Application of Behavioral Cloning on FIFA

Puranjay Rajvanshi
MS Computer Science
University of Southern California
Los Angeles, USA
pranjvans@usc.edu

Sangeeth Koratten
MS Computer Science
University of Southern California
Los Angeles, USA
koratten@usc.edu

Shelly Mehta
MS Computer Science
University of Southern California
Los Angeles, USA
shellyme@usc.edu

Rishika Verma
MS Computer Science
University of Southern California
Los Angeles, USA
rverma@usc.edu

Vandit Maheshwari
MS Computer Science
University of Southern California
Los Angeles, USA
vanditma@usc.edu

Aviral Upadhyay
MS Computer Science
University of Southern California
Los Angeles, USA
aviralup@usc.edu

Abstract— **As the world of machine learning grows, bots are seeming to perform and execute all the actions that a human can do. With numerous instances of the former already stated in the world, another one would be creation of an artificial intelligent bot that mimics the actions and the controls that humans perform via the game controller for the game FIFA to best of its ability. The work focused to clearly distinguish various items present on the field like the ball, other players and the goalpost using convolutional neural networks (CNN) as well as choosing the most suitable option of movement from the given options on the game controller via Long Short-Term Memory [1] (LSTM).**

I. Introduction

Artificial intelligence is defined as a study of rational agents. A rational agent is one which is capable of doing expected actions to maximize its performance measure on the basis of percept sequence and its built-in knowledge base. In this circumstance, the built-in knowledge base would be the colossal set of training data with snippets of the field with players and the ball in different locations and positions. The percept sequence in this case are the actions performed by the humans (in layman terms, the buttons pressed on the remote controller) to play the game of FIFA virtually.

FIFA, short for Fédération Internationale de Football Association was founded in 1904 to oversee international competition among the national associations of 8 countries in the beginning, and now contains 6 confederations and 211 national associations. Played by approximately 250 million players in over 200 countries, makes it the world's most popular sport. The game is played on a rectangular field called a pitch with a goal at each end. Each team comprises of 11 players (excluding substitutes), one of which is a goalkeeper. Players are not allowed to touch the ball with hands or arms while it is in play except the goalkeeper. Other players mainly use their feet to strike or pass the ball hence the name Football. The objective of the game is to outscore your opponents by moving the ball past the opposing goal line. These guidelines are accompanied by a set of rules and restriction making it more than a game of athletics and strength.

II. Literature Survey

There has been a lot of research on rule-based machine learning in which the bot is given a set of rules or actions to learn from. The model simply trains itself by imitating a human playing the game, thereby introducing the concept of behavioral cloning [2]. Behavioral cloning is a method by which human sub cognitive skills can be captured and reproduced in a computer program. While in action, the decisions made by human in different situations are recorded and then presented to these as input to the learning bot. For object detection in the environment, computer vision technology is used. Object detection draws bounding boxes around detected items thus allowing locating those objects [3][4]. This technique is vital to track the physical movement of the different articles in the game and to track other moving objects such as the ball while simulating the soccer game. An example-based learning approach is used where a model of an object class is derived implicitly from a set of training images. This ensures that the model utilizes the algorithm which is specialized to a specific domain related to the surroundings. The field of object detection has grown throughout the years and paved its way to single shot detection [5]. This technique uses a single shot to detect multiple objects present in an image using multibox. At the time of prediction, the network generates scores for the presence of each object category in each default box and produces adjustments to the box to better match the object shape. Since it eliminates proposal generation and subsequent pixel or feature resampling stages and encapsulates all computation in a single network, it is easy to train and integrate into the system. Also, convolutional neural network is a well-known machine learning algorithm for image recognition and image classification. Though this concept has been invented some time back now, it is still highly used for object detection. The model takes each input image and passes it through a series of convolutional layers with filters(kernels), pooling and fully connected layers and then applies SoftMax function to classify an object with some probabilistic values between 0 and 1. There are multiple types of CNN used for a variety of reasons. MobileNet are convolutional neural network architectures whose number of trainable parameters can be controlled by two hyperparameters – width and resolution multiplier. Although using MobileNet instead of single shot detection decreases average detection accuracy, it provides more speed relatively. Using MobileNet architecture is key while designing a game playing intelligent agent keeping in mind the constrained hardware settings. Deep reinforcement learning has ended up being extremely fruitful in learning human-level control policies in a wide assortment of tasks, for example, object recognition with visual attention [6], high-dimensional robot control [7] and solving physics-based control problems [8] are demonstrated to be powerful in playing Atari 2600 games [9] and all the more as of late, in crushing elite Go players [10].

