CS 5150: Game Artificial Intelligence
FIFA 18 Free kick Bot

Introduction

FIFA 18 is a football simulation video game that simulates soccer. Free kicks are an essential part of the game when the opposite team does a foul outside their own box area and its in shooting range. The free kick is 1v1 kind of effort where the kicker tries to put the ball inside the goal and can have multiple players building a wall-like structure to block the shot. Also, the keeper's job is to stop the kicker from doing so. It performs various actions (shooting high, shooting low, moving the player left and moving the player on the right)
It also simulates restarting the drill after the drill is over for continuous interrupted training.We proposed to apply deep Q learning, which provides right directions and power level to the kick the ball to achieve points in the skills training module of the game. The objective should be to maximize the points gained. We also proposed to attempt a tensorflow based neural network using actor critic approach to achieve the same task.

We have implemented a deep Q learning network which learns the conditions and state from the screen shots we grab screen and pass it to VGG16 a famous deep learning network which comes with keras.applications to extract features out of the screenshot that we capture. We pass this features to a dense neural network with 3 layers to generate Q value for all four of our actions which are shoot_high, shoot_low, left arrow and right arrow. We select the action with highest Q value and simulate that key press. We then grab screen and run OCR to extract the reward of that shot taken by DQN. We are currently using FIFA generated points.

In fifa 18, we give rewards based on player's freekick score after every shoot. Ideally, it gives 1000 or more than 1000 when it hits one of the four targets.It gives approximate 800 - 999 if it hits near target.It gives 500- 700 if player just scored nowhere near target but inside the goal.It gives 200 if the shot hits the goal post and 0 points if its off target.

We trained it for various amounts of time and were getting somewhat okay results with time. We also made sure that this results are just not because of random selection and decayed the probability with which the controller selects the actions. As time passes on it starts using model generated actions and reduces dependence on random actions. We also used feature extraction using Deep neural network as opposed to passing the whole image to the Q_function neural network.

We experimented on LSTMs as opposed to this memory consuming Experience Replay logic that is generally used in Deep Q Learning, but failed in implementing it properly, so we switched back to experience replay for further experiments.

We also implemented tensor flow based neural network using actor critic approach. We trained for a various amount of time.

Links to various training models are attached in zip file under Videos Links folder.

Topics

Milestone 1 - Able to control character & read game state using the emulator - everything necessary for both actor and critic to work.
Milestone 2 - Your Neural Network code is done, including the actor-critic interactions.
Final - Experiments showing comparison to the existing system or performance under different parameter settings.

Team

Manush Patel - patel.man@husky.neu.edu
Designed Actor Critic Architecture, Modified action space, Designed various Heuristic Patterns

Dharmish Shah - shah.dharm@husky.neu.edu
Designed Deep Q learning neural Network, Designed Reward Function, Modified the methods to extract rewards from the game

Instructions

Prerequisites -
FIFA 18
Python
Libraries - tensorflow, keras, cv2, tesseract, pydot

If you don't have FIFA 18, please do contact us so we can show we can show the demo in person.

Steps to run -

1. Download the zip file submitted.
2. Extract it in a folder
3. Install the necessary python dependencies using requirements.txt
4. Run Freekick.py file using python
5. Wait for it to load necessary files. Training is paused initially.
6. Start FIFA 18 and play the skills game to train in free kick mode.
7. After its done, press P in command prompt to run the simulation.
8. DONE!!! Wait for the magic to happen.

Systems

Deep Q Learning: The first model that we implemented was Deep Q learning technique and for this we implemented a deep learning based model. We chose to implement it our own way by following the original paper that was written and we tried to pass the feature extracted by VGG model which takes in the screen shot of the screen and processes it to provide a feature matrix which we feed to a neural network of 3 layers which produces the output Q values for all the possible actions from which we choose the action with maximum Q value and perform it in the game by simulating the key press on the keyboard for that action. We then observe the game again to capture the in game reward and pass it to our own reward system which reduces it to a fixed reward value according to the reward policies described in the experiment section. We continuously store every state,action,reward for the state and next state after that action to our experience replay buffer and load random samples from this 1000 experience buffer to train the model. We use this random sample to train the model. We have also tried last 4 experience based training and it did not produce proper results so we decided to stick with our random sampling approach. We also experimented with numbers of layers in the model and selected this configuration to train our model. We also checkpoint our evaluations and model weights after every episode so that we can continue at any time.

