Training Neural Network to play Super Mario

IIT2015032 Rohan MR
IIT2015039 Nishant Verma
IIT2015042 Raghav Saboo
IIT2015045 Harsh Vardhan

Under guidance of

Dr. Sonali Agarwal
IIIT Allahabad

Candidate's Declaration

We hereby declare that the work presented in this project report entitled “Training Neural Network to play Super Mario using Reinforcement Learning”, submitted as 5th semester B-Tech IT mini project is an authenticated record of our original work carried out from August 2017 to November 2017 under the guidance of Dr. Sonali Agarwal. Due acknowledgements have been made in the text to all the resources and frameworks used.

Signature:

Date: 14th September, 2017

IIT2015032 Rohan MR

IIT2015039 Nishant Verma

IIT2015042 Raghav Saboo

IIT2015045 Harsh Vardhan

Supervisor's Certificate

This is to certify that the project work "Training Neural Network to play Super Mario using Reinforcement Learning" is a bonafide work of Rohan MR (IIT2015032), Nishant Verma (IIT2015039), Raghav Saboo (IIT2015042), Harsh Vardhan (IIT2015045) who carried out the project work under my supervision.

Signature
Dr. Sonali Agarwal

Date: 14th September,2017

Acknowledgment

We would like to thank all the people who have helped us in this endeavour. Their support means a lot to us and we wish to thank them individually.

We begin by thanking Dr. Sonali Agarwal for her guidance and constant supervision as well as for providing necessary information regarding the project & also for her support in completing the project.

We would also like to express our special thanks to the panel under Dr. Vrijendra Singh, Dr. Jagpreet Singh, and Dr. Sonali Agarwal for their never ending support.

Table of Contents

Abstract

The aim of this project is to build a Neural Network Model using reinforcement learning that will complete all the levels of the game Super Mario Bros comparable to an expert human player or even better. We would first develop an initial model, upon achieving sufficient performance in the first model we would vary the number of layers in the Neural Network model, and compare them on the basis of performance and efficiency.

INTRODUCTION

Since A.I has increasingly become an integral part of human life, it has become important to develop better AI each day. In this endeavour we would like to train one such A.I.

Games, like many endeavours are about reacting to and predicting events in the pursuit of goals. Gaming domain involves application of Several A.I methods. Application of these methods results in development of more human like agents which raise player's emotional involvement.

For developing more human like behaviour in gaming agent we use an A.I technique called Reinforcement Learning that allows agent to learn from observation.

Our contribution include implementing and testing application of Reinforcement Learning to a Super Mario non player character. We will address optimizers, objective methods, choices for selecting them and evaluate our model in terms of performance and efficiency.

MOTIVATION

In the quest of developing a universal learning algorithm, a game is a perfect and simple environment with predefined set of rules, actions and rewards.

Games, like many endeavours are about reacting to and predicting events in the pursuit of goals. Gaming domain involves application of Several A.I methods. Application of these methods results in development of more human like agents which raise player's emotional involvement.

Developing gaming agent that imitates human behaviour has its own Pros. We create games that are decisive, intelligent and believable non player characters that engages with opponent in nondeterministic fashion. Gaming involves nondeterminism which is the basis of real world. So developing agent resembling real world is desirable.

Problem Statement

To train various neural network models to play Super Mario using Reinforcement Learning and to compare the different models on the basis of performance.

Literary Review Report

Introduction to Artificial Neural Network [1]

Artificial Neural Network (ANN) are statistical and mathematical models which take their inspiration from neurons in the human brain. These networks are designed to portray the same learning and logical adaptability capability as the human brain. These networks are represented as systems of interconnected neurons, which send messages to each other.

Artificial Neural Networks consist of several nodes, which work like the neurons of human brain. Like the neurons the nodes take inputs and accordingly performs operations on these inputs and pass the result to other nodes. The neural network contains several layers of nodes. Each link between layers has a weight parameter that manipulates the data in the calculations.

There are mainly 6 characteristics of Artificial Neural Networks-
1- Network Structure
2- Parallel processing
3- Fault Tolerance
4- Collective Solution
5- Distributed Memory
6- Learning Ability

There are mainly 2 types of Artificial Neural Networks-
1- Feedforward ANN 2- Feedback ANN

Feed forward ANN - In Feed forward ANN the information flows in only one direction. One layer after doing the calculations passes the information to next layer. There are no feedback loops.

