Implementation of a 2D A. I. Agent for non-deterministic games using Convolution Neural Network

Vishrut Sharma
Computer Science and Engineering
Jaypee Institute of Information Technology
Noida, India
vishrut11@hotmail.com

Devashish Satija
Computer Science and Engineering
Jaypee Institute of Information Technology
Noida, India
devashish.satija38@gmail.com

Ayush Srivastava
Computer Science and Engineering
Jaypee Institute of Information Technology
Noida, India
ayusharu8.24@gmail.com

Akanksha Mehndiratta
Computer Science and Engineering
Jaypee Institute of Information Technology
Noida, India
mehndiratta.akanksha@gmail.com

Abstract: **Artificial Intelligence (A.I.) in video games is largely used in designing an agent, also known as Non-Playable Characters (NPCs), that mimics characteristics of a human player. Evidently, the field of A.I. agent-based game development is flourishing due to recent advancement in the field of deep learning. This study presents the role of different Convolution Neural Networks (CNN) employed in the development of A.I agent-based games and also proposes a model which is a variant of CNN. For the purpose of this study the game used is a basic jumping platformer game in which the game character, which is of the primary focus, is tasked with dodging obstacles by either jumping or ducking. The obstacles are generated randomly and include objects like cactuses and flying vultures. The model proposed, is a variant of CNN that fashions an AI agent that predicts the game character's movement, based on the current environment of the game, to mimic the performance of a human player. The performance of the model proposed results in an accuracy of 97.03 % on both the testing and training set.**

Keywords—Convolution Neural Network (CNN), Game Development, A.I. agent, Non-Playable Characters (NPCs)

I. INTRODUCTION

Recent paradigm in the design of various video games focuses on fashioning non-playable characters (NPCs) that are derived from the field of Artificial Intelligence (AI). The development of powerful microprocessors enables game developers to create computer opponents that are challenging and can outsmart human player. AI based NPC's monitors the human player behavior, analysis his actions and surroundings and adapts itself to the responses provided by the human player. Few popular AI techniques employed in game development may include decision trees [11], pathfinding [12], etc. that employ simple if. else loops or complex algorithms like Dijkstra's algorithm [13], A* algorithm [15] and many more.

For analyses of human player behavior in a game using an AI agent, the game is inputted as a continuous sequence frames paired with human player action as output. In the context of this study, a frame refers to an image that captures the environment the player is placed in the game. Over the past few years, Convolution Neural Networks (CNN) have proven to be very efficient in the field of classification [21][22] and are usually the preferred technique for learning human player behaviors and predicting AI agent action for an input frame [14].

A CNN uses a special convolution operation to detect patterns in an image and learns to predict based on these patterns [6][9]. These patterns are generally made up of different features the net may find in the said image [1]. However, the learning of the AI agent is limited to frames captured in the training set [8]. For a human player, the time to make a move in the game is purely random and is based on the oncoming obstacles, the AI agent works in a similar way where it takes the screenshot of the game and feeds it to the trained CNN and predicts the next move based on the output given by the network [10].

In this study, we discuss CNN in detail and further develop an understanding of how supervised deep learning techniques can be used to bring a more intelligent behavior in the world of game development. Similar implementations are possible for games where actions depend on some random event which could be captured in an image [10]. Section II presents the related work in the development of AI agent-based game development. Section III describes a Convolution Neural Network and explores the different layers used in a typical CNN. The following section describes the proposed model. Section V describes the data capturing process to prepare the dataset along with the training process using the proposed model. In Section VI, we discuss the different analysis techniques used for CNN and tabulate our results to analyze

978-1-7281-3591-5/19/$31.00 ©2019 IEEE

the performance of the proposed model. Finally, we conclude and provide insight into the subject of supervised learning techniques being used in designing A.I. behavior in video games.

II. Related Work

For more than decade researchers have studied AI agents in game development. Techniques used vary from Decision Tree, Dijkstra's and A*Algorithm and, as a result of advancement in technology, deep learning. In recent years, deep learning techniques have gained the most acceptability in the development of AI-based games. Convolution Neural Network is usually the preferred technique for learning human player behaviors and predicting AI agent action for an input frame [6].

