14th IEEE International Conference on Industrial and Information System (ICIIS) 2019 (Accepted)

Convolutional Neural Network and Feature Encoding for Predicting the Outcome of Cricket Matches

Siyamalan Manivannan, Mogan Kausik
Department of Computer Science, University of Jaffna, Sri Lanka

***Abstract*— This paper proposes two novel approaches for predicting the outcome of cricket matches by modelling the team performance based on the performances of it's players in other matches. Our first approach is based on *feature encoding*, which assumes that there are different categories of players exist and models each team as a composition of player–category relationships. The second approach is based on a shallow Convolutional Neural Network (CNN) architecture, which contains only four layers to learn an end-to-end mapping between the performance of the players and the outcome of matches. Both of our approaches give considerable improvement over the baseline approaches we consider, and our shallow CNN architecture performs better than our proposed feature encoding-based approach. We show that the outcome of a match can be predicted with over 70% of accuracy.**

***Index Terms*—Convolutional Neural Networks, Feature Encoding, Winning team prediction in Cricket**

I. Introduction

Cricket is the most popular sport in Sri Lanka. It is played by two teams with eleven players in each side. Each team attempts to score runs, and team with the most runs will be the winner. International Cricket Council recognizes three forms of Cricket mainly based on the scheduled duration of the game. They are, Test Cricket, One Day International (ODI), and Twenty-20. ODI is considered the most popular form of Cricket, one of the reason is it allows day-night format compared to classical day-only form [1].

Outcome prediction of a particular match (which team will win in that particular match) even before the match is scheduled would be very useful in Cricket. If the prediction is based on the players' performance, a particular team could be composed against another team in such a way that the composition leads to victory.

There are many factors which can affect the outcome of a Cricket match, which includes who wins the toss, weather condition, condition of the ground, whether the match is held in a team's home country or not, etc. Other than these external factors, the main internal factor which determines the the winning team is the performance of its players. The performance of each player can be determined based on how well he played in other matches, e.g. his average batting score, how many wickets he captured, how many catches he caught, etc.

Corresponding author-email: siyam@univ.jfn.ac.lk

Applying machine learning techniques to the field of Cricket is recently gained popularity. There are a number approaches have been proposed in the literature for predicting the outcome of the Cricket matches [2]–[5], and predicting the performance of the players [6], [7]. Since we focus on the outcome prediction of the Cricket matches we limit our literature survey on the methods proposed for outcome prediction.

Various features and classifiers have been considered in the literature for predicting the outcome of a Cricket match. For example, factors such as home-field advantage, coin-toss result, bat-first or second, and day vs day-night effect were widely considered [1], [3], [5], [8]. Choudhury et al. [2] considered performances of the teams to determine the outcome; For each team the number of matches played, the number of matches won, the recent standing of the team, number of times a team reached the semi-final stages of a tournament, etc. have been considered as features to determine each team. However, a team's strength will differ based on the team composition (which players play for that team) and the players' performances, and the outcome of a match does not depend only on the factors such as home-field advantage, coin-toss result, bat-first or second, and day vs day-night effects. Differently from these approaches players' performances were considered to build the strength of the teams in [9] and [10] with various pre-determined parameters and rules. Various classifiers such as ANN [2], Random Forest [3], Support Vector Machines [3], Logistic regression [4], and Decision Tree [1], [5] also have been used for predicting the outcome of the matches.

Differently from these approaches, in this paper we propose two novel approaches, one is based on Feature Encoding, and the other is based on CNN, for predicting the outcome of a Cricket match even before the match is scheduled. Our proposed approaches are based on the past performances of the players in each team. In our first approach, we assume that there are different categories of players, and we model each team as a set of player-category relationships, and then train a linear SVM classifier on these relationships. Our second approach is based on a shallow (four layers) CNN, which is trained in an end-to-end manner to extract discriminative features and to learn a classifier to predict which team will win in a particular match. Although we apply our proposed approaches to ODI matches, we believe that our approaches are general, and hence, can be applied to other matches such

as Test Cricket and Twenty-20.

In the rest of this paper the proposed methods are explained in detail in Section II, experiments and results are reported in SectionIII, and finally, conclusion is given in Section IV.

