Similarity Model using Gradient Images to Compare Human and AI Agents

Gordon Johnson
Computer Science and Mathematics
Kingston University
London, UK
g.johnson@kingston.ac.uk

Vasileios Argyriou
Computer Science and Mathematics
Kingston University
London, UK
vasileios.argyriou@kingston.ac.uk

Christos Politis
Computer Science and Mathematics
Kingston University
London, UK
c.politis@kingston.ac.uk

***Abstract*—The objective is to determine how close an Artificial Intelligence agent is, in comparison to a human player, using only game play images. Identifying Artificial Intelligence agents during game play is typically done through the analysis and collection of bio-metric data, such as keyboard, mouse and other controller interfaces. This document presents a model of an Auto Encoder architecture, with Long Short Term Memory layers. Gradient and Non-Augmented Images have been evaluated and compared. Three distinct personalities of agents are evaluated, human players, a trained Artificial Intelligence agent, and a basic Artificial Intelligence agent. Through testing the Gradient Image augmentation sets present promising results, with the model successfully identifying the human as the closest similarity to the baseline, followed by the trained Artificial Intelligence agent.**

***Index Terms*—artificial intelligence, auto encoder, motion history image, similarity metric, anomaly detection**

I. Introduction

Artificial Intelligence (AI) can not only successfully compete against some of the top players in the world, in some cases the AI outperforms its human counterpart [1].

This paper seeks to determine how similar an AI agents actions are, to that of a human player. Having a metric that can distinguish or conceal an AI agents presence, has many applications depending on user requirements. Determining such a metric could help create an AI that equally matched its human counterpart, thus creating fairer gameplay and minimising player frustration, especially for those with health conditions. The same metric could also be used in conjunction with bio-metric data to minimise the use of AI agents to cheat in online games.

This main points of contribution are as follows:

- Pre-processing component for manipulation of the image augmentations.
- Introduction of the novel model for determining the similarity of the input sequences, based on a training scenario.
- Evaluation of the model, using three types of agents and how Gradient Images effect the model performance.

Related Works are discussed in Section II. Followed by the Methodology used, in Section III. This section covers the pre-processing component, the similarity model, training, and the evaluation process. Next is the Evaluation and Results Section in IV. Here the two games selected to use in this research are discussed, followed by the particulars of the collected datasets used for this research. Finally Section V concludes the paper.

II. Related Works

From reviewing the literature, much work has been conducted in the area of comparing AI agents to humans, with the objective of detection for anti-cheat systems. A common approach to determining AI behaviour, is analysing bio-metric input actions to determine how human-like the subject is, this can include mouse movements [2], [3], keyboard keystrokes [4], [5] and other user interfaces [6], [7].

This research uncovered a proposal to use a Convolutional Neural Network (CNN) classification model, to determine if mouse operation is consistent with that of a human, by generating an image from mouse movement data containing additional spatial and kinematic information [2]. Similarly, [3] proposes a CNN model with joint multi-label training for mouse-based user authentication, using pre-processing to convert the mouse movement sequences into an image.

For image sequences from video footage, both [8] and [9] make use of an Auto Encoder (AE) with both spatial convolution and temporal Long Short Term Memory (LSTM) layers. The purpose of which is to identify irregularities within the video footage, through the reconstruction of the input sequences. The reconstructed sequences are evaluated using a regularity score, to determine if the current sequence possesses any abnormalities.

For observation of visual data only, the Turing's Test [10] is still a valid method to determine if an AI or a human [11] is being observed.

III. Methodology

This section presents the pipeline of the methodologies used; including the pre-processing component, the similarity model architecture, training methods used, and a process for evaluation of the outputs.

In this approach an AE is utilised to predict the next frame in the sequence of game-play images. For pre-processing we have implemented different augmentations of the images, in order to determine which performed the best. All images are converted to a grey scale and resized to fit the input to the network, one

set is kept as is, and the other is augmented. For each of the augmentations an AE model is trained, using human played image sequences. For testing, AI controlled sequences, with a subset of human controlled sequences not previously seen during training, were utilised.