Feedback ANN- In Feedback ANN backward information flow is allowed. One layer after doing the calculations can pass the result to previous layers also.

Advantages of Neural Network-

1- Adaptive Learning 2- Self- Organisation

3- Real Time Operation 4- Pattern Recognition
5- Flexibility

Convolutional Neural Networks [2]

Convolutional Neural Networks fare particularly well when the feature set consists of image data than traditional fully connected neural networks. Convolutional Networks also take into account that objects can be shifted, scaled or distorted. To do this it uses shared weights, sub-sampling in the spatial domain and receptive fields. Each part of a layer takes input only from a specific small part in the previous layer. Each of these is a local receptive field. Local receptive fields help the neurons to get the basic visual features such as blocks, enemies, pipes etc. These features are then used in the next layer and so on, to develop higher level features. Each part of a layer is stored in different planes and in them all parts share the same set of weights. The set of outputs of such a part is known as a feature map. Parts of a feature map do the same operation on different parts of the image. A complete convolutional layer is composed of several feature maps, so that it can extract multiple features at each location. Other feature maps have different sets of weights and hence they describe different parts of the image. A feature map would scan the entire image and pick up some features and store it some location in the map. Once you obtain a feature, we do not need to have the information about where it came from, so that it can be compared to any part of any image. We can do this by reducing the spatial resolution of the feature map. This is also known as sub sampling of the layers. This reduces the sensitivity to shift in position of characters and different action they are performing.

Playing Atari games with deep reinforcement learning [3]

In this paper the aim was to create a single Neural Network model using Reinforcement learning that can play different Atari 2600 games without having to provide any game specific information except the actions you can take in each game and scoring function and keeping it as generic as possible.

Generally deep learning models are trained by using a huge data set whereas in RL training is done on the basis of observation. The agent

interacts with the environment and takes an action that maximizes it' s reward. It is easy to measure performance in case of supervised learning by using testing data but in RL it is a challenging task.

The input given was raw pixel data representing the game screen. The agent selects a particular action from the set of legal or valid actions in such a way that reward increases by a maximum amount. The action is given to the Atari emulator which performs it and changes the environment and game score accordingly. If it performs a right action the reward increases otherwise it decreases. It learns from a sequence of actions and observations. It always tries to predict the best move and use the outcomes of those moves to retrain the model and repeat the above process until the game is finished.

Thus a Neural Network model was built that could play various Atari 2600 games using only raw pixel data as input with no change in design and architecture according to a specific game.

Optimizers

Neural Networks use optimizers in their back-propagation to readjust their weight values. The most commonly used optimizers are:

1. Momentum Optimizer
2. AdaGrad Optimizer
3. RMSProp Optimizer
4. Adam Optimizer

Momentum Optimizer: [4]

The back-propagation learning algorithms used in neural networks are stochastic gradient descent algorithms. In the generic versions of algorithms, they tend to descend very slowly along the steepest valleys and tend to oscillate across the other axis. To hasten up his convergence process, another term is added to the objective function of the optimizer. The relation between this term and the original is analogous to that of weight and momentum in Newtonian physics.

In mathematical terms:
Stochastic gradient descent:
$\theta=\theta-\eta\cdot\nabla\theta J(\theta;x(i);y(i))$
Momentum optimizer:
$vt\theta=\gamma vt-1+\eta\nabla\theta J(\theta)=\theta-vt$
The adding of the momentum term hence reduces the oscillation and hastens the convergence.

ADAGrad Optimiser [5]

- Adagrad is an algorithm for gradient based optimisation, which is well suited for sparse data.
- It uses the fact that infrequently occurring features are highly informative and discriminative.
- This algorithm gives frequently occurring features very low learning rates and infrequent features high learning rates.
- Adagrad is useful for natural language processing and image recognition.

AdaGrad update rule is given by the following formula:

$$G^k = G^(k-1) + \nabla J(\theta^(k-1))^2$$
$$\theta^k = \theta^(k-1) - \alpha \cdot \nabla J(\vartheta^(k-1))/\text{sqrt}(G^(k-1))$$

where · and sqrt are element-wise operations.

ADVANTAGES

- Most implementations use a default value of 0.01 and leave it at that.
- It eliminates the need to manually tune the learning rate.