Actor critic approach

Actor critic model is generally favored for continuous action space and we wanted to experiment it with our discrete space which can be converted to a continuous space if we opened up our shooting time as an action. We tried discrete action space based actor critic approach which is slow but we were able to get the gist of the training process of the model, which according to us should require a lot of time. We have designed an actor neural network with following layers:

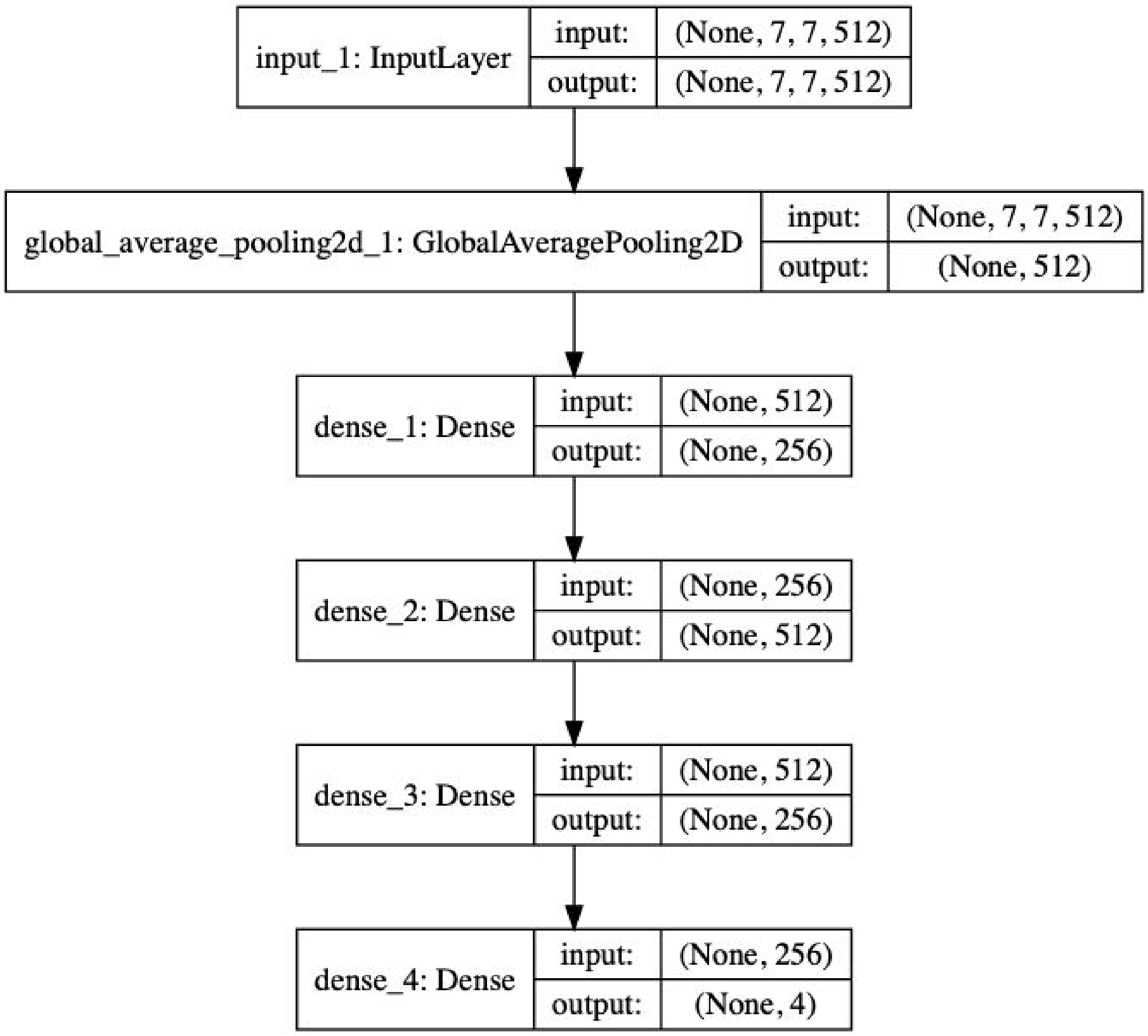
1 -> VGG16 based feature extractor
2 -> GlobalAveragePooling2D layer
3 -> 3 Dense layers to with last one with 4 nodes to predict output

And a critic network with following architecture:

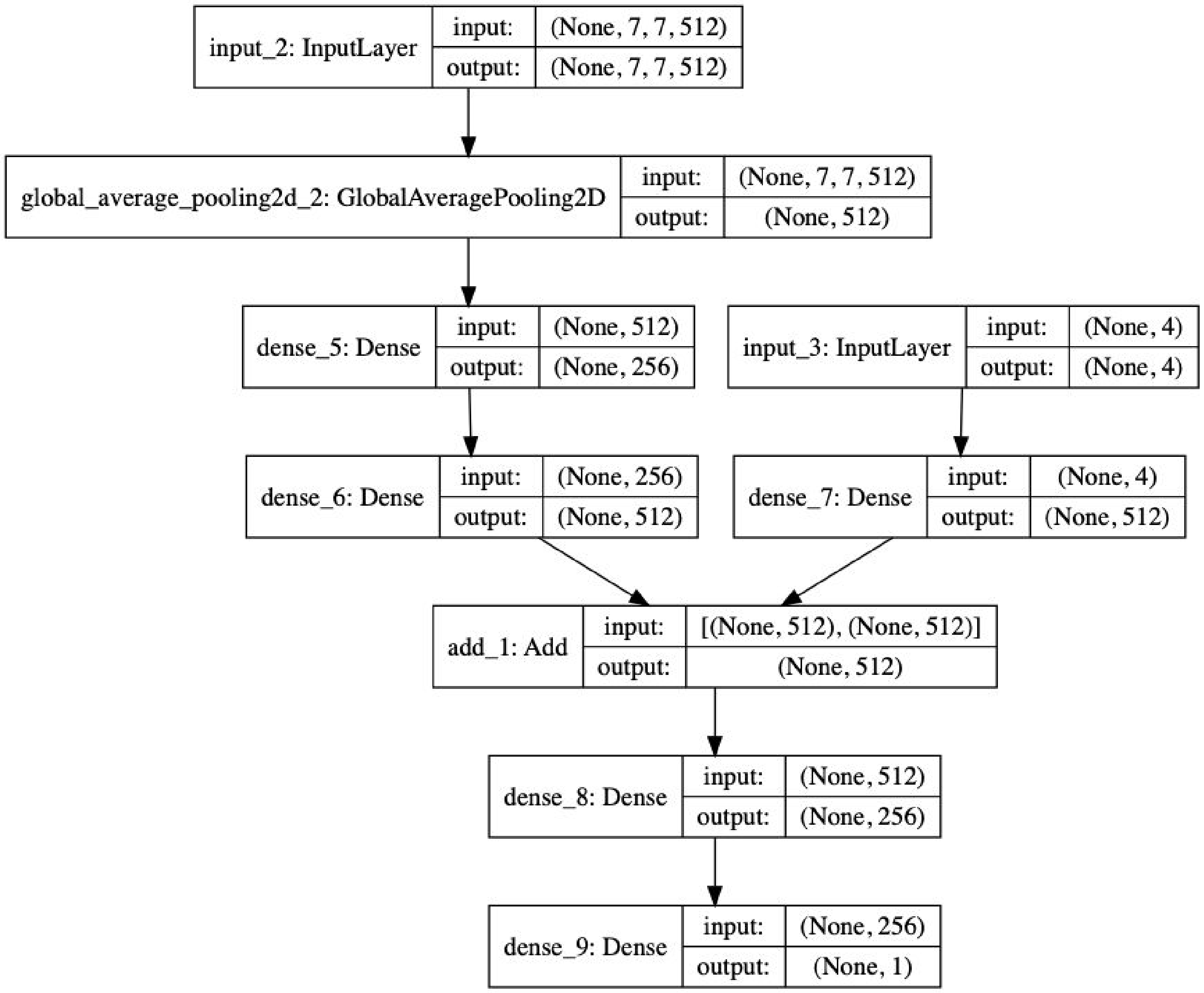
1-> Action Input layer with 4 nodes followed by 1 Dense layer
2-> State Input layer which takes VGG16 feature output as input, followed by 3 Dense Layers.
3-> A merge layer that simply adds two of these networks and gives a single layer output.

The critic and actor both the models take state which is extracted matrix by VGG16 and trains on that. Actor outputs same q values as the deep Q network's neural network, but the major difference between both the model is actor's action output is fed to the critic network along with current state to update the actor model even better.