For designing a decision tree based AI game agent Pramila et al [11] extract a set of physiological signals from the human player. The authors conclude that predicting a human's affective state may be considered as a classification problem and hence employ a decision tree to determine the effective state from a set of features derived from physiological signals. Extraction of such physiological signals affects the player's performance as the process may seem, to them as invasive [17].

Pathfinding in game development has been researched for the past few years. Although there are algorithms like Dijkstra's Algorithm, Breadth-first search and depth-first search algorithm to solve the shortest distance problem, but A*[18] Algorithm is the most optimal solution for pathfinding [15]. " Age of Empires" is a classic real-time strategy game that functions on grids to outputs map representation and A* algorithm to determine the optimal heuristic function that governs the movement of the militants. For designing real-time applications, an AI agent is embedded with sensors to comprehend the player's surroundings and act accordingly. The problem with current path-finding algorithms is they lack the capability of extracting important information in real-time without putting too much of a strain on the computer's resources. Artificial neural networks have proven to provide an effective solution to this problem [19]. Although the performance of the learning model in neural networks is dependent on the representation of data but it was deep architectures that enabled learning of high level abstractions [20].

Chris et al [14] and Clark et al [4] proposed AI agents for Go player that takes board image as an input. The authors trained the agent using deep convolution network. The agent proposed was able to win some games against the best A.I. routine for Go, namely Fueg and GnuGo, and in lesser time. Krizhevsky et al [16] trained a large deep CNN, that consists of 60 million parameters, 500,000 neurons with five convolution layers, to classify the 1.3 million high- resolution images. The error rates for the proposed agent were very less compared to the state of the art. Glorot et al [3] present the challenges in fashioning AI agents based on deep architectures.

III. Convolutional Neural Network

A Convolution Neural Network consists of five basic layers namely- Convolution Layer, Pooling Layer, Activation Layer, Flattening Layer, Fully Connected Layer [5]

Feature Extraction is performed by the first three layers and Classification of the Image is performed by the Flattening and Fully Connected Layer. The basic building block of a Convolution Neural Networks is a neuron. In CNN's, each neuron is incorporated with an activation function, output of which acts as input to next neuron. The final output classifies the input data into a class.

A. Convolution Layer

It is the first layer of the network that takes an Image in form of a pixel Value Array with dimension (h*w*d), where h and w are the height and width of the image respectively and d refers to the number of channels which is 1 for a grayscale image and 3 for an RGB image. The aim of neurons in this layer is to perform feature engineering by mapping extracted features from a learnt set of features and produce a high activation on correct mapping. In image processing, the computation done in the convolution layer involves superimposing the filter on the input as a – "receptive field" of interest. The filter is then moved across the input and the dot product (convolution) between the two is calculated and a 2- D feature map of the extracted spatial feature is produced. Padding is normally added to produce a desired size of the output map. The output of this layer is a convolved feature Map

B. Pooling Layer

Pooling layers are used in Neural Networks for down sampling an image over the width and height dimensions but the number of feature maps remains unchanged. There are two types of pooling: -

1) *Max Pooling:* It selects the maximum pixel value across the field of interest.

2) *Average Pooling:* It takes an average of the pixel values across the field of interest.

The Input to the Layer is the convolved feature Map that was output of Convolution Layer.

C. Activation Layer

In this layer, an activation function, exemplar Linear function, Sigmiod, TanH and RELU,are applied on the output of the previous layer. It performs a transformation on the input to enable it to learn complex function. The output from the Pooling Layer acts as input to the activation Layer. After using the Activation Function the dimension does not change but only important features are retained in the image and the unnecessary features are removed.

D. *Flattening Layer*

The input to a fully connected layer is a one-dimensional vector hence the task of flattening layer is to flatten the output of the previous layer to a one-dimensional vector.

E. *Fully Connected Layer*

The fully connected layer performs the high-level reasoning. It is named so as it is connected to all activations in the previous layer to perform compilation of all the learnt features. The compiled result is then used to define the output neurons (generally equal to the no. of classifications needed for image classifications task).