DISADVANTAGES

- Its main disadvantage is that it accumulates the square gradients in the denominator.
- This makes the rate of learning small and eventually become infinitesimally small.

RMS PROP OPTIMIZER [7]

- RMS PROP is a mini-batch version of RPROP optimizer.
- It utilizes recent gradients to normalize current gradients.
- We calculate average over root mean square of recent gradients and divide current gradient with this value.
- This also includes adaptive step rates, min and max value and if component of step and momentum points in same direction we add step rate to it else subtract it. Exceeding values are neglected.
- Mathematically,

$$r(t)=(1-y)*(f'(Theta(t))*f'(Theta(t)))+y*r(t-1)$$
$$v(t+1)=a*f'(Theta(t))/sqrt(r(t))$$
$$Theta(t+1)=Theta(t)-v(t+1)$$

Here, a=step rate
y=decay rate
f'(Theta(t))=derivative of loss function w.r.t parameters at time step t.
Theta(t)=Update function at time step t

ADVANTAGES

- Robustness
- Applicable to mini-batch learning as it deals with stochastic objective nicely.

DISADVANTAGES

- Unpublished method (theoretical method so implementing it involves other methods like momentum etc.)

METHOD USED

RmsProp (wrt, fprime, step_rate, decay=0.9, momentum=0,min=0,max=inf, step_adapt=False,args=None)
Wrt:-array that represent solution.
fprime:-return search direction as gradient.

CONCLUSION

We didn't use RMS Prop because of decay parameter and it's always implemented along with other for better output.

Adam Optimizer [6]

Adaptive Moment Estimation(Adam) is a gradient descent optimization algorithm. It performs optimization and is considered to be one of the best optimizers available at present.

Gradient descent is an algorithm for finding the minimum of a function. It is used to perform optimization and it can also optimize a Neural Network, all the best libraries available for machine learning and deep learning (like Tensor Flow , Keras , etc.) have implementation of these algorithms. Generally they are used as black box without going much into the implementation details. It is also used in machine learning because here we try to minimize error by reducing a cost function.

Stochastic gradient descent is also an algorithm for minimizing an objective function. Stochastic gradient-based optimization used in many areas in the industry. Many problems can be converted into an equivalent problem that is minimizing an objective function based on some parameters.

Key points used in this algorithm-

- We compute the gradient.
- Then we compute the running average of gradient and squared gradient
- Every parameter has a different learning rate unlike SGD in which we maintain a single learning rate which does not change during the entire process