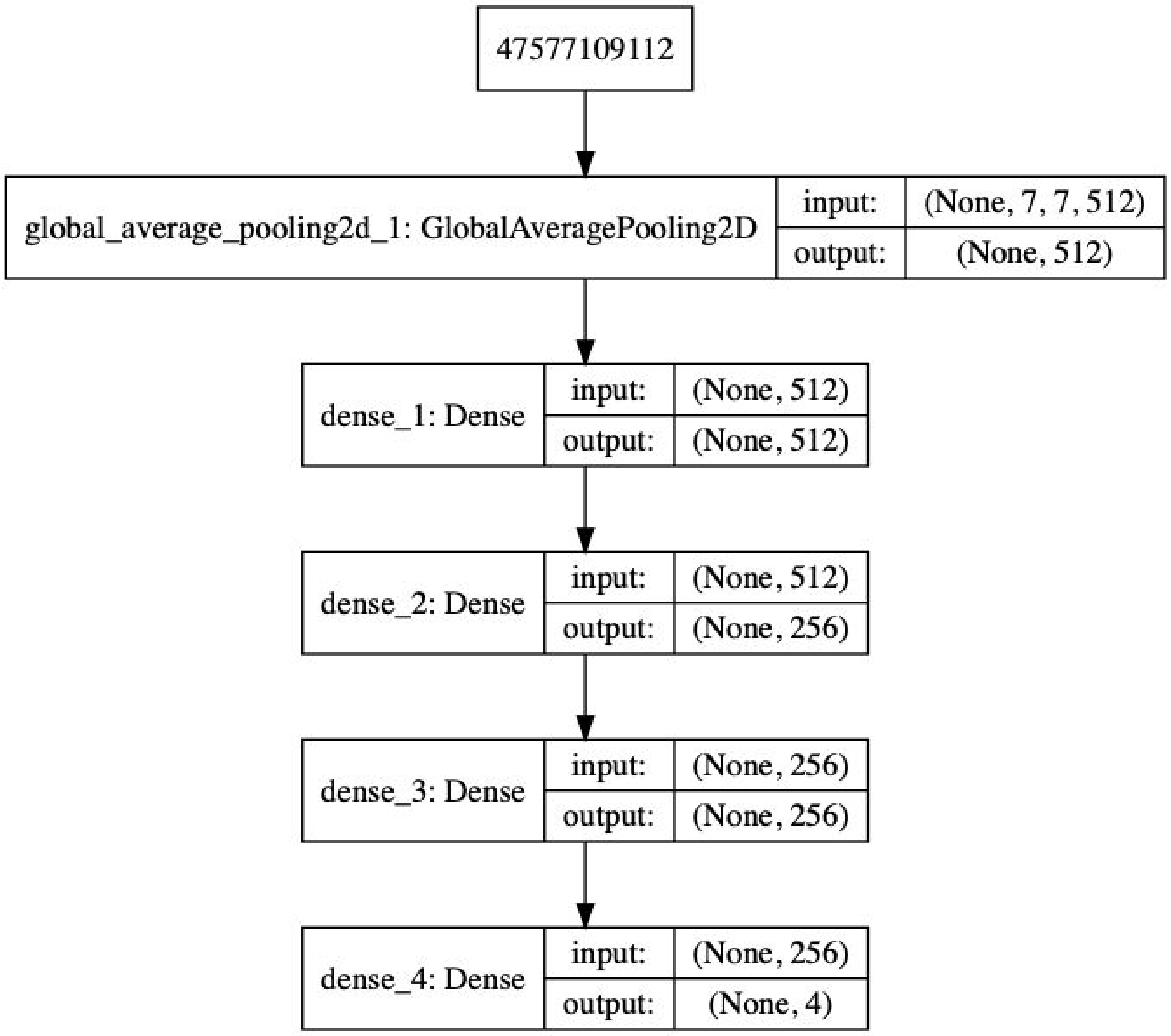
Model Architectures



Actor Model



Critic Model



Deep Q Model

Failures and Setbacks

We first had to deal with the situation where EA sports does not provide any kind of support for developers and we had to deal with actions,rewards and states manually with help of screen grabs,which was pretty easy when you have python libraries to do the task for you. Traditional deep Q learning models reshape the output to a two dimensional grid and pass it to a deep neural network, which we in our case would have caused problems because the state size when we grabbed the screen was equal to the whole screen of the pc that it was running on and we had to crop the image to pass it to the feature extraction neural network. We tried implementing an actor critic model with two neural network by investing significant amount of time in designing the model and combining the training of both the models and predict an action output . This does not bode well with the environment that we are training with because FIFA free kicks are time limited and if you do not take a shot within several time it automatically takes a fluke shot for you which generates 0 score for the model and this caused a long delay in training the model, considering the PC that we were running our model and FIFA both had only 8 GB of RAM and 2 GB of GPU, so even after trying with GPU based tensorflow we could hardly take two shots in 1 minute drill with 5 to 6 right and left movements so in general per minute it could make only 7 to 8 moves and which delayed the training for the model. This is the major reason why we could not experiment with the LSTM usage instead of experience replay in any of our models. We really wanted to see what LSTM did to store the previous sequences passed to it by training 1 image at a time and training the model without any experience replay. We however tried one off experiment with it and it did not go well as it took 1 shot per minute which indicated higher training time than existing times.

We have also attached images of model summary on our github at:

Extraction methods

1) We tried with various feature extraction method such as simple convolutional neural network and a famous deep neural network called VGG16 which is specifically designed and trained on millions of images to classify images, we used a part of this popular network to extract features from the Image we grabbed as a state from the FIFA gameplay.
2) Cropping the image using cv2 for calculating the rewards - We had to find amount of rewards that we wanted to give the value of score that is projected on the screen, we cropped the part of the image by experimenting OCR which we use from tesseract APIs to get text from the screen to increase the precision of score gathering mechanism and reduce the number of actions with 0 rewards, still due to OCR failures we sometimes get unnecessary outputs and in that case we give model a minor positive or negative reward based on random choice.

Experiments and Observations

We ran various experiments on our two models. Parameters which were considered are taking random actions, no random actions, using model prediction, different reward tweaking, experience replay batch sizes.

