however, are not fixed but are determined by a regression over past players.

Another system is ZiPS, developed by Dan Szymborski with Baseball Think Factory in 2003. In a sense, this system is a hybrid between Marcel and PECOTA. It functions by computing a weighted average of the player's last few years and finding comparable players like PECOTA, using these comparisons to generate its predictions [8]. ZiPS uses minor league data [11] and heavily relies upon expert knowledge when determining player similarity [12].

Though PECOTA, Steamer, and ZiPS provide additional perspective regarding industry-standard approaches for solving this problem, we are unable to comprehensively and fairly compare their results to DeepBall for two reasons: since they are kept closed, the algorithm and implementation details are not known to us, and we have no access to additional proprietary data used for training these systems.

1.2 Problem Formulation

In this paper, we generalize the prediction problem to predicting a player's statistics for some future season S using only information known before the start of season $S-k+1$ for some k. We can interpret the parameter k as the *shift factor* for our predictions: setting $k=1$ creates a projection system that estimates the subsequent year, setting $k=2$ yields a system predicting two seasons in the future, and so on. Each player's career consists of a sequence of seasons, so we can frame this as a time-series prediction problem. We will be predicting batter performance in eight categories. We will predict five rate statistics (AVG, SLG, K%, BB%, and OBP) and three count stats (PA, Games, and oWAR). These values together give us a well-rounded perspective on a player's overall offensive contribution in various aspects of the game.

A hurdle that presents itself is that our data is limited and thus, on its surface, not conducive to analysis with neural networks. There have only been 7901 players who have debuted between 1958 and 2014 with at least one career plate appearance, and these players have combined to bat in only 44336 batter/seasons. Therefore, there are only 7901 independent training sequences and 44336 dependent examples. This issue was noted by [13] when suggesting that complex neural networks may not have success in this problem. However, with proper care, this problem may be mitigated.

2. Data Sources

All play-by-play and pitch data used for training and evaluation comes from Retrosheet [14]. Since Retrosheet does not have complete play-by-play data from before 1957, we only trained on players who began their career during or after 1958, only using their regular season statistics. The Sean Lahman database was used for player biological information [15]. Finally, historical and up-to-date sabermetric data is freely available from Baseball Reference [3]. All players who appeared in any game were used as training or test data.

To compare DeepBall's performance against Marcel, we used a custom implementation of the algorithm [16].

3. Approach

In this section, we will outline our general approach in creating DeepBall as a recurrent neural network. We will discuss the inputs, outputs, and model architecture at a high level, and we will outline our ensemble technique which reduces the variance of the models.

3.1 Variables

There are many variables used by our model as inputs and outputs, and these are highlighted below.

- *out_stats* contains the five rate statistics we are interested in predicting. Because these values may be unreliable for small sample sizes, we used the square root of the batter's plate appearances for that season as sample weights. Furthermore, to account for the shifting league averages discussed in section 1.2, we predicted players' statistics relative to the league average as opposed to their absolute values [17]. As such, we normalized each of these statistics independently by dividing by the mean so that $\mu = 1$ for each season.[1] Furthermore, we standardize them to the same variance to not give more importance to statistics with more variance (e.g. SLG) than to those with less (e.g. BB%).

- *out_counts* contains the three output counts we are interested in predicting as described in section 1.2.

- *out_fielding* is the number of games started by this player in each season grouped by starting fielding position. We include the number of games started on the bench, when the player comes in as a substitute, including relief pitchers (position 0), each of the nine standard fielding positions 1-9, and the designated hitter is assigned position 10.

- Results of the player's previous season: various offensive counting and sabermetric statistics, elementary base-running and plate discipline statistics, games started at each fielding position, splits in common basic offensive statistics, and various league averages for that year.

- The combination of park factors for all stadiums played in by that player the previous season, weighted by plate appearances in that stadium. We use least squares park factors for this task [18].

- Factors relating to the first team for which the hitter will play in the coming year. First, this includes the park factors for the team's stadium which are determined by averaging the least-squares park factors for that stadium from the three previous years. This also includes two flags for whether interleague play will be present in the coming year and whether the player's team will use a DH in their home field.

- The player's age and projected fielding position for the upcoming season, based on depth charts. The studies done by [19, 20] are a part of the significant body of research regarding player aging curves, and we elected to generalize this concept by learning an arbitrary feature vector for each age, allowing it to encode different peak ages in different areas.

- The player's biological data: height, weight, handedness, and whether they are a pitcher.

[1]The task of converting the relative predictions back to absolute predictions is simple. A season-to-season model of league averages is required, which can be created using a linear combination of previous years' averages as suggested by [12]. In our implementation, we used Marcel's predicted league averages and scaled our outputs appropriately.

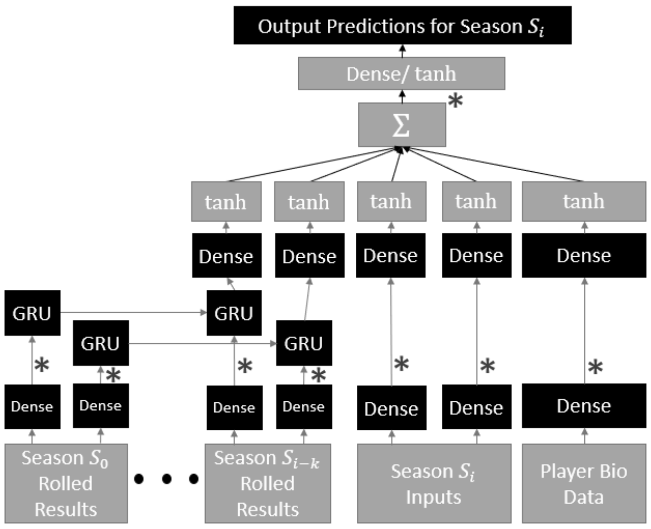


Figure 1. A basic overview of the recurrent neural network architecture

Of these, only *out_stats*, *out_counts*, and *out_fielding* are not known before the start of the season. Therefore, these will be the outputs of the model, and DeepBall predict their values from the other input values.