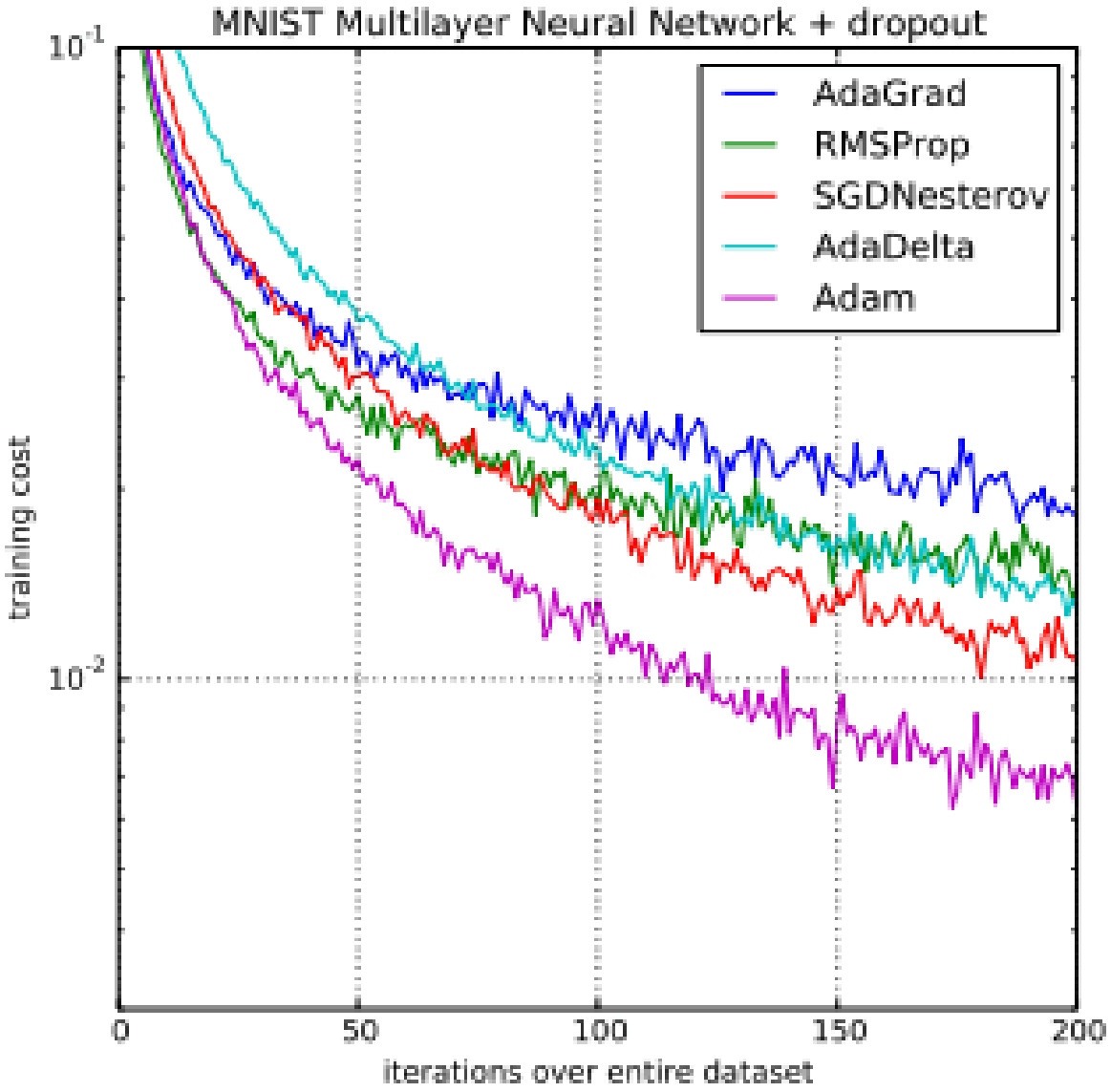
Advantages-

- It combines the advantages of other gradient descent optimization algorithm like Ada Grad and RMS Prop
- Easy to implement, less computation power required
- Good for problems with very varying or noisy gradient
- Results have shown that it works very well for solving practical deep learning problems

Parameters used are epsilon which is small no to prevent any division by zero in the implementation (like 10^--8). The second parameter is beta1 which is the exponential decay rate for the first moment estimates (like 0.9). The third one is beta2 which is the exponential decay rate for the second-moment estimates (like 0.999) and the last one is alpha which is the learning rate or step size.

Good default settings for machine learning problems use beta1=0.9, beta2=0.999, alpha=0.001, and epsilon=10^-8.

Amongst these optimizers, Adam optimizer is the most sophisticated and it is most commonly used as it gives satisfactory results. Based on this analysis, we will be using Adam optimizer in our project, hence forth.



ACTIVATION FUNCTION [8]

- Key Terms: Activation function, Vanishing Gradient problem

- ACTIVATION FUNCTION
  - It's functional mapping between inputs and response variables (possibly nodes of hidden layer).
  - It's non-linear in nature.
  - Its purpose is to introduce non-linearity. We pass sum of products of input and weights to activation function as input which it passes to hidden layer as input or to output layer for output.

- It should be differentiable.

- VANISHING GRADIENT PROBLEM
  - Adding more hidden layer enable network to learn more complex function which in turn enhances predicting future outcomes.
  - So neurons in earlier layer learn slowly in comparison to neurons in later network. As early neuron network are basic building blocks of network slow learning will result in long training process and prediction of model will decrease.

- TYPES OF ACTIVATION FUNCTION

  - SIGMOID
    - It's S shaped curve of form f(x)=1/(1+exp(-x)) whose range is between 0 and 1
    - ADVANTAGE
      - It's easy to use, understand and apply.
    - DISADVANTAGE
      - Suffers from Vanishing Gradient problem and slow convergence.

Non-zero centred making optimization harder as it makes gradient updates go too far in different directions.

  - HYPERBOLIC TANGENT FUNCTION
    - It is of form f(x)=(1-exp(-x))/(1+exp(-x)) whose range is between -1 and 1
    - ADVANTAGE
      - It's easy to use, understand and apply.
      - It's zero centred so optimization is easy.
    - DISADVANTAGE
      - Suffers from Vanishing Gradient problem.