Terminology -

1) Baseline Model - No saved data, training from scratch
2) Saved and Loaded Model - experiences are saved based on states and actions.These are then used as a reference for future training.
3) Rewards phases -
   S - score awarded after every shoot

Reward Phase	**S > 1000**	**500 < S < 1000**	**0 < S < 500**	**S == 0**
Reward Phase 1	1	1	1	-1
Reward Phase 2	5	2.5	1	-1

1) No random movements with baseline model and reward phase 1
    a) Sometimes always goes in right side, or always left. Bot doesn't even shoot sometimes
    b) After some training it does shoot.
    c) Better free kicks when a wall is present.
    d) After failing in shoot low when a wall is present, it takes high shoot after training
2) Random movements with baseline model and reward phase 1
    a) Does random left right movement
    b) After failing in shoot low when a wall is present, it again takes low shot after training.
3) Random movements with model prediction with Epsilon value with baseline model with reward phase 1
    a) To avoid continuous left or right actions, it shoots low or high randomly as an action, exploring new actions.
    b) Sometimes always goes in right side, or always left. Bot doesn't even shoot sometimes
    c) After some training it does shoot.
4) Rewards are changed with no random movements with reward phase 2
    a) Too many extreme left or extreme right movements and no shoot sometimes.(majorly extreme left)
    b) Repetitive actions.Should add negative rewards for repetitive actions.
5) Random and Predicted movements based on saved and loaded model
    a) Performs much better than previous experiments
    b) Scores are improved over time
    c) No unnecessary actions are performed (continuous left or right actions)
6) Random and predicted movements based on baseline model using actor-critic approach
    a) Takes too much time to shoot the ball
    b) Average number of shots taken per drill is 2-3 and very less than Deep Q Learning approach
    c) Performs worse in initial stages
    d) Performance score increases over time.
7) Random and predicted movements based on saved and loaded model using actor-critic approach.

a) Performs better with saved data
b) Still slow in taking shots as compared with Deep Q Learning approach

8) Batch size experiments with both models.

a) We experimented both the models with various experience replay and training data size and observed that our model did not perform well with smaller batch training and did comparatively better when we incremented the batch size from 4 to 8 to 16 and finally we decided that we will use 32 as our batch size as it helps us train the model better.

9) Using sigmoid activation in deep Q learning instead of standard RELU approach

We tried several experiments and in some of our experiments our model started predicting astronomically high values for outputs and after one point it started predicting infinite values on each output node. We tried to reduce these occurrences by using Sigmoid in activation of Dense layers instead of RELU activation, reasoning behind this is pretty simple if we look at RELU 's definition it does not clip positive values while sigmoid tries to reduce them between 0 and 1.