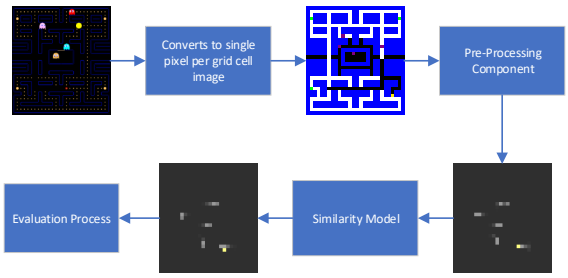

Fig. 1: Pipeline of proposed methodology.

A. Pre-Processing Component

Pre-processing is a phase that controls the format and structure of the input that is passed to the model, prior to training and inference.

1) Non-Augmented Image Sets: Before using the images in the Similarity Model, the images are processed to be a uniform size of w, h. Once resized the images are converted to a grey scale. giving the final dimensions of (w, h, c).

Each set of images is split into sequences of seq images, more sequences can be created by skipping over images with a pre-determined clip stride. On the first run, all the images are taken sequentially and then iterated through again, this time incremented by a clip stride as shown in Fig. 2. This is useful for the learning process, especially if the captured images run at a high frames per second rate, as there will be a lot of identical images.

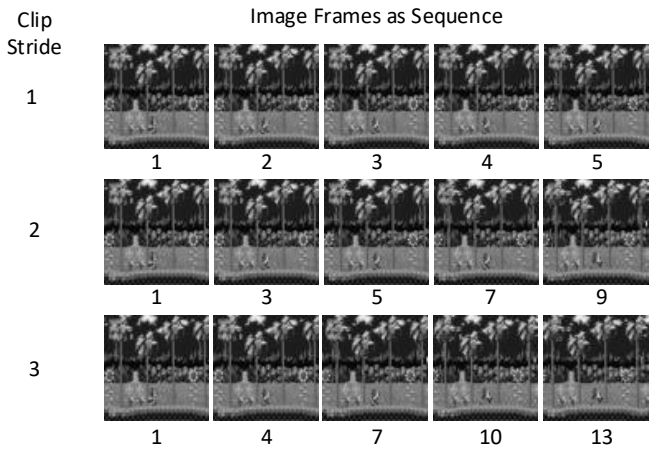

Fig. 2: Example of Clip Stride.

The result process provides an array of sequences, each containing other arrays of seq sequential images, resulting in dimensions of (X, seq, w, h, c).

2) Gradient Image Sets: To potentially improve on the Non-Augmented Image (NAI) Sets, an augmentation of the image can be applied which changes the way the image data is represented. In this example, prior to the images being placed sequentially, each image is converted to a Gradient Image (GI). Important features are extracted from the images, with a focus on the edges within the scene.

To compile the GIs, the Laplacian of the image is calculated by adding up the second x and y derivatives which are calculated using the Sobel operator [12], as seen in equation 1.

$$I = \Delta src = \frac{\partial^2 src}{\partial x^2} + \frac{\partial^2 src}{\partial y^2} \tag{1}$$

Once all images have been converted, they are then split into sequence arrays using the method described within the NAI Sets.

B. Similarity Model

The architecture for the Machine Learning (ML) method, chosen to perform the comparison, is an Auto Encoder (AE). Consisting of two internal models, referred to as the encoder and decoder. The encoder takes in a sequence of input images. The output of which becomes the input for the decoder. A reconstruction of the image sequence, based on the human learnt sequences, is then attempted.

The similarity model uses various layering techniques, including Convolutional 2D, Convolutional 2D Transpose, Time Distribution, Layer Normalisation, and Convolution 2D Long Short Term Memory (LSTM), as presented in Fig. 3. The final layer of the decoder, is a Convolutional 2D with single channel output, this results in dimensions of (seq, w, h, c) which use the Sigmoid activation function. The full breakdown of the architecture is available in table I.

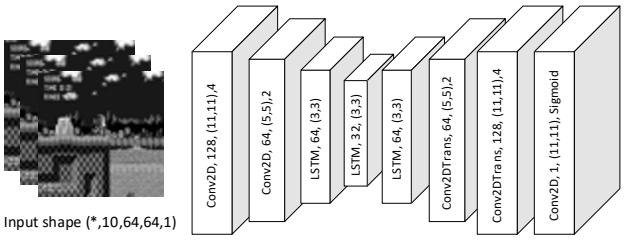

Fig. 3: Similarity model

As previously discussed, the model takes in a sequence of input images that have been through the pre-processing component modifications. The images are then made into training and validation sets. Starting with the second image in the sequence, the regression problem is $x = [i^0, i^1 ..., i^{N-t}]$, where an attempt is made to predict the next frame in the N time instances $Y = [i^t, i^{t+1} ..., i^N]$.