II. Methodology

To best of our knowledge, neither feature encoding-based nor CNN-based approaches have been proposed for predicting the outcome of a Cricket match, although these approaches have been widely used in other domains [11]–[14]. Therefore, in this section we propose two novel approaches, one is based on feature encoding, and the other is based on CNN for Cricket outcome prediction. In the following, first, we explain our baseline approach, then our proposed approaches.

Each Cricket team is formed as a group of 11 players. Let's assume that the dataset comprised of N teams and M players in total from all the teams. ODI Cricket matches happen in different seasons. The team composition of different teams for each season may be different, and the performance of each player in different seasons also will be different. Let $\mathbf{x}_{i,t}^s \in \mathbb{R}^d$ represent the performance of a player P_i from the team T_t at the season s, and d represents is its dimension. $\mathbf{x}_{i,t}^s$ can be represented by the player P_i's batting, balling and fielding capability at the season s. For example, the batting capability can be represented as a combination of the average runs he got, how many balls he faced, etc. Therefore, the composition of a team at season s, T_t^s, can be represented by$\{\mathbf{x}_{\mathbf{1,t}}^{\mathbf{s}}, \mathbf{x}_{\mathbf{2,t}}^{\mathbf{s}}, \ldots, \mathbf{x}_{\mathbf{11,t}}^{\mathbf{s}}\}$.

The objective of this work is to find out which team will win in a particular match in a given season, when two opposing teams, their team compositions and the players' performances in the previous seasons are given as the input. Also note that to achieve this task we have access to a training dataset which contains the outcomes of the matches in the past seasons for different teams as well as the team composition of each team, and the performance of each player in each season. In this work we consider this problem as a binary classification problem as our dataset contains no *draw* outcomes. The outcome y_{ab} is represented as either 0 or 1, where $y_{ab} = 1$ represents that the team T_a^s wins when it plays against the team T_b^s in season s.

A. Baseline approach

The feature representation of the team T_t^s can be represented as a concatenation of the performances of all the players in that team as

$$T_t^s = \left[\mathbf{x}_{1,t}^s, \mathbf{x}_{2,t}^s, \ldots, \mathbf{x}_{11,t}^s\right] \tag{1}$$

Then a classifier such as SVM can be trained based on the following sets of representations

$$\left\{\left[T_a^s, T_b^s\right], y_{ab}\right\} \tag{2}$$

This representation, however, may capture the position information – the order of each player in each team, and hence, may reduce the classification performance. Therefore we average the performances of the players from each team to get the feature representation for that team. i.e.

$$T_t^s = \frac{1}{11}\sum_{i=1}^{11} \mathbf{x}_{i,t}^s \tag{3}$$

Since, this representation is not affected by the players' order it shows improved classification performance (Table IV) compared to the representation given in Eq. 1.

B. Feature Encoding for Cricket Outcome Prediction

The approaches explained in Section II-A do not utilize any hidden concepts in the input feature space, such as different categories of players; some players may be all rounders, some are good at batting, but not good at bowling and fielding, and so on. These hidden concepts/categories may capture useful information to the classifier to improve its performance.

One of the easiest way to include these category information in the feature representation of a team is to first apply a clustering (e.g. KMeans) on the feature representations of all the players from all the teams and seasons, and then for each team aggregate the information in each cluster based on the feature representations of its players. In detail, let $\{\mathbf{c}_q\}, q = 1, 2, \ldots, K$ ($\mathbf{c}_q \in \mathbb{R}^d$) represents the *dictionary* obtained by clustering the feature representations $\{\mathbf{x}_{i,t}^s\}$ from all the players, teams and seasons, where, K is the size of the dictionary. Each cluster in this dictionary will capture a hidden concept in the input feature space. The similarity between each of these concepts and each player (the *player-category* similarity) can be obtained as a weighted distance measure, $w_{iq}(\mathbf{x}_i - \mathbf{c}_q)$, where, w_{iq} determines the contribution of the player $\mathbf{x}_i$ to the concept $\mathbf{c}_q$ and it can be computed by Eqn. 4. (Note that we omit other subscripts of $\mathbf{x}_i$ for clarity.)

$$w_{iq} = \frac{\exp(-\beta\|\mathbf{x}_i - \mathbf{c}_q\|_2^2)}{\sum_j \exp(-\beta\|\mathbf{x}_i - \mathbf{c}_j\|_2^2)} \tag{4}$$

where β is a parameter, and it is set to $\beta = 0.1$ in our experiments.