Nonetheless, there is a constraint in the entirety of the above applications in their supposition of having the full knowledge of the current state of the environment, which is normally not valid in real-world situations. On account of partially observable states, the learning agent needs to recall past states to choose optimal actions. As of late, there have been endeavors to deal with partially observable states in deep reinforcement learning by presenting recurrency in Deep Q-networks. For instance, [11] use a deep recurrent neural network, especially a Long-ShortTerm-Memory (LSTM) Network, to learn the Q-function to play Atari 2600 games. [12] consider a multi-agent situation where they utilize deep distributed recurrent neural networks to communicate between different agents in order to unravel puzzles. The utilization of recurrent neural networks is powerful in situations with partially observable states because of its capacity to recollect data for a subjectively long measure of time. Once the observable environment has been studied, the output can be generated using recurrent neural networks. The next action of the bot needs data from different time steps instead of relying on a particular time instance. RNN like long short-term memory networks have shown to provide state-of-the-art results on time series data classification. A study has shown remarkable accuracy scores with a full detection framework involved of many Bi-directional Long Short Term Memory (Bi-LSTMs) neural networks [13]. However, in addition to extensive data preprocessing, the computational overhead involved in training multiple supervised and unsupervised models is critical. A significant amount of assets is needed to execute and keep up such a framework, which makes such frameworks less scalable for real time execution. Feature vectors obtained from the CNN network can be passed to multiple LSTM networks that run in parallel to obtain different types of output. A player has to control the movement of the player along with the action to be taken on the ball. So, one network outputs the next movement of the player at a given instance while the other outputs the action on the ball by providing the feature vectors of the environment in the previous time frames.

III. Methodology

To achieve our end result of the project, the tasks were divided into different segments. The first part for recognition of various types of objects i.e., object detection and feature extraction of players, ball and goal post and the second part for deciding the most appropriate in-game action. Convolutional neural networks were used for identifying objects in the environment and long short-term memory networks were used for deciding the next keystrokes to help the player move and pass or shoot the ball.

A. Object Detection and Feature Extraction

1) Data Collection and Preprocessing:

For the purpose of the project, the model is trained on different images of the game environment. The desktop version of FIFA 18 was downloaded to set up the environment. Further, various **time spaced screenshots** of a human playing the game are collected and stored in .jpg formats. To record **in-person actions**, **key-press simulation** is used to communicate the output that needs to be taken. The image is cropped to obtain desired frame of the game window.

2) Object Detection:

The **Object Detection API** of TensorFlow is used to get the **bounding boxes** for each object on all the images. The **labelImg** library supported by Tensorflow is used for the above task. Installation is done using PIP. The labelImg library creates a separate window which helps to manually create bounding boxes for object and also helps in annotation. This tool helps us to create training images.

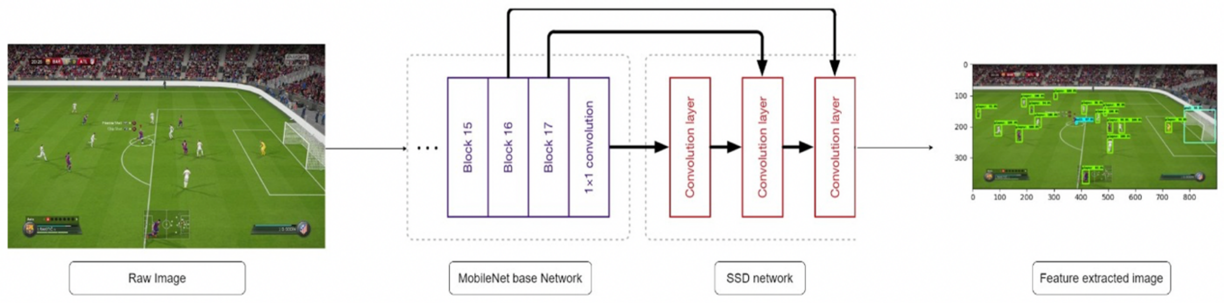

Fig. 1. Pipeline for Object Detection

After the annotation is completed, for each training image, a corresponding .xml file is created which holds the object tag along with its xmax, ymax, xmin and ymin coordinates. The .xml files are then converted into .csv format files which are in turn used to generate **tfrecord** files. These tfrecords are then passed as input to the CNN model.