3.2 Learning Process

The information flow and basic architecture of our neural network is outlined in Figure 1. In this diagram, data flows from the inputs (bottom) to the outputs (top), and each step represents a mathematical operation applied to the previous step's output. These mathematical operations can be adjusted, or learned, to minimize the prediction error on a dataset. By training our model on many real-life player careers and forcing it to reduce the prediction error on these players, we will adjust the operations so as to return reliable and accurate predictions when can be applied to unseen future data. These adjustments are made in many passes, or epochs, through the same training data, until the model's operations have converged.

The goal of machine learning is to uncover, generalize, and apply trends learned from raw data. A difficulty in this is the

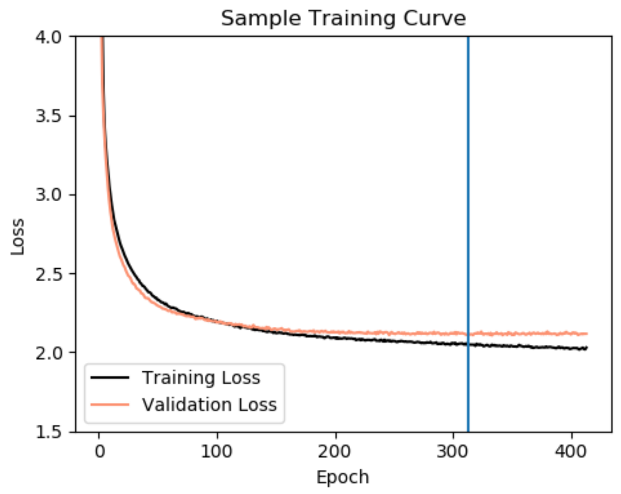
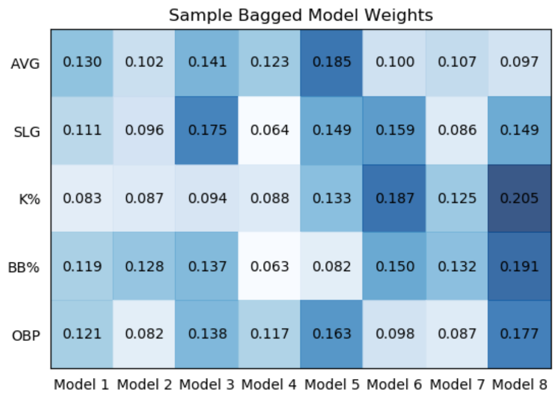


Figure 2. A sample training curve. The vertical line represents the epoch with the lowest validation loss. In our application, this is considered the best model.


Sample Bagged Model Weights

	Model 1	Model 2	Model 3	Model 4	Model 5	Model 6	Model 7	Model 8
AVG	0.130	0.102	0.141	0.123	0.185	0.100	0.107	0.097
SLG	0.111	0.096	0.175	0.064	0.149	0.159	0.086	0.149
K%	0.083	0.087	0.094	0.088	0.133	0.187	0.125	0.205
BB%	0.119	0.128	0.137	0.063	0.082	0.150	0.132	0.191
OBP	0.121	0.082	0.138	0.117	0.163	0.098	0.087	0.177

Figure 3. Weight values of the 8 models being aggregated for *out_stats* ($k = 1$), trained using data from 1958-2014. Notice that each model has areas of strength and weakness and some models are overall stronger than others. Note also that the sums of the rows are approximately 1.

problem of overfitting: if a model is sufficiently complex, this learning process may yield a model that has essentially memorized the training data but has not generalized any knowledge. When applied to previously-unseen examples, it will fail. To combat this, we use two techniques. The first is regularization, which encourages the training process to yield a simpler model which is more likely to generalize. The second is model selection using a held-out validation dataset. We reserved approximately 20% of our data to test the model throughout the training process. When a model yields good predictions on this validation data, we assume that it is able to generalize, and we keep the model that has the lowest validation error.

3.3 Training

Our implementation was based on Keras [21] and TensorFlow [22]. For each season being predicted, different models were trained. For example, models designed to make 2016 predictions were trained using 1958-2015 data, whereas those making 2017 predictions were trained separately on 1958-2016. We did this to give each model as much training data as possible.

Each epoch, the training examples were used to update the parameters of the operations, and the model which performed best on the validation data was chosen. A sample training curve is shown in Figure 2.

3.4 Bagging

Due to how neural networks are initialized and trained, two networks trained on the same data can make contrasting predictions for the same player. To counter this, we form an ensemble of models that learns a weighted mean of the networks' predictions [23]. For each architecture, we trained 8 different models on the same data but from different random starting points, and for each statistic we learned the optimal weights for each model so as to minimize the error on the validation set. Some weights have been visualized in Figure 3.

4. Experiments on Expectation

To evaluate DeepBall's performance in predicting the players' expected performances, we have implemented baseline systems to which it will be compared. The first baseline is Marcel, which was discussed in detail in section 1.1 and is commonly used as a standard baseline.

Our second baseline is a Random Forest regression [24], trained on the same inputs, outputs, and sample weights as DeepBall. Since our problem deals with sequential data that cannot be represented with a traditional random forest, we kept data from the previous four seasons, flattened it, and replaced missing values with zeros.

The third baseline is an RNN whose simple architecture is laid out in Figure 4. The final layers and loss functions were the same as DeepBall. Like DeepBall, it used the validation set for model selection, and we bagged 8 independently-trained models to make it comparable to DeepBall.

The 2015-2018 seasons were used as test data. The errors were weighted according to each player's sample weight (plate appearances). In total, there were 959 distinct players and 2489 distinct batter/seasons in the test set. We evaluated *out_stats* using mean squared error and we evaluated *out_counts* using mean absolute error.