By aggregating these distances for each concept with all the players playing in a team, the *team-category* relationship can be obtained as given below,

$$\mathbf{z}_q = \sum_{i=1}^{11} w_{iq}(\mathbf{x}_i - \mathbf{c}_q) \tag{5}$$

Now, the representation of a team will be obtained as the concatenation of these team-category relationships as given below.

$$T_t^s = [\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_K] \in \mathbb{R}^{d \times K} \tag{6}$$

In this way the final representation of a team can be obtained as an order-less representation of the performances of its players. The above approach is a soft-assignment version of the popular *VLAD* feature encoding scheme [12]. To train the classifier from these representations we use the same approach explained in Eqn. 2.

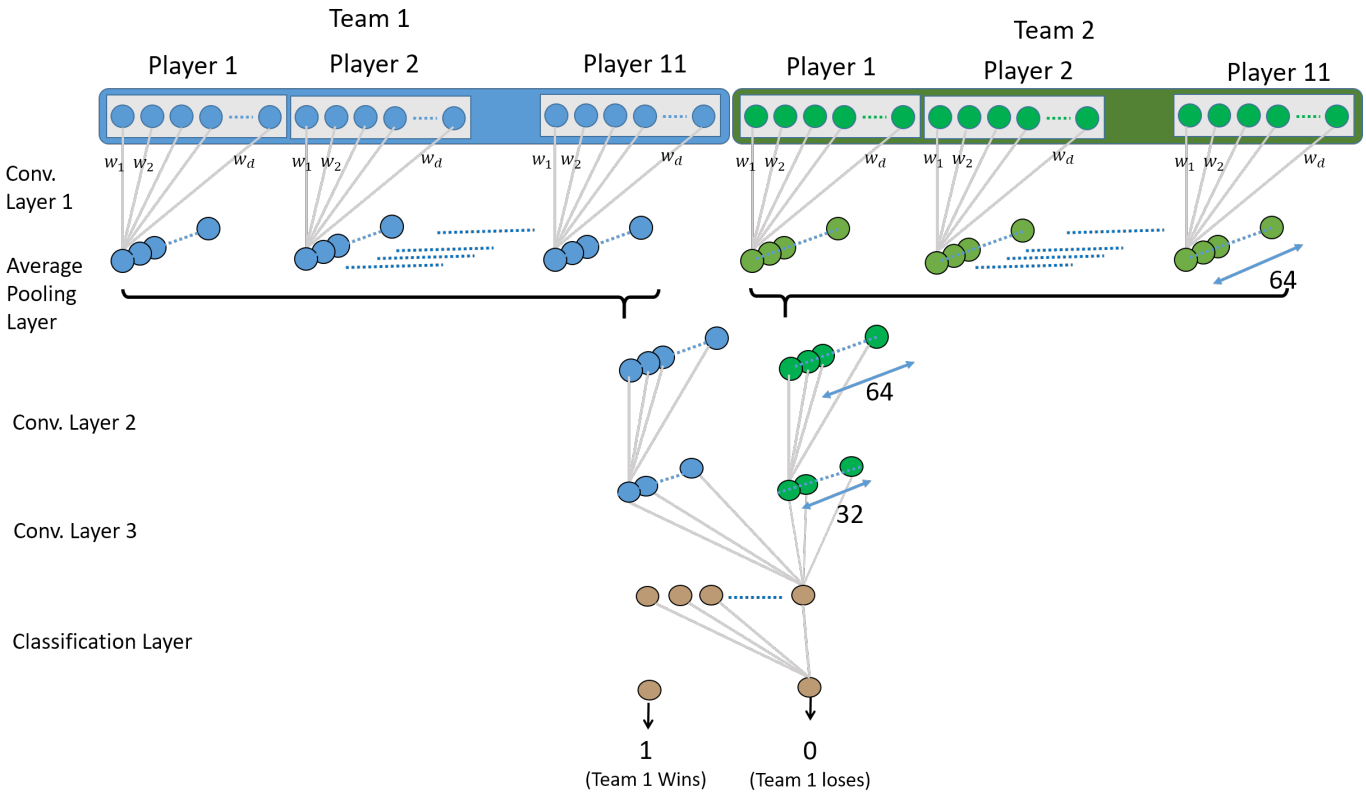

Fig. 1. The proposed CNN architecture. Some connections of the CNN in this figure are not shown to avoid clutter.