Successes

We tried several experiments on our reward system by giving slabs of rewards not just based on game provided score,but a logic that used various steps to give rewards. We successfully made the whole pipeline that captured images, calculated rewards and took actions all in time to perform 4 to 5 average shots per minute which made 300 epoch per hour training possible for our Deep Q learning model. We have two major models, implemented with this setup.

Evaluation

Our evaluation is mainly based on the grade to the drill given to the player by FIFA free kick taking skill game and we monitor these score per drill to ensure whether or not our model is performing well overall not. We have explained the various evaluation experiments we made to our reward system in the experiment section.

Lessons learned

Learning new libraries to simulate key presses and do OCR to collect scores
Both of us have never explored python that much to do actual applications that required OCR and key simulation. So this was a new experience for us to simulate and analyze OCR images to extract whatever data we wanted out of the screen grab that we had. We experimented various crop sizes and also successfully tackled OCR errors.
Learning how to manage sessions in tensorflow which we entered and exited several times for actor and critic model.

We had to design the actor critic model in tensorflow due to the nature of the model that took in multiple inputs and updated the gradients according to the model training that used various functions to train both the model and this created a problem where tensorflow session was getting closed and opened multiple times and we had to manage that so that it does not blow apart the training process. We had to learn how to do that and it was a pretty good learning experience,

<u>Experiments</u>

Various act setups:

a)With no random movements, it just takes shoots after some training, before that the Model is more of a random predictor and does movement randomly.

b)With random movements, it does give left or right actions to the model whenever model fixates itself into shooting while training due experience batch.

State experiments:

Image cropping with performance: Giving in various parts of screen as state and experimenting to reduce the size of image so that the deep learning model sees only the parts of the image it needs to. For eg. we removed some of the top pixels which displayed score and sky and will never be needed to decide the movement for the free kick. This is a domain knowledge based experiment in order to reduce the size of the image.We also tried feature extraction with MobileNetV2 and VGG19 but found no sufficient evidence to switch to either of them as they both were acting same as VGG16.

Action Experiments:

Number of actions - 4 [left move, right move, shoot low,shoot high]

Number of actions - 6 with power variations[same as above but with 2 variations of shoot each]

Continuous training with saved and loaded model for multiple 100 epochs.

Cropping the image using cv2 for calculating the rewards:

We had to find amount of rewards that we wanted to give the value of score that is projected on the screen, we cropped the part of the image by experimenting OCR which we use from tesseract APIs to get text from the screen to increase the precision of score gathering mechanism and reduce the number of actions with 0 rewards, still due to OCR failures we sometimes get unnecessary outputs and in that case we give model a minor positive or negative reward based on random choice.

Using sigmoid activation in deep Q learning instead of standard RELU approach:
We tried several experiments and in some of our experiments our model started predicting astronomically high values for outputs and after one point it started predicting infinite values on each output node. We tried to reduce these occurrences by using Sigmoid in activation of Dense layers instead of RELU activation, reasoning behind this is pretty simple if we look at RELU's definition it does not clip positive values while sigmoid tries to reduce them between 0 and 1.

Observations -

8) No random movements with baseline model and reward phase 1
    a) Sometimes always goes in right side, or always left. Bot doesn't even shoot sometimes
    b) After some training it does shoot.
    c) Better free kicks when a wall is present.
    d) After failing in shoot low when a wall is present, it takes high shoot after training
9) Random movements with baseline model and reward phase 1
    a) Does random left right movement
    b) After failing in shoot low when a wall is present, it again takes low shot after training.
10) Random movements with model prediction with Epsilon value with baseline model with reward phase 1
    a) To avoid continuous left or right actions, it shoots low or high randomly as an action, exploring new actions.
    b) Sometimes always goes in right side, or always left. Bot doesn't even shoot sometimes
    c) After some training it does shoot.
11) Rewards are changed with random and no random movements with reward phase 2
12) Rewards are changed with no random movements with reward phase 2
    a) Too many extreme left or extreme right movements and no shoot sometimes.(majorly extreme left)
    b) Repetitive actions.Should add negative rewards for repetitive actions.

Rewards Phase 1 -

Greater than 1000	1
500 to 1000	1
0 to 500	1

No score change : -1

Rewards Phase 2 -

Greater than 1000	5
500 to 1000	2.5
0 to 500	1
No score change	-10