4.1 MSE for Rate Statistics

The first metric we used to evaluate our results is weighted mean squared error for both $k = 1$ and $k = 2$. This metric is useful because it corresponds to the loss function we used to train our model and it is the standard loss function for

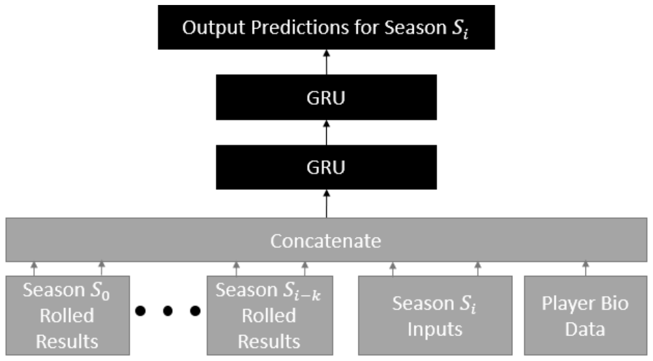

Output Predictions for Season S_i
GRU
GRU
Concatenate
Season S_0 Rolled Results
Season S_{i-k} Rolled Results
Season S_i Inputs
Player Bio Data

Figure 4. The architecture for the baseline RNN model

regression analysis. The results for position players are shown in Table 1.

As expected, the error for $k = 2$ is universally larger than that for $k = 1$. We also see that the neural models (DeepBall and the RNN baseline) significantly outperform Marcel and surprisingly, there is little difference in the errors between the two. This highlights the strength of the neural models, which have the ability to properly combine high-dimensional input data to create high-quality predictions. These results also suggest that DeepBall is not sensitive to changes in the architecture of the network (the main source of domain knowledge used in the model), since a carefully-designed network only slightly outperforms a network with a simple architecture.

4.2 MAE for Counts

Second, we evaluated the mean absolute error of the count statistics. Unlike the MSE experiment, we did not use sample weights or normalize to the league average. The results for position players relative to Marcel are shown in Table 2.

From these results, we see that the gap between Marcel and the other models is significantly wider than the first experiment, possibly attributable to Marcel's simple model of playing time that in no way accounts for performance in other areas. We generally observe the same ranking of the systems as in the previous experiment, with DeepBall performing the best.

We gather two significant insights from these experiments. First, we have seen that neural models perform better than both a simple traditional baseball model, Marcel, and a traditional machine learning regression, the Random Forest. Second, we see that the RNN and DeepBall perform at a similar level, but DeepBall's errors are slightly lower than the RNN's. Since DeepBall was designed with domain knowledge and the RNN was not (with the exception of the selection of features, which all models except Marcel shared), we see that domain knowledge is useful in creating a reliable projection system, though even without domain knowledge we can still build a good system.

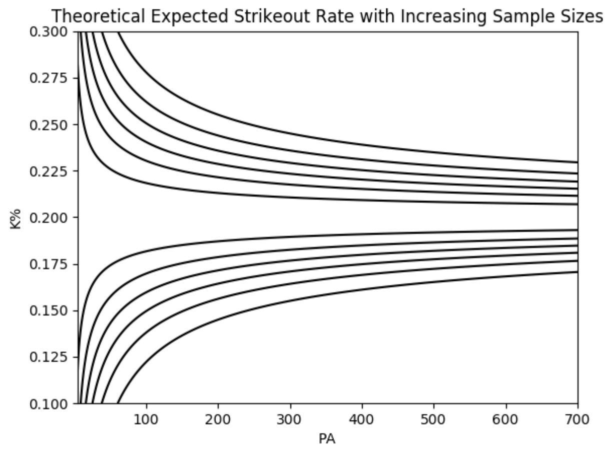


Figure 5. If a player's talent level dictates that they will have K% $= 0.2$, we will most likely not observe this exact value. By modeling the result of each as a Bernoulli process, this contour plot of probabilities marginalized over PA shows that estimated variances vary proportional to $\frac{1}{\text{PA}}$.

5. Modeling Uncertainty

While the analyses in section 4 and in the industry at large appear to make the assumption that predictions are homoscedastic, this assumption does not hold in theory or in practice. In reality, heteroscedasticity is apparent and abundant in many aspects of the problem in that the variance expected will vary both within and among players' predictions.

The first cause of this is that rate statistics with smaller sample sizes will generally have higher error rates: as a batter has fewer plate appearances, the resulting statistic is more vulnerable to noisy outcomes in any individual plate appearance (Figure 5).

A second cause is limited information regarding rookies or uncertainty about aging players. For example, we would expect young players to have a higher variance in possible outcomes since they have not established themselves at the major league level. Uncertainty would also be expected in aging players, partly because there are fewer training examples for older players and their role on their team may become more uncertain.

Finally, some statistics and therefore errors are correlated, and understanding these correlations is useful. For example, the correlation between AVG and SLG is visualized in Figure 6. However, these are not the only correlated statistics: OBP and BB% are very correlated, as are PA and oWAR. We also expect rate statistics to be correlated with playing time because better players often receive more playing time. We also expect these correlations to vary from player to player: power hitters like J.D. Martinez may have a different AVG/SLG correlation than on-base hitters like Ichiro Suzuki.