TABLE I
PROPOSED CNN ARCHITECTURE. THERE WAS NO PADDING USED IN ALL THE LAYERS.

	Conv. Layer 1	Avg. Pooling	Conv. Layer 2	Conv. Layer 3	Linear classification layer
Input dimension	$d \times 1 \times 1$	$22 \times 1 \times 64$	$2 \times 1 \times 64$	$2 \times 1 \times 32$	$16 \times 1 \times 1$
Output dimension	$22 \times 1 \times 64$	$1 \times 1 \times 64$	$2 \times 1 \times 32$	$1 \times 1 \times 16$	2
Filter size	$d \times 1 \times 1$	$22 \times 1 \times 1$	$1 \times 1 \times 64$	$2 \times 1 \times 32$	$16 \times 1 \times 1$
Stride	d	1	1	1	–
No. of filters	64	–	32	16	–

C. Convolutional Neural Networks for predicting the outcome of Cricket matches

In the approach proposed in Section II-B the hidden concepts are learned in an unsupervised way, therefore, the learned concepts may not be discriminative enough for classification. Also computing the feature representation of the teams and training the classifier based on these representations are performed in two independent stages. CNN, on the other hand, can learn the discriminative feature representations as well as the classifier in an end-to-end manner and shows state-of-the-art performance compared to these two stage approaches for various problems, e.g. image classification [14]. However, to best of our knowledge, the power of CNN has not yet been utilized for predicting the outcome of Cricket matches. Therefore, in this section we propose an approach for Cricket match outcome prediction based on CNN.

Note that one can simply train a Fully Connected Artificial Neural Network (ANN) by connecting all the input nodes to the hidden nodes with the representation given in Eqn. 1 and 2 as the input. However, this will increase the number of parameters to be learned, and more importantly, this will tend to capture the relative position information of the players, which is an undesirable property as discussed in Section II-A. We build our CNN architecture in such as way that it will not be affected by the ordering of the players in the teams. For this purpose we use the CNN property, *weight-sharing* – the same set of filters can be repeatedly applied to the input space to capture some hidden discriminative information. Weight sharing also reduces the number of parameters to be learned compared to fully connected ANNs.

Our proposed CNN architecture is illustrated in Figure 1. The main aim of this architecture is to first transfer the feature representation of each player from the original feature space to a discriminative space, and then aggregate discriminative

representation of each player to get the team representation in this space and then apply the classifier on these discriminative representations. These representations and the classifier will be learned in an end-to-end manner.

Our CNN architecture contains a convolution block, followed by an average pooling layer, followed by two more convolutional blocks and finally a linear classification layer (Table I). Each convolution block consists of a set of convolutions, followed by a batch normalization layer and which is followed by a rectified linear non-linearity. At the end of each convolutional layer we apply dropout [15] with a probability of 0.5 to reduce overfitting as we have limited training data. The input of this architecture is the concatenated representations of the features from the players of both teams as given in Eqn. 2. Our first convolutional block is aimed to capture player-level information. Therefore the kernel size is set to d (recall that the dimensionality of the feature representation of each player is d, and $d = 8$ in our case, refer Section III for detail) and the stride is also set to d. In this way the same sets of filters will be applied to each player separately. The average pooling layer aggregates the player information from each team individually to obtain two separate team-level feature vectors. The second convolutional block is applied to each of the team-level features to learn discriminative information from teams. The third convolutional block then gets the input features from both teams and transforms it into a single discriminative feature vector, on which the linear classifier is applied. To optimize this network end-to-end we use the *Focal loss* [16] instead of *Cross entropy loss* as focal loss reportedly performs better than cross entropy for classification problems in [16].

D. Adding Temporal Information of the Players Performances

We won't be able to get the performance of a player in season k until the end of that season. One simple way to approximate this is to use his performance in the immediate last season. However, his performance for season k may not just depend on his performance at season $k-1$. It also depends on all the previous seasons. Therefore, instead of using his statistics such as batting average in season $k-1$, number of wickets he captured in season $k-1$, and so on (refer Section III for the complete list of features) we weight all these statistics from all the previous seasons $k-1, k-2, \ldots, 1$ to approximate $\mathbf{x}_{it}^k$, i.e.

$$\mathbf{x}_{it}^k = \frac{1}{\sum_{s'=1}^{k-1} a_i^{s'}} \sum_{s=1}^{k-1} a_i^s \mathbf{x}_{it}^s \tag{7}$$

We apply higher weights to recent seasons in Eqn. 7. If a player did not play in a particular season that season will be omitted from calculation in Eqn. 7. In our experiments the weights a_i^s are approximated by a Gaussian filter with mean at k and standard deviation of 5.