Krizhevsky et al [7] discuss the different types of layers which are typically present in a ConvNet and summarized different analysis techniques which are used to test the performance and capability of a ConvNet.

IV. Proposed Model

A. Convolution Layer

The neurons in this layer maps extracted features from a learnt set of features and produce a high activation on correct mapping. The model proposed contains a Conv2D having 32 features, 64 and 128 features of the size 3X3 with stride of 2X2 having the activation as ReLU,

B. Pooling Layer

In our model the image is down sampled using a 3X3 filter Max pooling layer, this type was chosen for pooling because of the high contrast between the features and the background in the images in our dataset. There is a possibility of unnecessary noise being added when average pooling layer is used.

C. *Activation Layer*

In our proposed model the RELU activation function is applied as the activation speeds up the computation and therefore results in an overall optimized run-time.

D. Flattening Layer

The model performs flattening by simply arranging the 3D volume of numbers into a 1D vector.

E. Fully Connected Layer

The model is using a Dense Layer having 256 1D elements with ReLu activation and also classification as three (for following the three classes identified by the game - jumping, ducking and doing nothing) with softmax activation. In our model we have implemented this layer using the dense layer in Keras to get the predictions in three defined classes (to jump, to duck, to do nothing). To implement the pre-trained model ResNet50, Keras comes packaged with the model's weights and biases trained on the ImageNet Dataset. To train the model on our dataset, we froze all the layers in the model except the last 6 layers to address the time constraint and also to help the model converge faster.

F. Dropout Layer

The model also introduces a dropout layer. In this layer, the contribution of certain neurons is suppressed during the forward pass weight updates are not applied during the backward pass. The dropout layer here is used with a probability of 0.25.

V. Dataset Preprocessing And Preparation

The game chosen for the implementation is a simple jumping platformer where the aim of the player is to duck, jump or stay idle over certain obstacles coming infinitely towards the player. As the game precedes, the number of obstacles per frame increases to escalate difficulty. For the purpose of the study, we use the game developed by Google "Easter egg" in offline mode in the Chrome browser. Hence, the following three classes are identified by the AI Agent - jumping, ducking and doing nothing. The obstacles are generated randomly and include objects like cactuses and flying vultures.

To capture the data, the agent performs video-capturing and stores a frame (the image of the play area) and the data associated with the input of the player at that particular frame in a CSV file. In the file, we store 0 for the frame in which the player is idle, 1 for the frame in which the player is jumping and 2 for the frame in which the player ducks under an obstacle.

In order to reduce noise, the following steps are performed to

1. Frames are converted to grayscale

2. The frames are cropped in order to include only the area in front of the player which will reduce the problem space.

To further reduce the memory used, the frame only includes the edges of the obstacles using an edge detection algorithm. After preprocessing, the size of the frame came out to be 240 X 345 X 1 because the frame consists of only one channel (grayscale image). Hence, the dataset prepared identifies the frame as the input image and the input of the player as the output class in the CSV file. After the preparation of the dataset, the model was designed using the sequential model class in Keras. It is to be noted that if the player cannot dodge the obstacle it will result in a game over and would require a key press to restart the game. Then we compare the performance of our model with VGGNET [1] for which we have to resize our image frame to 224 X 224 X 3 where 3 depicts the number of channels in which the image is captured and also compared it with RESNET50. The ResNet and VGGnet are trained for the ImageNet dataset which has over 1000 classes to predict from hence the last dense layers are modified in order to predict three classes.