In summary, we expect the distribution of possible outcomes to be heteroscedastic with respect to the player, season,

k = 1	AVG	SLG	K%	BB%	OBP
Marcel	0.02910	0.04334	0.06718	0.11801	0.02109
RF	-3.74%	-4.30%	-5.85%	-8.57%	-3.01%
RNN	6.67%	6.58%	8.01%	2.82%	7.62%
DeepBall	**7.46%**	**7.38%**	**14.58%**	**5.43%**	**10.97%**
k = 2	AVG	SLG	K%	BB%	OBP
Marcel	0.03043	0.04572	0.08017	0.13040	0.02219
RF	-4.02%	-1.99%	1.67%	-5.08%	-1.90%
RNN	**9.55%**	9.46%	12.59%	6.97%	**10.94%**
DeepBall	9.33%	**10.34%**	**15.36%**	**8.32%**	10.78%

Table 1. Marcel MSE performance (first row), and MSE relative improvement vs. Marcel (subsequent rows) for $k \in \{1,2\}$. We observe that DeepBall outperforms Marcel and all other baselines for $k = 1$ and, along with the RNN, $k = 2$.

k = 1	PA	G	oWAR
Marcel	142.18	34.57	N/A
RF	19.58%	15.91%	0.8788
RNN	25.31%	18.81%	0.8510
DeepBall	**28.42%**	**21.45%**	**0.8232**
k = 2	PA	G	oWAR
Marcel	167.05	40.24	N/A
RF	20.94%	19.66%	0.9515
RNN	29.95%	26.11%	0.8891
DeepBall	**30.96%**	**26.70%**	**0.8729**

Table 2. Marcel MAE performance for *out_counts* (first row), and relative MAE improvement vs. Marcel (subsequent rows). oWAR is not predicted by Marcel, so absolute errors are presented instead of relative improvement. We see that the systems' relative rankings remained the same from the first experiment.

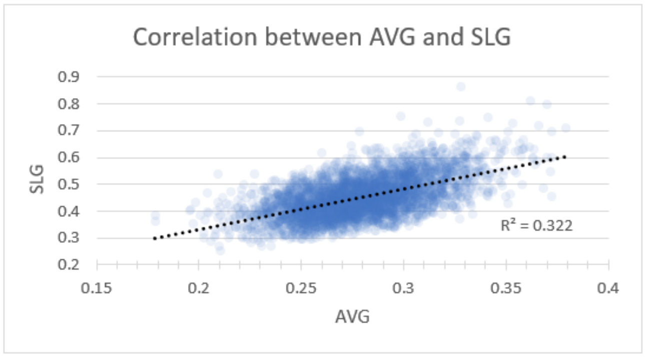


Figure 6. The batting average (AVG) and slugging percentage (SLG) for batter seasons since 1990 with at least 500 PA. It is apparent that these two statistics are correlated, so an error in one may suggest an error in the other.

and statistic. Therefore, it is logical to seek to model not only the expected outcomes, but the expected error in these predictions. In the context of team front offices, knowing the uncertainty associated with a player's prediction would provide valuable insight if a team is investigating a possible trade: some teams may be seeking a higher risk player with a higher upside, but some may be seeking a player whose outlook is relatively certain. For fans, understanding uncertainty will be useful when building a fantasy roster or tracking and comparing a player's career over time.

Ideally, a projection system should not only be able to give an expected outcome but a measure of uncertainty as well. Marcel and Steamer give reliability scores, where a high value indicates high confidence or low uncertainty. Similarly, ZiPS, in addition to expected results, gives thresholds in many areas along with probabilities that a player will exceed these thresholds. Likewise, we would like DeepBall to predict uncertainty.

Unlike other systems, we generalize these by modeling a fully-defined probability distribution of a player's possible outcomes for a season. By visualizing this distribution, analysts may obtain a deeper understanding of what to expect from a player. Likewise, it allows us to quantitatively compare the predictions of batter seasons not simply by their expected outcome but also by their variance and skewness.

5.1 Approach

We wish to model the vector

$$\mathbf{s} = [(\text{PA})\text{AVG}, (\text{PA})\text{SLG}, (\text{PA})\text{K\%}, (\text{PA})\text{BB\%}, (\text{PA})\text{OBP}, \text{PA}, \text{G}, \text{oWAR}] \quad (1)$$

as a joint distribution rather than each statistic in isolation, so we add a new output layer to the original architecture dedicated to making second-order predictions as well. That is, for each output s_i, rather than simply outputting $\mathbb{E}[s_i]$, we will predict both $\mathbb{E}[s_i]$ and $\text{Var}[s_i]$. To account for the correlations among output statistics, we can also predict the covariance $\text{Cov}[s_i, s_j]$ between any outputs s_i and s_j.

Based on these requirements, a natural first step would be to model $\mathbf{s}$ as a multivariate Gaussian. However, we do not wish to constrain our model to only modeling linear correlations between the output statistics. Instead, we model $\mathbf{s}$ as a mixture of Gaussians distribution with full covariance matrices. By choosing the number of Gaussians wisely, this distribution is sufficiently general to model a complex distribution of outcomes for a given player while having few enough parameters to make learning it tractable. Each player will then not only have his own expected predictions in each stat as discussed in the previous sections, but also his own unique distribution over all potential outcomes which can be analyzed.

5.2 Training

Like before, we created eight separate models for $n \in \{1,2,4,8\}$ and each year in the range $[2015, 2018]$ using the same architecture, replacing the original loss functions with a mixture of Gaussian loss (the negative log likelihood) and leaving the rest of the model architecture and training procedure from section 3 unchanged, thus encouraging the model to learn the optimal parameters for each distribution.

6. Experiments on Uncertainty

In this section, we show multiple ways of analyzing the distribution predictions. We evaluate the weighted MSE of the distribution means, similar to section 4.1, and we examine anecdotal results qualitatively, therein observing heteroscedasticity and DeepBall ability to account for this. For brevity, all analyses are done with $k = 1$, though $k > 1$ models can be obtained using the same steps as before.

6.1 Mean Squared Error

To compare the mean rate statistic predictions from the mixture of Gaussian distributions with the results from section 4, we sampled from the output distributions[2] from all models and used the same bagging technique from section 3.4 to aggregate these eight separate results.

In addition, we have learned a ridge regression, DeepBall+, between the original system, the RNN baseline, and the mixture of Gaussian predictions for $n \in \{2,4,8\}$. This regression was trained using all player/seasons in the validation set from 2000 to 2014. Two DeepBall+ models were trained: one for batters, and one for pitchers. Table 3 contains the results of the analysis for batters.[3]

From these results, we observe that for most statistics, any $n > 1$ gives results comparable to DeepBall, and sometimes the sampled means from the mixture of Gaussians model outperform DeepBall. This supports our hypothesis that using $n = 1$ Gaussian is too simplistic and does not accurately represent the data distribution.