3) Model and Training:

Transfer learning is performed for the first part of the project. The CNN model used is the **MobileNet SSD** network(Fig. 1). MobileNet is a pre-trained CNN model which is part of the **Object Detection API.** The output layer is updated to accommodate the output for the given training image dataset. After training the model on our processed image dataset, we obtain a **128-bit feature vector** for each image as a result from the network. The **hyperparameters** used for the models was as follows :

Loss:

classification_loss: weighted_sigmoid

localization_loss: weighted_smooth_l1

Batch_size: 24

Optimizer:

rms_prop_optimizer:

exponential_decay_learning_rate

initial_learning_rate: 0.004

decay_steps: 800720

decay_factor: 0.95

momentum_optimizer_value: 0.9

decay: 0.9, epsilon: 1.0

Epochs: 5000

Data Augmentation: Random Horizontal Flip, SSD Random Crop

Training this model took about 24 hours to complete.

B. Action Selection on time-series data

To decide the next key stroke for the player's action on the ball and movement, previous instances of the game window are required since the game is being played in real time. Two parallel LSTMs are used to output these actions.

1) Data Collection and Labelling:

Data collection for the LSTM networks is done by collecting **128-bit feature** vector for the images of the 10 previous time-steps for a given instance and recording the key stroke i.e. "Movement" and "Action" as the class labels for the two LSTMs respectively. The CNN model is used to obtain the feature vectors for the input to the LSTM networks. The data is **split** as **70%-30%** into training and test datasets.

2) Model and Training:

Two parallel running LSTM networks are fed with **128-bit x 10 feature vectors** for the time-series data. Each LSTM model is defined as mentioned in the **Fig 2**. 10 feature vectors are passed through **two serially connected LSTM layers** of **256 units** each. The output from the second LSTM layer is then passed to a **fully-connected layer**. The fully-connected layer in each LSTM network has **5 outputs**. The network for "**Movement**" has **left, right, up, down and nothing** as outputs, while network for "**Action on ball**" has **pass, shoot, cross, through and nothing** as output. The **hyperparameters** used for both the models was as follows : **optimizer – SGD, loss – categorical_crossentropy, epochs – 300, batch size – 32, learning rate – 0.001** . Training the two LSTM networks parallelly took 2.5 days to complete.

3) Oversampling Techniques:

Since for most of the game, there is no action taken on the ball, the training dataset had about **60%-70%** of the "Action on ball" to belong to "**nothing**" class. So, the dataset had to be **oversampled** for other actions like **pass, shoot**, **cross** and **through**. To oversample the data for **minority class**, the data for all these classes was copied multiple times into the training dataset. Also, more samples with higher frequency of actions like pass, cross, shoot and through were added to the dataset. This ensured that the player not only runs around with the ball but also takes some action on it.

IV. Results

The entire game of FIFA was translated to an automated game on the basis of **behavioral cloning**. The bot tried to imitate human actions taken during the game for particular instances. Time-series screenshots of the game window

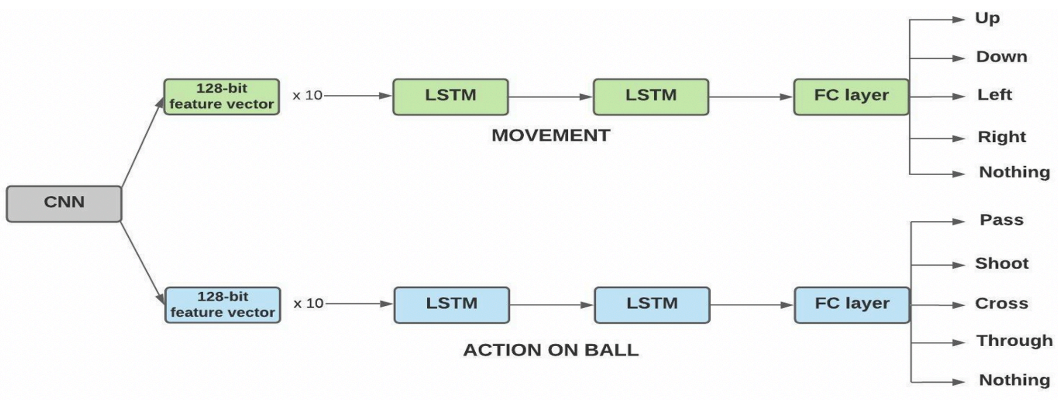

Fig 2: Two parallel LSTM models for action selection

helped take accurate actions and bounding boxes helped identify objects in the images.

Additional learning: In addition to taking corners, penalties, free-kicks, throw-ins, the bot also learned to **quick-skip** the celebration sequences.