III. Experiments

This section explains the dataset and the experimental setting we used, and reports the experiments and results.

A. Dataset

We crawled the website www.espncricinfo.com using our Python wrapper codes to collect the dataset. We were able to collect players information and the details of ODI matches from May 2013 to October 2017. Our dataset contains $2,581$ players and 474 ODI matches in total. Tables II and III report the number of matches in each season, and the features representing each player and the matches respectively.

B. Experimental Settings

To represent the performance of each player we considered six attributes from Table III (player name and the team name were excluded). Since different attributes of the players' data are in different ranges, first we applied a normalization so that each attribute in the players' data will have zero mean and unit variance. After this step the features representing each player in each season ($\mathbf{x}_{it}^s$) was computed as explained in Section II-D. For training, we use all the seasons in the training set to get the player performance, because, if we use only the previous seasons there won't be any way to get the players' performance at the very first seasons. But for testing, we use only the performances of the players from previous seasons to compute the performance at a particular season using Eqn. 7. This representation is then used by all the approaches used in this paper for representing $\mathbf{x}_{it}^k$.

The final feature vector (concatenated representation as in Eqn. 2) representing each match (extracted using the baseline and the proposed feature encoding approaches) are then normalized using the following normalization [11] before giving it to a linear SVM [17] classifier.

$$T_t^s = \frac{\text{sign}(T_t^s)|T_t^s|^{\frac{1}{2}}}{\|T_t^s\|_2^2} \tag{8}$$

Here $|T_t^s|^{\frac{1}{2}}$ applies the square root to each component of T_t^s.

The optimal value of the cost parameter C of the linear SVM was found by applying a 3-fold cross validation on the training set.

We apply a simple data augmentation for both training and testing. For each match we obtain two sets of instances, i.e. in addition to adding $\{\left[T_a^k, T_b^k\right], y_{ab}\}$ we also add $\{\left[T_b^k, T_a^k\right], y_{ba}\}$. Note that $y_{ba} = 1 - y_{ab}$.

We use *Accuracy* as the evaluation measure. For the proposed Feature Encoding-based and CNN-based approaches, we iterate each experiment 10 times and report the average and standard deviation of accuracy values over these iterations. At each iteration a separate dictionary is learned in the feature encoding approach. At each iteration CNN is initialized with small random values.

For the baseline and the Feature Encoding approaches after obtaining the team representations we extend these representations by adding two binary features: (1) whether that particular match is held in Team 1's home country or not, and (2) whether Team 1 batted first or not. For the CNN these two features are added at the player level, i.e. the feature representation of each player in a particular match is extended

TABLE II
DATASET: NUMBER OF MATCHES IN EACH SEASON.

Season	2013	2013/14	2014	2014/15	2015	2015/16	2016	2016/17	2017	2017/18
No. of matches	47	59	35	91	36	49	37	60	48	12

TABLE III
ATTRIBUTES OF THE PLAYERS AND THE MATCHES.

Attributes of each of the player in each season	name, team name, no. of matches played, no. of runs scored, batting average, no. of wickers captured, bowling average, no. of catches caught
Attributes of each of the matches	season, name of the Team 1, name of the Team 2, players' names in Team 1, players' names in Team 2, whether the match played in Team 1's home country, whether Team 1 batted first, winner of the match

by adding the following two features, (1) whether the match is played in the player's team's home country, and (2) the team the player plays batted first or not. Therefore the input to the CNN has a dimension of $8\,(\text{atrributes}) \times 11$ (players in each team) $\times 2\,(\text{ no. of teams})$.

For training the CNN we used the PyTorch library [18] on a NVidia Titan X Pascal GPU. We used a learning rate of 0.01, batch size of 32, and the number of *epoches* was set to 400. The testing accuracy at last ten epoches were averaged to get the final accuracy of each CNN trained.