Additionally, we see that DeepBall+ has the lowest and most stable error rates in all statistics compared to any single model. Given this, in terms of weighted mean squared error, it is clear that DeepBall+ is the most accurate, robust system analyzed in this paper.

6.2 Season Case Studies

Beyond the quantitative analysis described above, we seek to evaluate our predicted distributions qualitatively. To do so, we will examine contour plots of some players' 2018 distributions. We selected Jason Heyward, Mike Trout, and Matt Olson as representative players and visualized the probability density function for $n = 8$ Gaussians (Figure 7). Because visualizing an 8-dimensional distribution is impossible, we have selected three pairs of representative statistics to visualize: PA and oWAR, PA and K%, and PA and K% directly.

Each of these players has a unique past and an equally-unique 2018 outlook, and these outlooks manifest themselves in the distributions. Jason Heyward began his career as a rookie sensation but has increasingly struggled as his career has progressed. As such, DeepBall projected him to only receive a part-time load of 489 PA in 2018. Interestingly, his strikeout rate and playing time predictions are correlated negatively (Figure 7g): the lower his strikeout rate, the better his overall performance will likely be and the more playing time he will receive. In 2018, he did lower his strikeout rate marginally in an overall slightly better season, and he had exactly 489 PA.

Often heralded the best player of his generation, Mike Trout, in his last seven seasons, has never posted an oWAR below 7.3. As seen from the graphs, his oWAR predictions for 2018 were very high and he was easily projected to be a full-time player. Furthermore, compared with Heyward, Trout's strikeout rate and plate appearances are not as correlated, since established players will generally be given a full-time load regardless of their performance. Additionally, our model predicted a negatively-skewed distribution over PA, reserving some probability density for less playing time, possible if an

[2]By modeling, for example, (PA)K% rather than K% directly, we lose the benefit of analyzing K% directly. To analyze the results, we sample from the distribution and aggregate those samples, though this is computationally expensive.

[3]Pitchers are not included in this analysis since their offensive statistics are not predicted by Marcel.

k = 1	AVG	SLG	K%	BB%	OBP
Marcel	0.02910	0.04334	0.06718	0.11801	0.02109
DeepBall+	*8.25%*	*9.56%*	*16.29%*	*6.27%*	*9.32%*
RF	-3.74%	-4.30%	-5.85%	-8.57%	-3.01%
RNN	6.67%	6.58%	8.01%	2.82%	7.62%
DeepBall	7.46%	7.38%	14.58%	5.43%	8.21%
$\text{MOG}_{n=1}$	2.40%	3.80%	11.62%	-3.56%	3.38%
$\text{MOG}_{n=2}$	6.26%	8.35%	14.90%	4.26%	6.96%
$\text{MOG}_{n=4}$	7.63%	9.16%	15.60%	**5.46%**	8.26%
$\text{MOG}_{n=8}$	**8.07%**	**9.41%**	**15.91%**	5.17%	**8.62%**

Table 3. MSE relative improvement for our baselines, original DeepBall model, and the means of the mixture of Gaussian models for $n \in \{1,2,4,8\}$. For $n > 1$, the Gaussian means have approximately the same error as the model trained using MSE loss. DeepBall+ has been implemented as a linear combination of the systems listed, and it has the highest improvement in all statistics.

injury occurs. Indeed, his 2018 results were impressive again, and he overperformed his oWAR predictions.

Finally, Matt Olson, after reaching the Major Leagues in 2016, played in 59 games in 2017, hitting a notable 24 home runs and contributing 2.1 oWAR in only 216 PA. Despite his youth and associated uncertainty, DeepBall projected him to be nearly a full-time player and contribute 2.63 oWAR in 2018. However, it displayed more uncertainty than Heyward's projection because little was known about Matt Olson. Similar to Heyward, Figure 7i suggests a slight negative correlation between playing time and strikeout rate, which we attribute to the same assumption as seen with Heyward. Though he was able to lower his strikeout rate in 2018, he only contributed 2.7 oWAR in 660 PA, underperforming his projections in that regard.

7. Applications

Of the innumerable applications of this approach, we give some that we believe are interesting, useful, or both.

7.1 Park-Neutral Expectation Evaluation

The most natural application to what we have discussed is predicting the expected performance of players. However, since our network is simply a regression combining the results from previous seasons and inputs for the upcoming season, by adjusting any of the inputs, we can simulate how the players will perform in the upcoming season under different circumstances. Using this, we can study the anecdotal effects of any variable by changing it and comparing the resulting expectations or distributions, thus isolating various environment-specific components of a player's projection.

Of particular interest are the park factors, which if set to 0's will generate predictions for a park-neutral environment. For example, we wish to examine how Charlie Blackmon would perform in a park-neutral environment away from hitter-friendly Coors Field. We have generated his SLG predictions both at Coors Field and in a neutral environment (Figure 8).

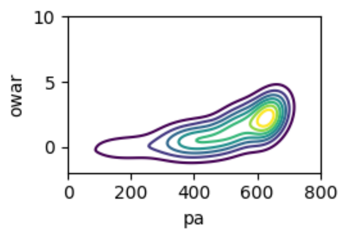
We observe that Blackmon's predictions in a park-neutral environment are lower than at Coors Field, and we claim that his park-neutral projections better reflect Blackmon's talent. Generating predictions for a different stadium is a useful feature for a team front office considering a potential trade.