- RECTIFIED LINEAR UNITS(RELU)

  - It is of form R(x) = Max (0, x).
  - ADVANTAGE
    - ➢ It's simple and easy to implement
    - ➢ It's improvement over tanh function in terms of convergence.
  - DISADVANTAGE
    - ➢ Suffers from Dead Neuron problems in which some gradients dies out during training because of fragility and weight update during backpropagation is useless. We use Leaky RELU which introduces small slope to prevent from dead neuron problem.
    - ➢ It's only used with hidden layers. So for output layer we use Softmax function for computing probabilities of classes.

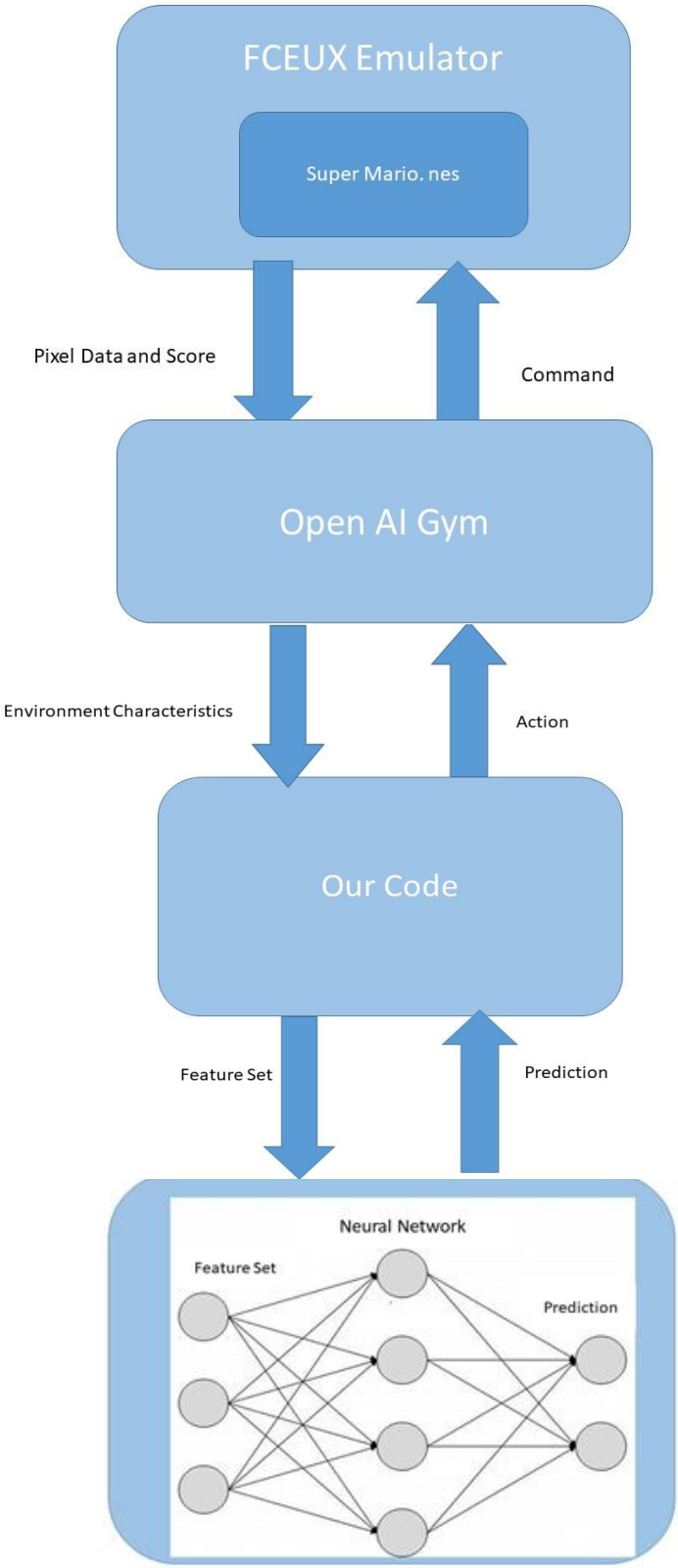
➢ CONCLUSION

We are using RELU in our project because of above mentioned advantages as other activation function suffers from Vanishing Gradient problem.