Fig 3 depicts the localization loss between the predicted box and the ground-truth box parameters.

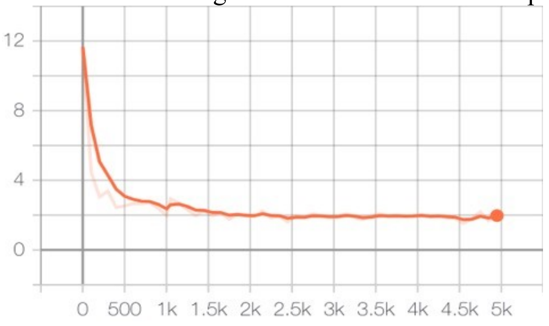

Fig 3: Localization loss between predicted and ground-truth box

Although, training the agent took a lot of time, the bot played exceptionally well at the initial stages. The bot was run at various difficulty levels to measure the efficiency. At beginner level, the agent played exceptionally well but was unable to perform too well at the legendary level. The results for the LSTM models were stored as graphs before and after oversampling the dataset. But the **effectiveness of the model was observed while actually playing the bot**. There was vast improvement in the model performance for both movement and action.

The model performance for both the LSTMs was recorded in graphs before and after oversampling as shown in **Fig 4**.

V. Challenges

Understanding Object Detection APIs for creating bounding boxes around objects still remains a difficult task. The complexity of installation and then integration with the code base with these tools is higher and sometimes challenging. Object Detection algorithms are still very sensitive to good frames per second (fps). Thus, on a normal machine, object detection for videos is a sphere which could require some improvement.

Training the LSTM networks to achieve better accuracy was time-consuming as different architectures with different number of layers were tried and a lot of hypertuning of parameters had to be performed. Also, the training time of the CNN model and LSTM networks proved to be another substantial hurdle for the project. Overcoming class imbalance was another hinderance for which a lot of oversampling has to be done.

VI. Future Work

Currently, the model is only trained to attack from the left side and playing from the right side would result in own goals. To overcome this, another model can be built to train the model to attack from the right side. This could be used to play two agents against each other. Also, the bot currently is unable to show exemplary results at legendary level, so to attain better performance in real-time against an advance player, more processing power can be used along with deeper