7.2 Distribution Inspection

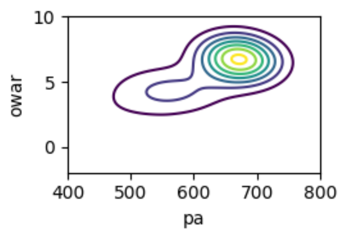
Another application of modeling probability distributions is inspecting players' distributions and inferring meaning from them or comparing them with other players' distributions. Due to our claims regarding heteroscedasticity, we expect to find two players with similar expectation predictions but different distributions of outcomes. Indeed, one such pair of players is Daniel Descalso and Yandy Diaz, who had nearly the same playing time and oWAR mean predictions for 2018 but had unique distributions. Daniel Descalso was projected to receive 254 PA and contribute 0.35 oWAR 2018, whereas Yandy Diaz was projected to receive 232 PA and contribute 0.42 oWAR. On the surface, this might seem to indicate that they may be of similar talent levels or have similar outlooks for the upcoming season. However, upon inspecting the distributions themselves (Figure 9), we see that this is not the case.

We observe that Diaz's PA distribution is skewed positively whereas Descalso's is more symmetric, because Descalso is an established player but Diaz is not. Therefore, under this model, even though Descalso's oWAR mean is lower than Diaz's, Descalso had a slightly higher probability of achieving 1 oWAR (≈ 0.215, compared to ≈ 0.207), though Diaz had a slightly higher probability of achieving 2 oWAR (≈ 0.071, compared to ≈ 0.058). If we were building a team and were presented the option to choose either of these players, knowing the risks associated with each player's upcoming season is a valuable resource.[4]

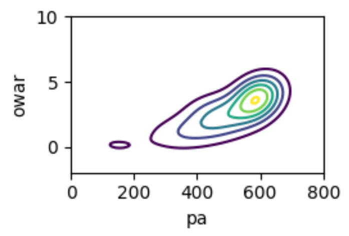
[4]Of course, this analysis only considers the upcoming year and does not consider subsequent seasons or intangibles that would be valuable to a team front office, such as prior MLB experience and leadership traits. However, we are not concerned with DeepBall modeling these intangibles.



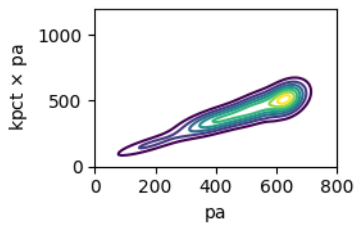
(a) Heyward, oWAR



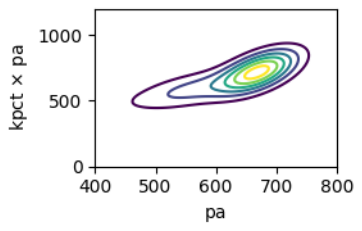
(b) Trout, oWAR



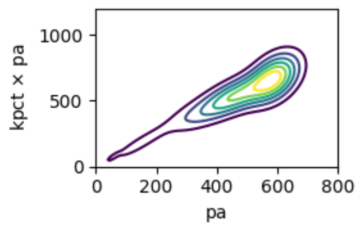
(c) Olson, oWAR



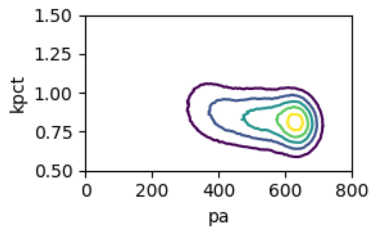
(d) Heyward, (K%)(PA)



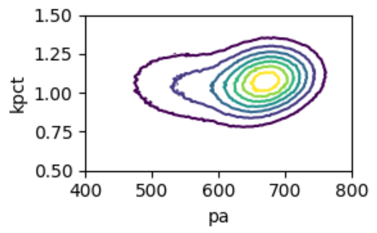
(e) Trout, (K%)(PA)



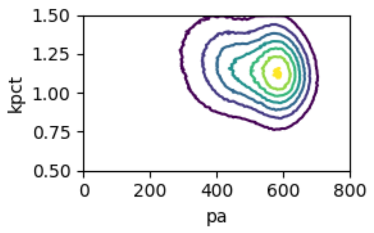
(f) Olson, (K%)(PA)



(g) Heyward, K% relative to league average



(h) Trout, K% relative to league average



(i) Olson, K% relative to league average

Figure 7. Visualizations of 2018 predictions for Jason Heyward, Mike Trout, and Matt Olson using $n = 8$ Gaussians. Each player has a different outlook, so the distributions vary both qualitatively and quantitatively.

7.3 Divergence of Distributions

Generalizing the approach above, we wish to evaluate the dissimilarity of any two players' predictions. As seen, evaluating the distance between the means is not sufficient, so we must compute the divergence of the expected distributions. For this, we will use the Jensen-Shannon Divergence as a dissimilarity metric between two distributions. In simple terms, the Jensen-Shannon Divergence measures how easily the samples of two distributions can be differentiated. Two identical distributions will have a divergence of 0, and two completely different predictions (e.g. Mike Trout and any relief pitcher) will have a divergence close to 1. We will construct a square dissimilarity matrix A for all players, where the entry in row i and column j is the divergence between player i and j's distributions. We can search through A to find players with similar outlooks. For example, Joey Gallo and Miguel Sano had very similar park-neutral distributions for 2018.

We can also apply t-SNE scaling to A, obtaining a d-dimensional point for each player [25] and visualize or cluster these points, finding groups of similar players. We show the two-dimensional visualization of all third basemen for 2019 (Figure 10). As expected, similar players are grouped together.

8. Conclusions and Future Work

We have seen how DeepBall is able to tackle two critical problems in the baseball industry. Using surface-level, noisy statistics from the past, DeepBall generated reliable predictions for expected performance for any player for one or more future seasons. In this task, it outperformed both traditional machine learning algorithms and the industry-standard baseline, Marcel, in every category examined. Additionally, we have shown the flexibility of DeepBall by developing sister model to estimate a complex probability distribution over possible outcomes for a player's future season.

There are a few areas for improvement upon this first iteration of DeepBall. Notably, we are missing at least two major data sources: injury data and minor league data. Currently, the only way DeepBall can detect an injury is by observing reduced playing time in a season. However, not all injuries are equal and reduced playing time can have other causes, and DeepBall cannot distinguish among these. Additionally, having minor league data would improve DeepBall's projections for young players.