TABLE I: The proposed Auto Encoder architecture for the prediction of similarity in image sequences.

Type	Strides	Activation	Filter
Spatial Encoder			
TimeDist(Conv 2D)	4	TanH	128
Layer Normalization			
TimeDist(Conv 2D)	2	TanH	64
Layer Normalization			
Temporal Encoder			
Conv LSTM 2D	1	TanH	64
Bottleneck			
Conv LSTM 2D	1	TanH	32
Temporal Decoder			
Conv LSTM 2D	1	TanH	64
Spatial Encoder			
TimeDist(Conv 2D Trans)	2	TanH	64
Layer Normalization			
TimeDist(Conv 2D Trans)	4	TanH	128
Layer Normalization			
Output			
TimeDist(Conv 2D)		Sigmoid	1

C. Training Models

The Adam optimiser with a learning rate of 0.0001, is used for training. The loss metric used, as shown in equation 2, is a measurement of reconstruction error is Mean Square Error (MSE):

$$mse = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 \tag{2}$$

Two models can be trained for each type of image input. The process can be repeated for each experiment performed:

- Non-Augmented Image Model
- Gradient Image Model

Each model is trained for a maximum of 50 epochs, or stops early if the loss metric converges for a period of three epochs. The training process occurred on a computing spec of; CPU: Intel i9-10900K, RAM: 64GB, GPU: Nvidia GeForce RTX 2080 Ti 11GB.

D. Evaluation process

The trained model, produces a reconstruction based on the sequence of images received. A regularity score is then calculated using this reconstruction [8]. Within equation 3, t defines the time step, v is the validation sequences over a time series, and r represents the reconstructed sequence; which is the result of the output from the similarity model;

$$e(t) = ||v(t) - r(v(t))||_2 \tag{3}$$

where e provides a reconstruction error as the Euclidean distance between the v frame and the r frame, for all time steps.

$$s_a(t) = \frac{e(t) - e(t)_{\min}}{e(t)_{\max}} \tag{4}$$

Scaling to between 0 and 1 s_a calculates an abnormality score.

$$s_r(t) = 1 - s_a(t) \tag{5}$$

Finally, the regularity score s_r can be calculated by subtracting the result s_a from 1.

IV. Evaluation and Results

This section discusses the games chosen for the experiments, how the datasets were collected and received from the game play, and the level of data samples available for training and testing.

Two experiments were conducted using the chosen games; with two distinct sets of images being passed to the model. The results of all experiments are then presented.

A. Selected Games

Two games were chosen for this experiment. The first game was Pacman (PM), due to its unique AI concepts and simple game mechanics. Each ghost hosted a different distinct personality, as well as different game states. This made it an interesting starting point. During game play, the aim is to navigate the grid while collecting pellets and avoiding ghosts. Control decisions for both ghosts and PM, are typically made and executed at certain nodes within the grid, the only exception to this is if PM changes direction mid-journey. At these nodes, the aim is to determine how likely, or human-like, a given action is.

The second game and level used is Sonic the Hedgehog (StH) Green Hill Zone - Act 1 [13]. This game has more complex visuals and core control mechanics. StH is a 2D platform type game, where StH needs to navigate obstacles, avoid bad objects and collect good objects, until the end of the level is reached.

B. Datasets and Collection Methods

For training and testing, a significant amount of image data is required. Table II shows the spread of image samples collected from both games.

TABLE II: Data Samples Collected

	Training	Testing			
Type	**Human**	**Human**	**AI Agent**	**Basic**	**Total**
PM Sets	104	8	8	8	128
PM Images	70,680	5,335	5,266	5,536	86,817
StH Sets	24	8	8	8	48
StH Images	99,162	21,013	26,198	288,752	435,125

Human data was collected for both games, from multiple participants, using a PC keyboard as the control input. Each set consists of a single level, without any progress to the next level. In the testing sets, with the exception of the basic agent, all single runs are completed to the end of the level.

The AI agent used to collect the data samples, was trained enough to effectively complete the target level. For PM this is a State Machine (SM) AI algorithm, that reacts to the current state of the grid; for example where the closest pellet is and where the ghosts are situated. This allows PM to complete the level. For StH, the NeuroEvolution of Augmenting Topologies (NEAT) algorithm was used to train the AI agent, using the neat-python library [14].