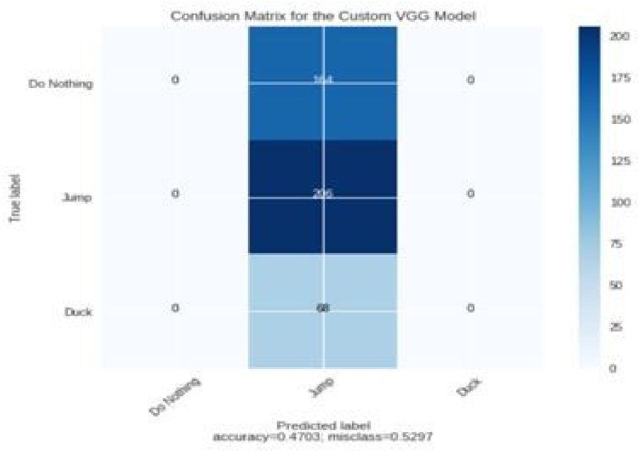

Fig. 1. Confusion Matrix for VGGNet

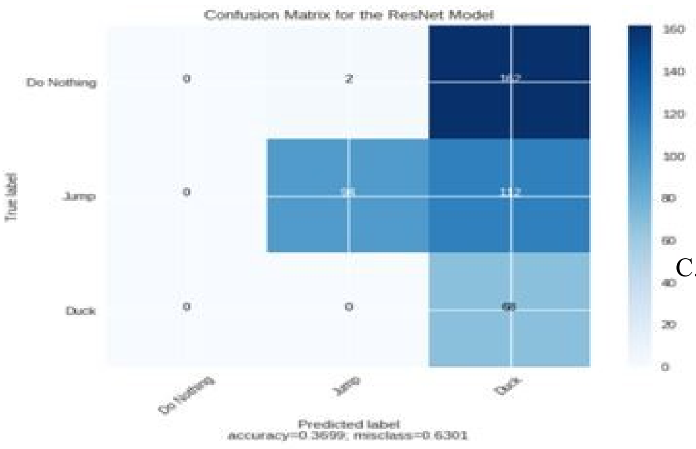

Fig 2. Confusion Matrix for ResNet

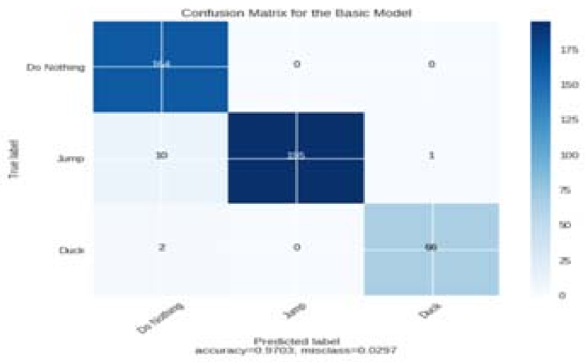

Fig 3. Confusion Matrix for the proposed model

VI. Analysis of Results

The training and testing process were monitored for every epoch and the following quantitative measures were plotted for a better understanding of how the proposed model was functioning

A. *Confusion Matrix*

A confusion matrix is a matrix which contains all the right and wrong classification. Fig 1 and Fig 2 presents the confusion matrix obtained for model based on VGGnet and ResNet while Fig 3 shows the confusion matrix obtained for the proposed model.

The label on the right axis serves as a visual reference to the numeric value plotted and, on the bottom-axis, it is the predicted label. On the left axis, it represents what actually the agent took the action. According to these confusion matrices, our proposed performs best compared to the other two models i.e. ResNet and VGGNet. The accuracy of this model on the testing set is coming out to be 97% which is far better as compared to both the other models. In the confusion matrix of the ResNet model, we can observe for the majority number of frames in the testing set the prediction is to "DUCK" which tells us that the performance is not very good. In the ResNet case, the accuracy is coming out to be around 37% which is very poor compared to our proposed model. The confusion matrix of VGG-net states that accuracy comes out to be around 47% which is better compared to ResNet but very poor compared to our own model.

B. *Validation Curves: Loss and Accuracy Graph*

C. The Validation curves for quality metric as Accuracy and Loss are used as an evaluation metric on the vertical axis and number of epocs on the horizontal axis. Fig 4 and Fig 5 presents the loss and accuracy graph obtained for model based on VGGnet, Fig 6 and Fig 7 for ResNet while Fig 8 and Fig 9 shows the same obtained for the proposed model. In Fig 4, it is observed that the accuracy steeply decreases at around 64th epoch, which makes the data over fit and the gradients to vanish after a certain epoch. Similar effects are also observed in the loss graph which gives the indication of gradients being vanished after a certain threshold of epochs. Also note in Fig 6 the problem of heavy overfitting which causes the Resnet testing plot to diverge heavily from the training data as the epochs increase. Hence the performance reduces as the no. of epochs increase. The model here clearly overfits in the dataset and hence keeps diverging from the training plot. If we increase the no. of epochs, the error should come down in theory.