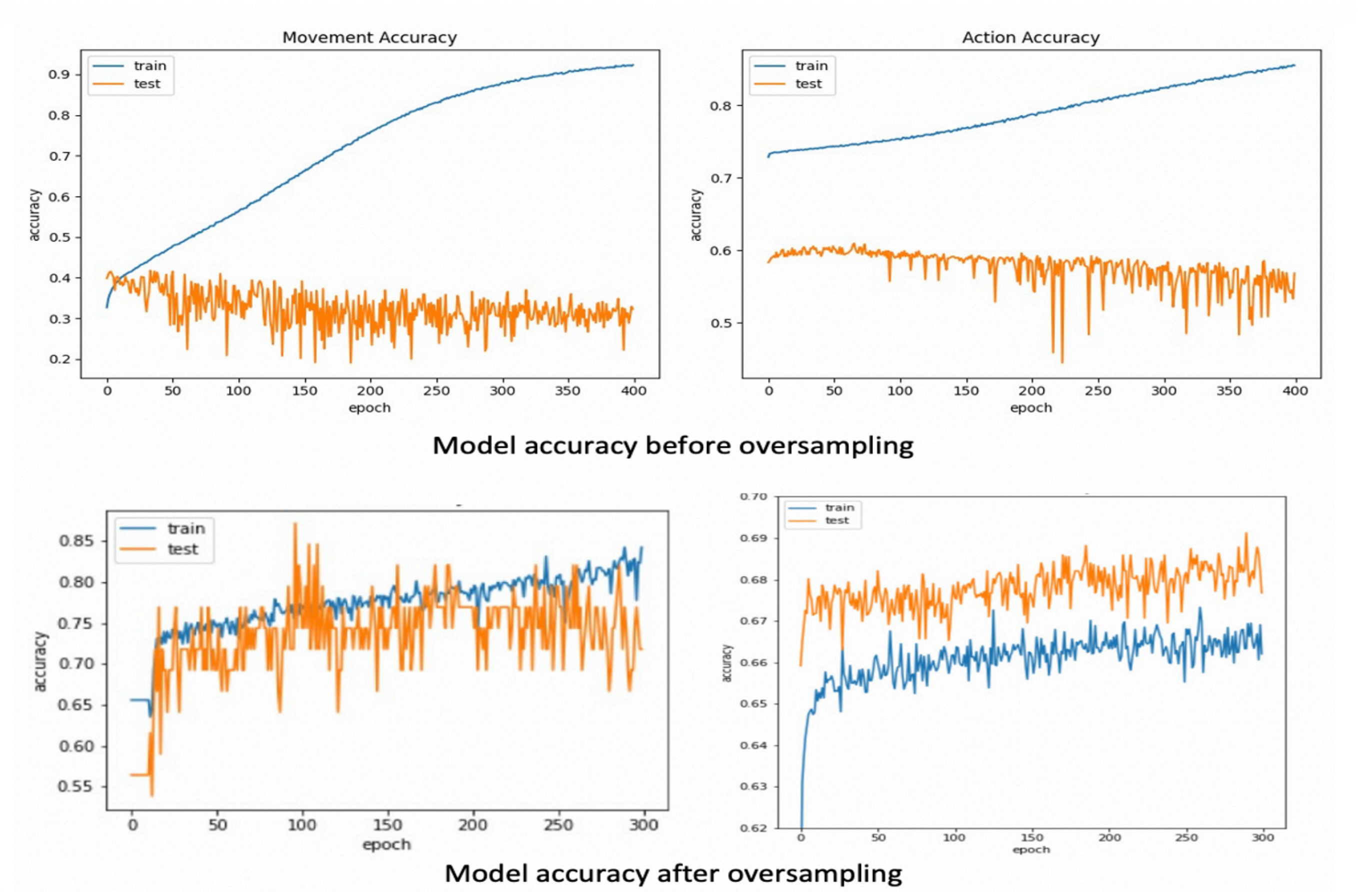

Fig 4: LSTM model performance for movement and action

models. This will not only result in a more efficient bot but will reduce the training time as well. Another improvement that can be made in the future is to use the minimap as an input to figure out the position of other team players. This can help in deciding which player the ball has to be passed to.

REFERENCES

[1] Danilo Sorano1, Fabio Carrara2, Paolo Cintia,1Fabrizio Falchi2, and Luca Pappalardo2 ; Automatic Pass Annotation from Soccer Video Streams Based on Object Detection and LSTM

[2] Robert G. Abbott; Behavioural Cloning for Simulator Validation

[3] Juarez Monteiro_1, Nathan Gavenskiy1, Roger Granada_, Felipe Meneguzziz and Rodrigo Barrosz ; Augmented Behavioral Cloning from Observation

[4] Okihisa UTSUMI, Koichi MIURA, Ichiro IDE, Shuichi SAKAI, Hidehiko TANAKA; AN OBJECT DETECTION METHOD FOR DESCRIBING SOCCER GAMES FROM VIDEO

[5] Wei Liu1, Dragomir Anguelov2, Dumitru Erhan3, Christian Szegedy3, Scott Reed4, Cheng-Yang Fu1, Alexander C. Berg1; SSD: Single Shot MultiBox Detection

[6] Ba, J.; Mnih, V.; and Kavukcuoglu, K. 2014. Multiple object recognition with visual attention. arXiv preprint arXiv:1412.7755

[7] Levine, S.; Finn, C.; Darrell, T.; and Abbeel, P. 2016. End-to-end training of deep visuomotor policies. Journal of Machine Learning Research 17(39):1– 40.

[8] Heess, N.; Wayne, G.; Silver, D.; Lillicrap, T.; Erez, T.; and Tassa, Y. 2015. Learning continuous control policies by stochastic value gradients. In Advances in Neural Information Processing Systems, 2944–2952

[9] Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; and Riedmiller, M. 2013. Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602

[10] Silver, D.; Huang, A.; Maddison, C. J.; Guez, A.; Sifre, L.; Van Den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; et al. 2016. Mastering the game of go with deep neural networks and tree search. Nature 529(7587):484–489.

[11] Hausknecht, M., and Stone, P. 2015. Deep recurrent q-learning for partially observable mdps. arXiv preprint arXiv:1507.06527

[12] Foerster, J. N.; Assael, Y. M.; de Freitas, N.; and Whiteson, S. 2016. Learning to communicate to solve riddles with deep distributed recurrent q-networks. arXiv preprint arXiv:1602.02672.

[13] J. Tao, J. Xu, L. Gong, Y. Li, C. Fan, and Z. Zhao, ''NGUARD: A game bot detection framework for NetEase MMORPGs,'' in Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, Jul. 2018, pp. 811–820.