Lastly the basic agent is a part trained AI that can stay alive long enough to collect sufficient data, whilst not being able to complete the level. This provides a very different style of play than when the AI is focused on winning. This set is considerably larger per set, than other agents, due to game play lasting 10 minutes in real time. For testing during the experiments, frames were limited to a maximum of 10,000.

For the PM samples, the full Red, Green and Blue (RGB) image was converted into a single colour pixel per grid space. The colour of the pixel, depends on what game object exists within the corresponding grid cell, at the captured time. This is managed from within the game engine and converted at run-time. This results in a sequence of images at (28,31,3), as shown in Fig. 1.

The StH samples where taken using the Gym Retro library [13]. This allows game play sequences to be recorded and playback data to be recovered. The output of these sequences, provides the full RGB image at (320,224,3), which is used for the second experiment.

C. Experiments

Here we discuss the experiments performed on the two datasets previously presented for training and testing.

All training models were subject to the following parameters; learning rate of 0.0001, decay of 0.00001, an epsilon of 0.000001, and a batch size set to 64. Further to this, for the datasets loaded into memory, the clip stride was set to 5, sequence size set to 10, with the images resized to 64x64 and grey scaled.

The results of the human training sets were calculated, this provided a Baseline (BL) for each of the four models, whilst also validating that no over or under fitting has occurred during the training process. The Human Player, AI agent and Basic agent, testing sets were then evaluated. With the individual scores delimiting the mean average of a complete set.

The expected results for the trained models, was that the human player would be the closest agent to the baseline, followed by the AI agent, with the basic AI agent being the furthest from the baseline.

1) Pacman Sets: The images passed to the pre-processing component, were PNG files, the dimensions were (28,31,3) grid images. The data samples used for training included 104 sets, consisting of 70,680 images, for the PM game.

As shown in the results table III, for the NAIs of PM, the basic agent is highlighted as the most similar to the baseline value, followed by the AI agent, leaving the human players as the least similar.

However, for the GIs set of PM, the results were reversed. With the human player agent being closer to the baseline, followed by the basic agent as the second closest.

TABLE III: Overview Results of Pacman Sets

Set	Baseline	Human	AI Agent	Basic Agent
Non-Augmented Image	0.7824	0.7661	0.7754	0.7782
		0.0163	0.0070	**0.0042**
Gradient Image	0.7898	0.7502	0.7634	0.9676
		0.0218	0.0396	0.0264

(a) Original (b) Reconstruction (c) Target

Fig. 4: Pacman Non-Augmented Image Sample of Testing Images

(a) Original (b) Reconstruction (c) Target

Fig. 5: PM Gradient Image Sample of Testing Images

The issue with the NAI images within the results, is likely related to the number of pixels which do not change from one frame to the next. This makes the model susceptible to learning the layout, rather then identifying the individuals movements within the game grid. Even for training sets with clip stride, the grid remains similar throughout, this can be seen in Fig. 4b when compared closely to 4c. The results for the GI PM, show a similar prediction score.

2) Sonic the Hedgehog Sets: The images passed to the pre-processing component for StH, are in PNG file format; with the dimensions of (320,224,3) images. The data samples used for training, included 24 sets, consisting of 99,162 images for the StH game.

Due to memory constraints, only the first 10,000 frames were used for the StH basic AI testing set.

TABLE IV: Overview Results of Sonic the Hedgehog Sets

Set	Baseline	Human	AI Agent	Basic Agent
Non-Augmented Image	0.6616	0.6635	0.6305	0.6590
		0.0019	0.0311	0.0026
Gradient Image	0.5686	0.5778	0.5479	0.5977
		0.0092	0.0207	0.0291

(a) Original (b) Reconstruction (c) Target

Fig. 6: Sonic the Hedgehog Non-Augmented Image Sample of Testing Images

(a) Original (b) Reconstruction (c) Target

Fig. 7: StH Gradient Image Sample of Testing Images

For the StH test agents, the results for both NAI and GI showed the successful identification of the human players, as being the most similar to the baseline value. However in the NAI set, the basic AI was closer to the baseline then the AI agent. Whilst the GI set, achieved the expected results.