In Fig 8 and 9 it is observed that the accuracy increases with the number of Epochs of both the training and testing data. As the plots almost match each other, the model is not over fitting.

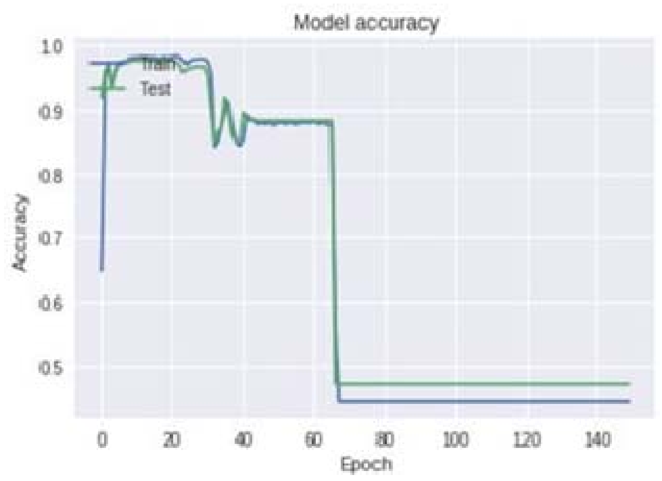

Fig 4. Accuracy Curve for VGGNet

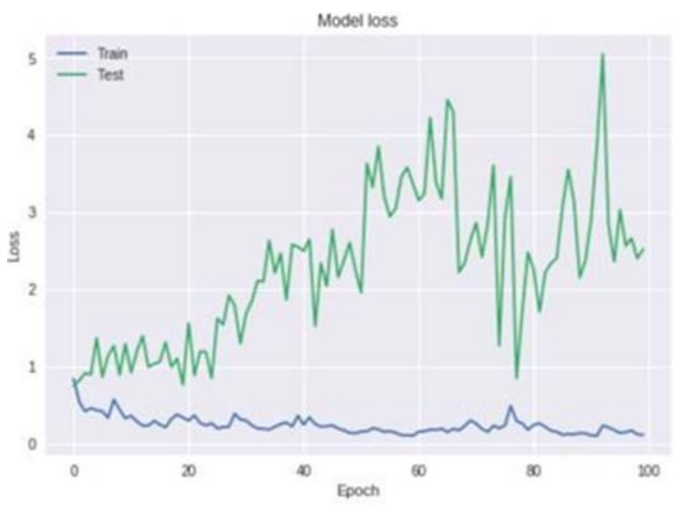

Fig 7. Loss Curve for ResNet

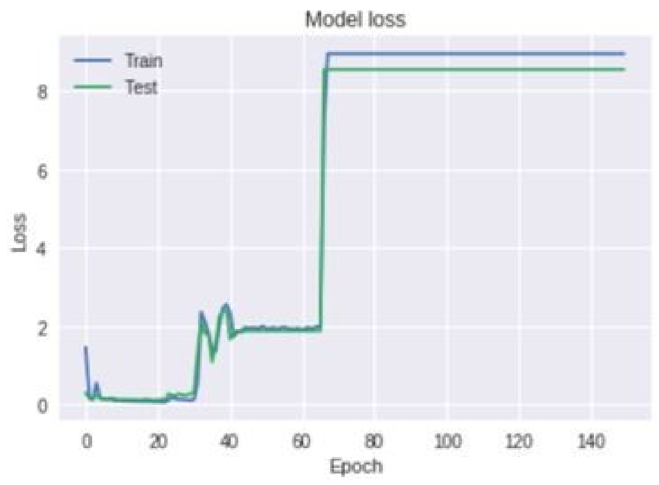

Fig 5. Loss curve for VGGnet

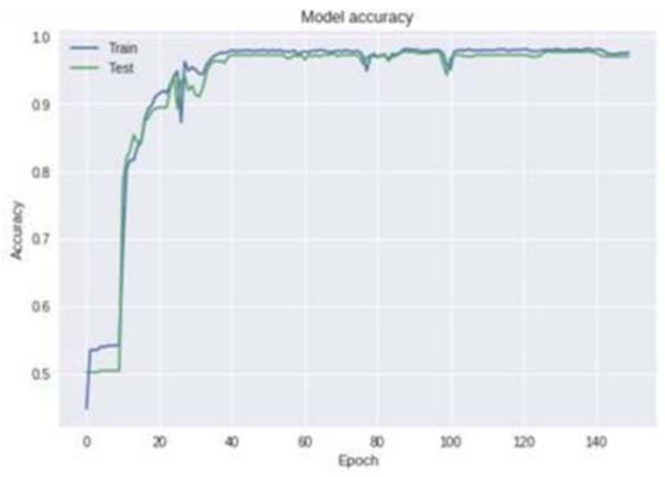

Fig 8. Accuracy curve for our poroposed model

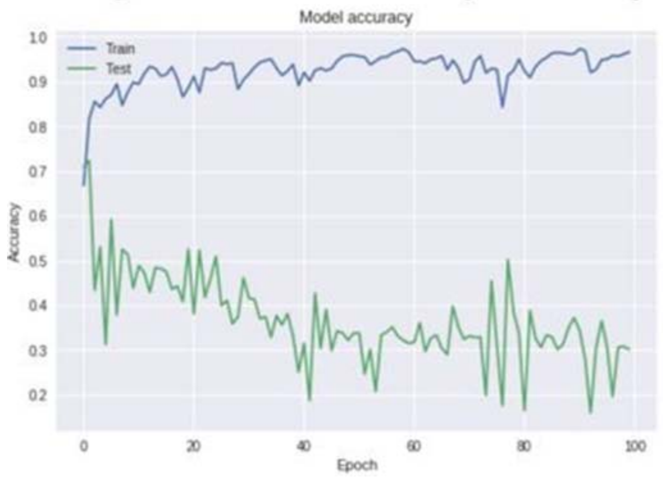

Fig 6. Accuracy curve for ResNet

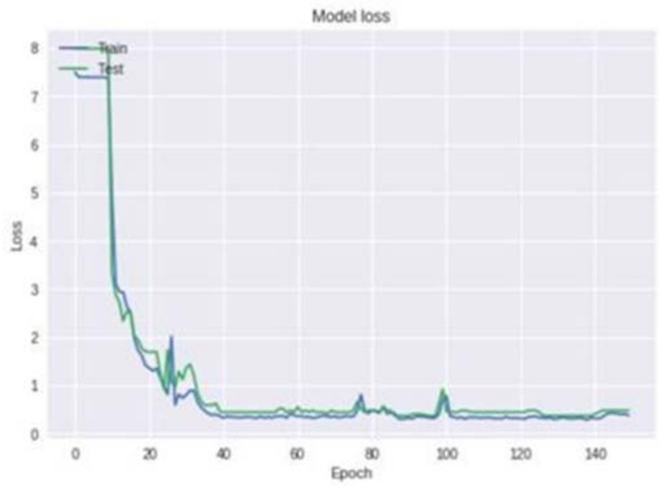

Fig 9. Loss curve for our proposed model

VII. Conclusion And Future Scope

After running the training and validation tests on the three models, it can be observed the model proposed based on a shallow and pooled architecture, performed very well, achieving an accuracy of 97.03%, which is far better than the other models. The loss decreases rapidly after each epoch, giving the indication that our model has performed well on the validation set as well. In these plots, we can see the accuracy increases with the number of Epochs of both the training and testing data. As the plots almost match each other, the model is not overfitting. Although the current dataset if limited in size.