C. Results and discussion

From Table IV we can see that when the concatenated representation of the players (Eqn. 1) is given as the input to the classifier, the classifier performs worse (Baseline 1 in Table IV) than using the averaged player performances (Eqn. 3) as the input (Baseline 2 in Table IV). This suggests that considering the players' ordering when building the feature representation of the teams is not a good idea, and hence, give the motivation to use a CNN architecture than a fully connected ANN.

The proposed Feature Encoding approach performs considerably better than our baseline approaches, for example, it gives $\sim 5\%$ improvement over the baseline approach when testing on seasons $\{2016/17, 2017, 2017/18\}$. The effect of dictionary size for the feature encoding approach for different test sets are given in Figure 2. The classification performance increases when increasing the dictionary size from 1 to 10 and then for larger dictionary sizes the performance reduces. We believe this is mainly because of the limited amount of data for training. When the dictionary size is set to a large value the classifier tends to overfit the training data.

Our proposed CNN architecture gives better performance than the baseline and the proposed feature encoding approaches as it can learn the features as well as the classifier in a discriminative end-to-end manner. We also experimented with an ensemble approach, where the predictions (probabilities) from 10 CNN models trained with different initializations are averaged to get the final prediction. This ensemble approach gives better performance than any other approaches we consider in Table IV. It is very interesting to see the good performance obtained by CNN, even with a limited training set as usually neural networks need large amount of data for training. We believe better performance than the one reported in this work can be obtained when a much larger dataset is used for training.

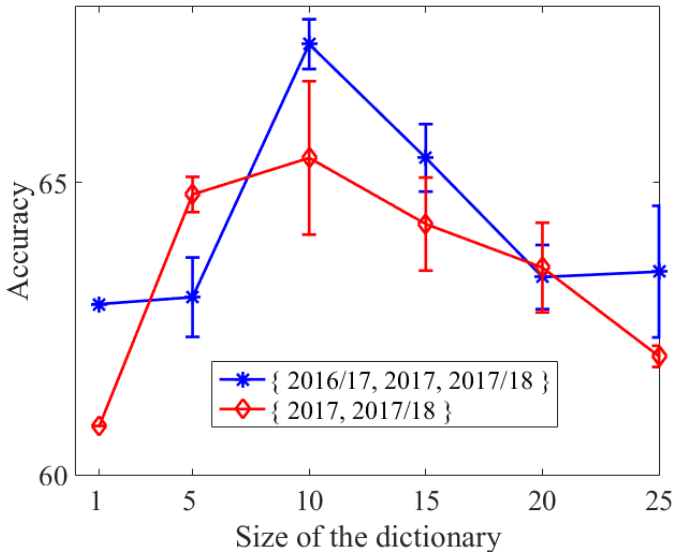

Fig. 2. Effect of dictionary size when tested on different seasons. Vertical bars shows the *standard errors*.

IV. CONCLUSION AND DISCUSSION

In this paper we presented two novel approaches for predicting the outcome of Cricket matches. Both of our approaches perform considerably better than the baseline approaches we considered. Usually deep CNN architectures need larger training data as they have large number of parameters to train. Therefore, in this work, we considered a shallow CNN architecture, which contains only four layers and we showed that it can be trained successfully with our limited training dataset. Compared to deep CNN architectures which usually contain millions of parameters, our architecture contains less than 4,000 parameters to train, and hence, it does not require a large dataset for training. To the best of our knowledge, we are the first group to apply CNN for Cricket outcome prediction. We showed that the outcome of a Cricket match can be predicted with an accuracy of over 70%. In this work, we

TABLE IV

COMPARISON OF DIFFERENT APPROACHES FOR CRICKET OUTCOME PREDICTION: BASELINE 1 AND BASELINE 2 REPRESENT THE CONCATENATED FEATURE REPRESENTATION GIVEN IN EQN. 1, AND THE AVERAGED REPRESENTATION GIVEN IN EQN. 3 RESPECTIVELY. FE REPRESENTS THE PROPOSED FEATURE ENCODING APPROACH WITH A FIXED DICTIONARY SIZE OF 10. N_{tr} AND N_{te} REPRESENT THE NUMBER OF TRAINING AND TESTING MATCHES (AUGMENTED) RESPECTIVELY.