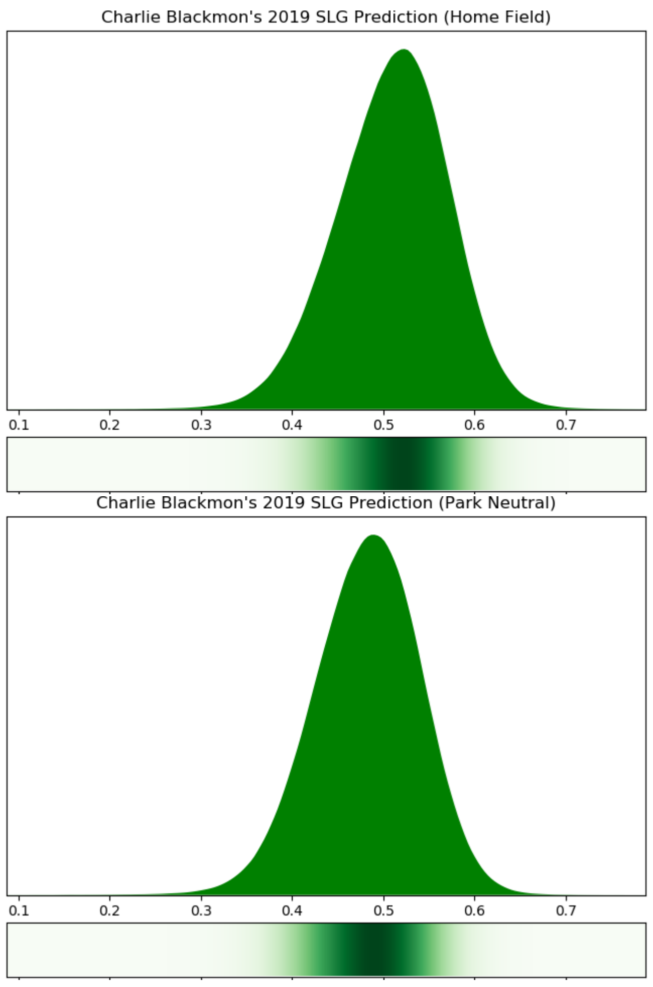


Figure 8. Charlie Blackmon's 2018 SLG predictions in Coors Field in Denver, compared to his predictions in a park-neutral environment. As expected, we see that Blackmon's park-neutral projections are lower than his projections at Coors Field.

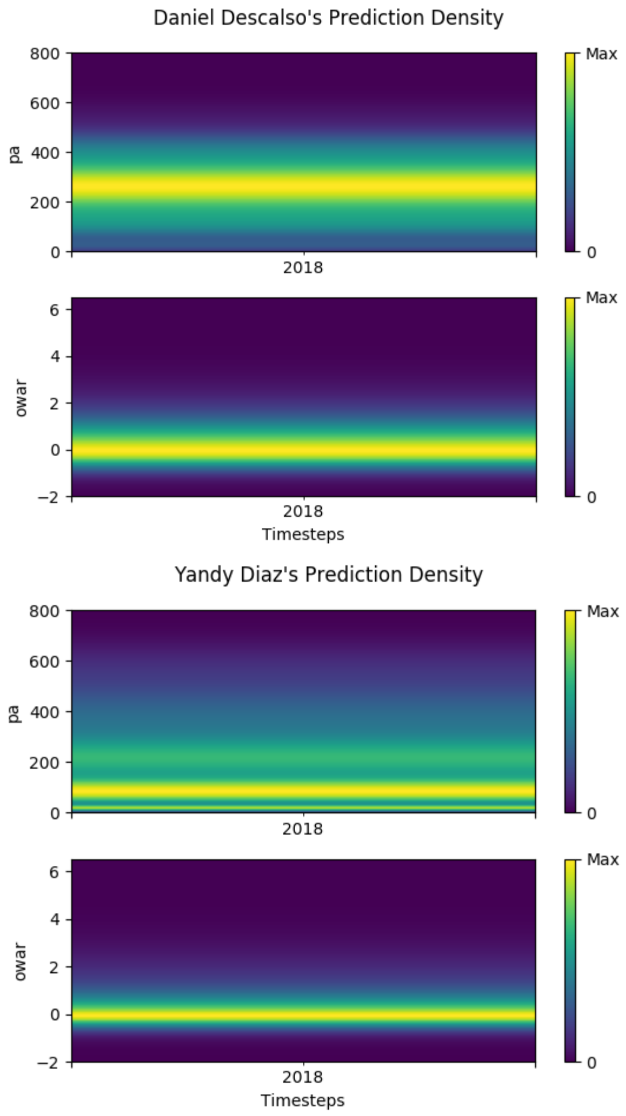


Figure 9. Even though the means of the 2018 projection distributions for Daniel Descalso and Yandy Diaz are almost identical, closer inspection reveals that they have divergent outlooks.

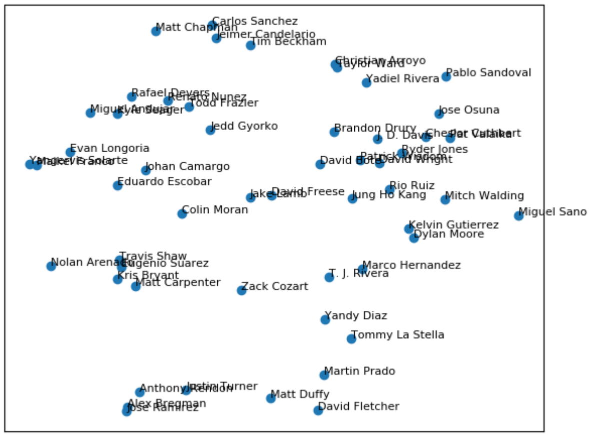


Figure 10. We have used t-SNE to visualize the dissimilarity matrix *A* between players' joint SLG, K%, OBP, PA, and oWAR distributions for 2019. We observe that similar players are grouped together.