V. Conclusion

This paper aimed to identify a similarity model, that could determine if a set of time series images, were those of human or AI players. It was expected that the human players would be closer to the baseline target, than either a fully trained AI agent, or a part trained basic AI agent.

An Auto Encoder was trained using human played game play sequences, as images only. Two types of pre-processing were used for the training and testing. The first phase converted an image, to a grey scale, which was resized to fit the network input. The images for training were split into smaller sequences. A clip stride to skip x frames was offset, to enlarged the data sets sample size and improved learning outcomes. The second pre-processing phase, involved converting the images into a Gradient Image. This allowed the network to focus only on the edges within the scene.

The testing sets showed promising results; especially when using the Gradient Image augmentation. In the first phase of Non-Augmented Images, only the resized, grey scale images were used. The successful identification of the human test sets, being closest to the human trained baseline, was achieved with the Sonic the Hedgehog sets. However, in the Gradient Image augmentation phase, the Pacman game successfully identified the human player as most similar. Whilst the Sonic the Hedgehog game achieved the expected results, highlighting that the human test sets were the closest to the human baseline, followed by the AI agent, and finally the basic AI agent being the furthest from the baseline.

References

[1] J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel, T. Lillicrap, and D. Silver, "Mastering Atari, Go, chess and shogi by planning with a learned model," *Nature*, vol. 588, no. 7839, pp. 604–609, 12 2020. [Online]. Available: http://dx.doi.org/10.1038/s41586-020-03051-4http://www.nature.com/articles/s41586-020-03051-4

[2] A. Wei, Y. Zhao, and Z. Cai, "A Deep Learning Approach to Web Bot Detection Using Mouse Behavioral Biometrics," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11818 LNCS. Springer International Publishing, 2019, pp. 388–395. [Online]. Available: https://doi.org/10.1007/978-3-030-31456-9

[3] P. Chong, Y. X. M. Tan, J. Guarnizo, Y. Elovici, and A. Binder, "Mouse authentication without the temporal aspect - What does a 2D-CNN learn?" *Proceedings - 2018 IEEE Symposium on Security and Privacy Workshops, SPW 2018*, pp. 15–21, 2018.

[4] A. Serwadda and V. V. Phoha, "Examining a Large Keystroke Biometrics Dataset for Statistical-Attack Openings," *ACM Transactions on Information and System Security*, vol. 16, no. 2, pp. 1–30, 2013.

[5] A. Harilal, F. Toffalini, I. Homoliak, J. Castellanos, J. Guarnizo, S. Mondal, and M. Ochoa, "The Wolf of SUTD (TWOS): A dataset of malicious insider threat behavior based on a gamified competition," *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications*, vol. 9, no. 1, pp. 54–85, 2018.

[6] K. H. Park, E. Lee, and H. K. Kim, "Show Me Your Account: Detecting MMORPG Game Bot Leveraging Financial Analysis with LSTM," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11897 LNCS, pp. 3–13, 2020.

[7] K. T. Chen, A. Liao, H. K. K. Pao, and H. H. Chu, "Game bot detection based on avatar trajectory," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 5309 LNCS, pp. 94–105, 2008.

[8] Y. S. Chong and Y. H. Tay, "Abnormal event detection in videos using spatiotemporal autoencoder," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 10262 LNCS, pp. 189–196, 2017.

[9] M. Hasan, J. Choi, J. Neumann, A. K. Roy-Chowdhury, and L. S. Davis, "Learning temporal regularity in video sequences," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2016-Decem, pp. 733–742, 2016.

[10] A. M. Turing, "Computing Machinery and Intelligence," *Mind*, vol. 59, no. 236, pp. 433–460, 1950.

[11] D. Livingstone, "Turings Test and Believable AI in Games," *ACM Computers in entertainment*, vol. 4, no. 1, pp. 1–13, 2006.

[12] G. M. H. Amer and A. M. Abushaala, "Edge detection methods," *2015 2nd World Symposium on Web Applications and Networking, WSWAN 2015*, pp. 1–7, 2015.

[13] A. Nichol, V. Pfau, C. Hesse, O. Klimov, and J. Schulman, "Gotta learn fast: A new benchmark for generalization in RL," *arXiv*, pp. 1–21, 2018.

[14] A. McIntyre, M. Kallada, C. G. Miguel, and C. F. da Silva, "neat-python," https://github.com/CodeReclaimers/neat-python.