Plan of Action

- Study basics of Neural Network and Reinforcement learning
- Setting up environment
- Generating initial population
- Training the Neural Network
- Analyze and compare different models on the basis of performance and efficiency

Flow chart of overall process



Software Requirements

Software's used -

- OS used - Ubuntu 16.10
- Python 3.5.2
- Pycharm Community Edition 2017
- Fceux Emulator
- Libraries used-
    - Gym
    - Tensor Flow
    - Keras
    - Numpy

Implementation Details

We generated our initial training data by assembling together many games played by human experts. This training data was fed to our neural network which we build using Tensor Flow and Keras which are python libraries for deep learning. We have used four fully connected layers two are hidden layers and one is input layer and the other one is output layer. We have used two activation functions that are rectified linear and linear activation function. Activation function plays a major role as it determines the output of a neuron for a given set of inputs. We have used Adam optimizer as it is considered as one of the best optimizers available for optimizing the neural network by adjusting the weights thereby producing better results.

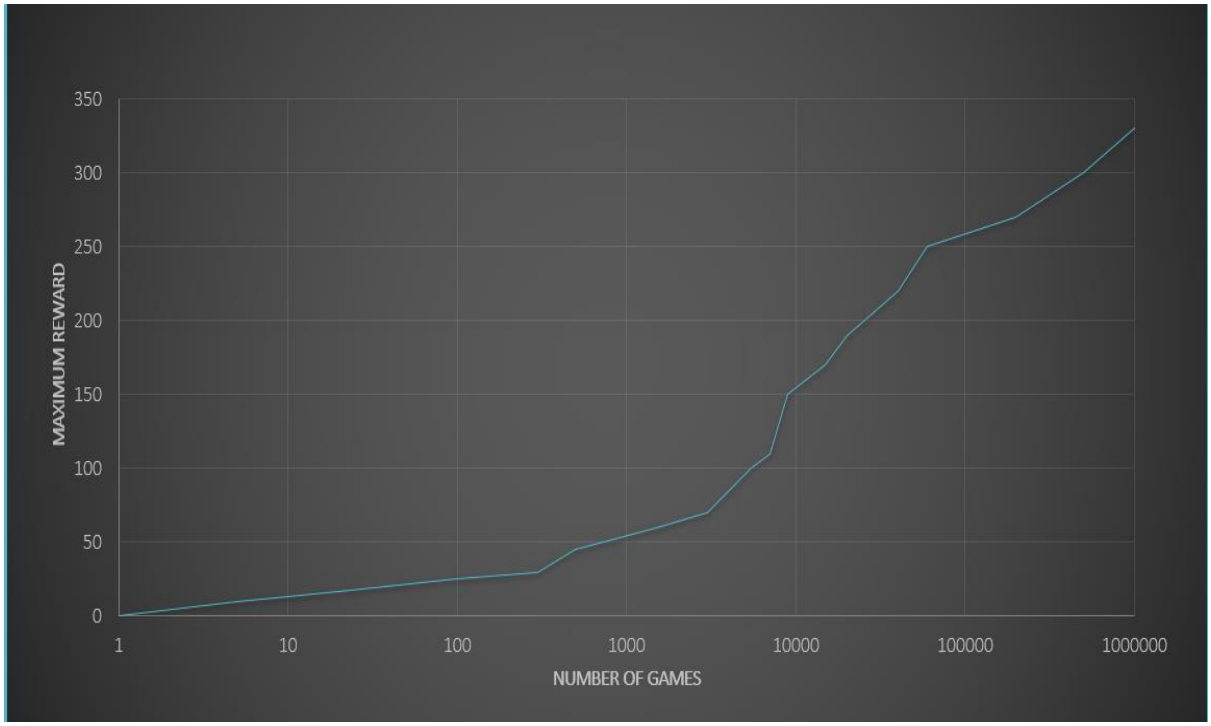
Comparison between Supervised and Reinforcement Learning

Training with reinforcement learning has pitfalls of time and memory requirement.