Second, DeepBall lacks breadth in the statistics it predicts. It is reasonable for a fan to desire an estimate of how many home runs a player will hit, but this is not an output statistic in *out_counts* and thus cannot be obtained without modifications and retraining the models. A possible solution is to approximate any new statistic by regressing on the existing output statistics. For example, given a player's AVG, SLG, PA, and oWAR, one could likely arrive at a decent prediction for the player's home runs.

Furthermore, DeepBall only predicts offensive statistics, whereas all major systems predict pitching statistics as well. This significant hole in the model can be patched by applying the same RNN architecture to predict pitcher performances. We reserve this task for a future work.

This work also opens the door to some areas of potential future research. First, our network architecture allows for the possibility of generating latent representations of the data. Transfer learning is an area active research, and we may be able to examine the latent values within the network to be used for other prediction tasks. Second, we may be able to interpret this model to gain insights regarding which input statistics are most important in predicting future performance, and we can generate new statistics based on this information.

In summary, DeepBall has applied recurrent neural networks and performed well on batter projection tasks including modeling general uncertainty, and we believe it shows great potential to be extended into other areas of baseball prediction in the near future.

Acknowledgments

We would like to thank Microsoft Azure for providing computational support for the development of DeepBall, as well as the volunteers at Retrosheet who have spent countless hours capturing and recording baseball data from the last century. Our research would also not have been possible without the work of the numerous researchers and enthusiasts who have studied the numbers, statistics, and trends in the game of baseball, and whose work has paved the way for the baseball side of this research. I would also like to thank my thesis advisor, Oluwasanmi Koyejo, for his input, guidance, and instruction throughout the process of developing the first iteration of DeepBall. Without this contribution, this work would not have been possible. Finally, many thanks to Clark Csiki, Ashli Cean Landa, and Tyler Hobbs, who have invested much time into the editing of this paper and the subsequent publishing of the predictions online.

References

[1] Daniel Calzada. Deepball: Modeling expectation and uncertainty in baseball with recurrent neural networks, 2018.

[2] Fangraphs. What is war? `http://www.fangraphs.com/library/misc/war/`, 2013. [Online; accessed 15-Aug-2017].

[3] Baseball-Reference. War explained. `https://www.baseball-reference.com/about/war_explained.shtml`, 2017. [Online; accessed 1-Feb-2018].

[4] Fangraphs. woba. `http://www.fangraphs.com/library/offense/woba/`, 2013. [Online; accessed 15-Aug-2017].

[5] Tom Tango. `http://tangotiger.net/marcel/`, 2001. [Online; accessed 24-May-2017].

[6] Henry Druschel. A million monkeys at a million spreadsheets: 2015's projection systems in review, part one. `http://www.beyondtheboxscore.com/2016/1/14/10733872/steamer-zips-pecota-marcel-projections-review`, 2016. [Online; accessed 30-May-2017].

[7] Henry Druschel. Grading the projections: 2016. `http://www.beyondtheboxscore.com/2017/1/8/14189138/pecota-zips-steamer-marcel-projection-systems-graded`, 2017. [Online; accessed 30-May-2017].

[8] Henry Druschel. A guide to the projection systems. `http://www.beyondtheboxscore.com/2016/2/22/11079186/projections-marcel-pecota-zips-steamer-expl`, 2016. [Online; accessed 24-May-2017].

[9] Jared Cross et al. `http://steamerprojections.com/blog/about-2/`, 2008. [Online; accessed 24-May-2017].

[10] Luke Gloeckner. Introduction to the baseball projections — 2014 version. `http://mrcheatsheet.com/2014/01/22/introduction-to-the-baseball-projections-2014-v` 2014. [Online; accessed 31-May-2017].

[11] Dash Davidson et al. `http://steamerprojections.com/blog/evaluation-of-2009-hitter-forecasts/`, 2010. [Online; accessed 25-May-2017].

[12] Dan Szymborski. Zips q and a. `http://www.baseballthinkfactory.org/szymborski/zipsqa.rtf`, 2008. [Online; accessed 30-May-2017].

[13] Nikolai Yakovenko. Machine learning for baseball (my story). `https://medium.com/@Moscow25/machine-learning-for-baseball-my-story-84dbe075` 2017. [Online; accessed 11-Feb-2018].

[14] Retrosheet, 2017. The information used here was obtained free of charge from and is copyrighted by Retrosheet. Interested parties may contact Retrosheet at "`http://www.retrosheet.org`".

[15] Sean Lahman. `http://www.seanlahman.com/baseball-archive/statistics/`, 2016.

[16] Tom Tango. `http://www.tangotiger.net/archives/stud0346.shtml`, 2004. [Online; accessed 24-May-2017].

[17] Tom Tango. Forecast evaluations. `http://www.insidethebook.com/ee/index.php/site/comments/forecast_evaluations/`, 2007. [Online; accessed 30-May-2017].

[18] Rohit A Acharya et al. Improving major league baseball park factor estimates. *Journal of Quantitative Analysis in Sports*, 4(2):4, 2008.

[19] J.C. Bradbury. How do baseball players age? `http://www.baseballprospectus.com/article.php?articleid=9933`, 2010. [Online; accessed 25-May-2017].

[20] Neil Weinberg. Aging curve. `http://www.fangraphs.com/library/principles/aging-curve/`, 2017. [Online; accessed 25-May-2017].

[21] François Chollet et al. Keras. `https://github.com/fchollet/keras`, 2015.

[22] Martín Abadi et al. Tensorflow: Large-scale machine learning on heterogeneous systems, 2015.

[23] Michael P. Perrone and Leon N Cooper. When networks disagree: Ensemble methods for hybrid neural networks. *Published in R.J. Mammone, editor, Neural Networks for Speech and Image processing.*, 1992.

[24] F. Pedregosa et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[25] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.