We have kept the focus of this study on different types of Convolution Neural Networks, which is based on supervised learning. Due to recent advances reinforcement learning based AI agents [23][24][25] or a combination of both supervised and reinforcement learning [2] have gained a lot of attention. It would be interesting to analyze the performance of an AI agent in deterministic and non-deterministic games using reinforcement learning.

Acknowledgment

We would like to place our deep sense of gratitude to Mrs. Akanksha Mehndiratta and Mr. Santosh Kumar Yadav for their constant guidance, help and suggestions.

References

[1] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, 2016, pp. 770-778.

[2] Silver, David, et al. "Mastering the game of go without human knowledge." Nature 550.7676 , 2017, p. 354.

[3] Glorot, Xavier, and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks." Proceedings of the thirteenth international conference on artificial intelligence and statistics. 2010.

[4] Clark, Christopher, and Amos Storkey. "Training deep convolutional neural networks to play go." International conference on machine learning. 2015.

[5] Thoma, Martin. “Analysis and Optimization of convolutional neural network architectures.” arXiv preprint arXiv:1707.09725, 2017.

[6] Jordan, Michael I., and Tom M. Mitchell. "Machine learning: Trends, perspectives, and prospects." Science 349.6245, 2015,pp 255-260.

[7] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems. 2012.

[8] Simard, Patrice Y., David Steinkraus, and John C. Platt. "Best practices for convolutional neural networks applied to visual document analysis." Icdar. Vol. 3. No. 2003. 2003.

[9] Szegedy, C., et al. "Going deeper with convolutions in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.", 2015,pp 1-9.

[10] Oh, Junhyuk, et al. "Action-conditional video prediction using deep networks in atari games." Advances in neural information processing systems. 2015.

[11] Rani, Pramila, Nilanjan Sarkar, and Changchun Liu. "Maintaining optimal challenge in computer games through real-time physiological feedback." Proceedings of the 11th international conference on human computer interaction. Vol. 58. 2005.

[12] Silver, David. "Cooperative Pathfinding." AIIDE 1, 2005, pp 117- 122.

[13] Chang, Wen-Chih, Yan-Da Chiu, and Mao-Fan Li. "Learning Kruskal’s Algorithm, Prim’s Algorithm and Dijkstra’s Algorithm by board game." International Conference on Web-based Learning. Springer, Berlin, Heidelberg, 2008.

[14] Maddison, Chris J., et al. "Move evaluation in Go using deep convolutional neural networks." arXiv preprint arXiv: 1412.6564, 2014.

[15] Cui, Xiao, and Hao Shi. "A*-based pathfinding in modern computer games." International Journal of Computer Science and Network Security 11.1, 2011, pp 125-130.

[16] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems. 2012.

[17] Yannakakis, Georgios N., et al. "Player modeling." Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2013.

[18] Hart, Peter E., Nils J. Nilsson, and Bertram Raphael. "A formal basis for the heuristic determination of minimum cost paths." IEEE transactions on Systems Science and Cybernetics 4.2, 1968, pp 100-107.

[19] Graham, Ross, Hugh McCabe, and Stephen Sheridan. "Neural networks for real-time pathfinding in computer games." The ITB Journal 5.1, p. 21, 2004.

[20] Bengio, Yoshua. "Learning deep architectures for AI." Foundations and trends® in Machine Learning 2.1, 2009, pp 1-127.

[21] Chen, Tao, et al. "Improving sentiment analysis via sentence type classification using BiLSTM-CRF and CNN." Expert Systems with Applications 72, 2017, pp 221-230.

[22] Araújo, Teresa, et al. "Classification of breast cancer histology images using convolutional neural networks." PloS one 12.6 (2017): e0177544, 2017.

[23] Mnih, Volodymyr, et al. "Human-level control through deep reinforcement learning." Nature 518.7540, 2015, p 529.

[24]. Mnih, Volodymyr, et al. "Playing atari with deep reinforcement learning." arXiv preprint arXiv:1312.5602, 2013.

[25]. Narasimhan, Karthik, Tejas Kulkarni, and Regina Barzilay. "Language understanding for text-based games using deep reinforcement learning." arXiv preprint arXiv:1506.08941, 2015.