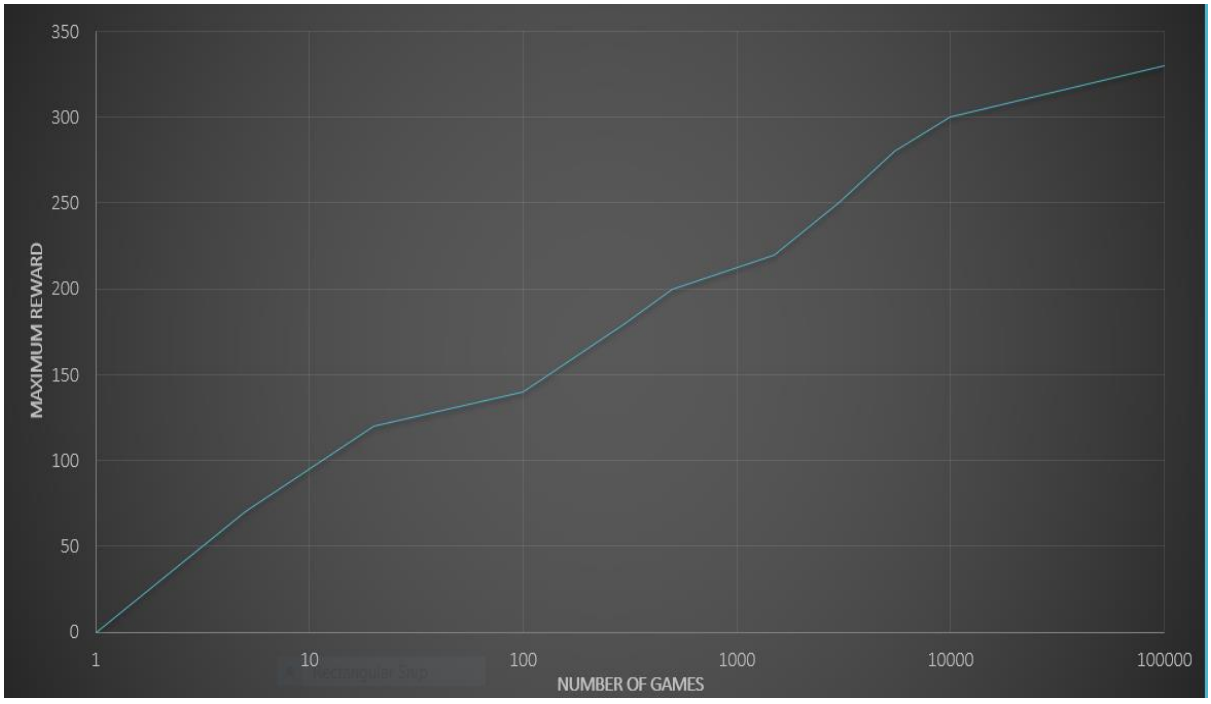
For achieving significant improvement in performance in limited time and memory there is an urge to choose suitable model and training method. Using supervised learning, instead of trying to generate random moves and then training on it, we rather train it on the provided good move.

So, rather than predicting and taking actions on random moves and then learning from it , it takes actions on provided moves due to which model trains within feasible time and within physical limitations.

There is pitfall of using supervised learning as it gives level specific solution but it is negligible with respect to the performance and reward it achieves.