Testing set (seasons)	N_{tr}	N_{te}	Baseline 1	Baseline 2	FE	CNN	CNN ensemble
$\{2016/17, 2017, 2017/18/\}$	634	314	56.68	62.42	67.36 ± 1.02	68.11 ± 1.07	$\mathbf{69.58 \pm 3.33}$
$\{2017, 2017/18\}$	708	240	62.02	62.50	65.42 ± 3.21	71.02 ± 3.33	$\mathbf{75.00}$

modelled each team based on the performance of the players play for that team. We considered features such as number of matches played in each season, number of runs scored, number wickets captured, and so on. However, we haven't considered some of the crucial factors such as the ranking of the players, player's strike rate, etc. These factors along with a larger dataset will be considered in our future work.

V. ACKNOWLEDGMENT

We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan X Pascal GPU used for this research and the University of Jaffna Research Grant, No: URG/2017/SEIT/05, for the support of a high- performance computer system.

REFERENCES

[1] K. P. Jayalath, "A machine learning approach to analyze odi cricket predictors," *Journal of Sports Analytics*, vol. 4, pp. 73–84, 2018.

[2] D. R. Choudhury, P.Bhargava, Reena, and S.Kain, "Use of artificial neural networks for predicting the outcome of cricket tournaments," *International Journal of Sports Science and Engineering*, vol. 1, no. 2, pp. 87–96, 2007.

[3] P. Neeraj and W. Hardik, "Applications of modern classification techniques to predict the outcome of ODI cricket," in *International Conference on Computational Science*, 2016, vol. 87, p. 55 – 60.

[4] N. Abhishek, P. Shivanee, N. Minakshee, and M. Sahil, "Winning prediction analysis in one-day-international (ODI) cricket using machine learning techniques," *International Journal of Emerging Technology and Computer Science*, vol. 3, no. 2, pp. 137–144, 2018.

[5] A. Kaluarachchi and S. V. Aparna, "CricAI: A classification based tool to predict the outcome in ODI cricket," in *International Conference on Information and Automation for Sustainability*, 2010, pp. 250–255.

[6] P. Kalpdrum and P.Niravkumar, "Increased prediction accuracy in the game of cricket using machine learning," *International Journal of Data Mining and Knowledge Management Process*, vol. 8, no. 2, pp. 19–36, 2018.

[7] P. Kalpdrum and P.Niravkumar, "Predicting players' performance in one day International cricket matches using machine learning," in *8th International Conference on Computer Science, Engineering and Applications*, 2018, p. 111 – 126.

[8] B. M. de Silva and T. B. Swartz, "Winning the coin toss and the home team advantage in one-day international cricket matches," *The New Zealand Statistician*, vol. 32, no. 2, pp. 16–22, 1997.

[9] S. Viswanadha, K. Sivalenka, M. G. Jhawar, and V. Pudi, "Dynamic winner prediction in twenty20 cricket: Based on relative team strengths," in *4th Workshop on Machine Learning and Data Mining for Sports Analytics, European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, 2017, pp. 41–50.

[10] M. G. Jhanwar and V. Pudi, "Predicting the outcome of ODI cricket matches: A team composition based approach," in *Proceedings of the Workshop on Machine Learning and Data Mining for Sports Analytics, European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, 2016.

[11] F. Perronnin, J. Sánchez, and T. Mensink, "Improving the fisher kernel for large-scale image classification," in *European Conference on Computer Vision*, 2010, pp. 143–156.

[12] H. Jegou, F.Perronnin, M. Douze, J. Snchez, and andC. Schmid P. Perez, "Aggregating local image descriptors into compact codes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 9, pp. 1704–1716, 2012.

[13] S. Manivannan, R.Wang, and E.Trucco, "Inter-cluster features for medical image classification," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI*, 2014, pp. 345–352.

[14] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[15] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.

[16] T.Y. Lin, P. Goyal, R. B. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *IEEE International Conference on Computer Vision*, 2017, pp. 2999–3007.

[17] R. Fan, K. Chang, C. Hsieh, X. Wang, and C. Lin, "LIBLINEAR: A library for large linear classification," *Journal of Machine Learning Research*, vol. 9, pp. 1871–1874, 2008.

[18] A Paszke, S Gross, S Chintala, G Chanan, E Yang, Z DeVito, Z Lin, A Desmaison, L Antiga, and A Lerer, "Automatic differentiation in pytorch," in *Neural Information Processing Systems - Workshop*, 2017.