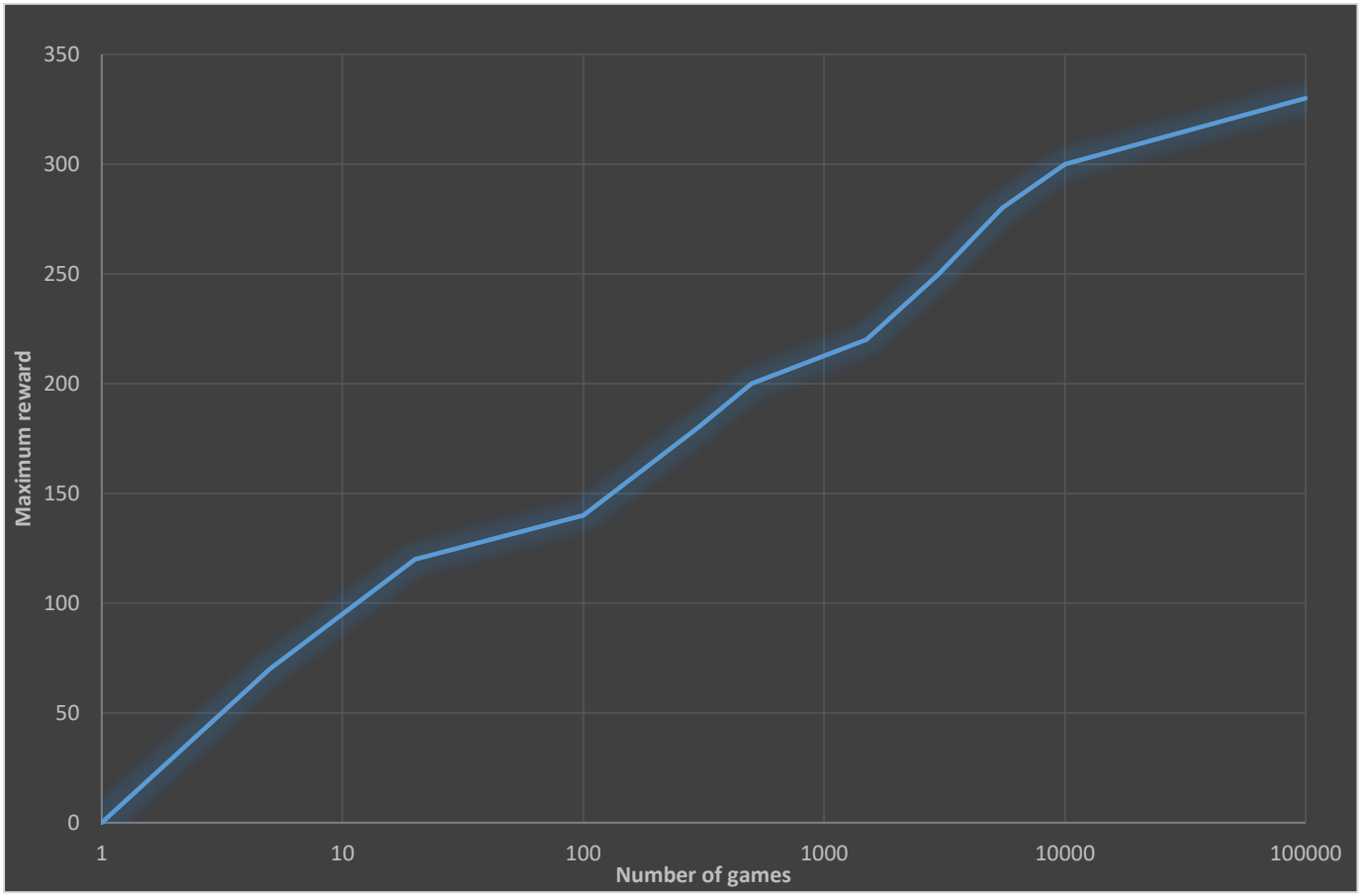


Reinforcement Learning



Supervised Learning

Results and Conclusion



The above graph is plotted for Number of games vs Maximum Reward that we observed by training our model using Supervised Learning. Thus we developed a model which can successfully complete the initial level of the game Super Mario Bros.

References

[1] P. S. Praveen Kumar, “Artificial Neural Networks-A Study,” *International Journal of Emerging Engineering Research and Technology ,* vol. 2, no. 2, pp. 143-148, 2014.

[2] P. H. B. B. Yann LeCun, “Object Recognition with Gradient-Based Learning,” in *Shape, Contour and Grouping in Computer Vision*, Springer, Berlin, Heidelberg, 1999, pp. 319-345.

[3] K. K. D. S. A. G. I. A. D. W. M. R. Volodymyr Mnih, “Playing Atari with Deep Reinforcement Learning,” eprint arXiv:1312.5602, 2013.

[4] N. Qian, “On the momentum term in gradient descent learning algorithms,” *Neural Networks,* vol. 12, no. 1, pp. 145-151, 1999.

[5] E. H. S. John Duchi, “Adaptive Subgradient Methods for Online Learning and Stochastic Optimization,” *Machine Learning Research,* vol. 12, pp. 2121-2159, 2011.

[6] J. B. Diederik P. Kingma, “Adam: A Method for Stochastic Optimization,” in *3rd International Conference for Learning Representations*, San Diego, 2015.

[7] “RMS Prop Documentation.,” *URL: http://climin.readthedocs.io/en/latest/rmsprop.html.*

[8] “Activation Function,” *URL: https://medium.com/towards-data-science/activation-functions-and-its-types-which-is-better-a9a5310cc8f.*

Suggestions And Remarks: