Final Project: CSCI 4511W

Interpreting Fighting Video Game Animations to Generate New Attack Frame Data

Michael Boschwitz

May 8th, 2021

Abstract

The goal of this project was to create and train an algorithm to correctly identify images from fighting games as either Start-up Frames, Active Frames, or Recovery Frames of an attack. The images used for this were frames from the attack animations from "Street Fighter V", "Super Smash Bros. Ultimate", and "Tekken 7". These games are some of the most popular and competitive fighting games currently available, and it was tested on accuracy of classification vs current data from said games. Using TensorFlow, a genetic algorithm was trained to recognize and classify images based on which of the three phases of the frame data it thought it belonged to. The results were then compared to the real frame data to determine accuracy.

1 Introduction

Summary

The problem I will be addressing in this paper revolves around using fighting game animation data and a genetic algorithm to classify and generate frame data. A genetic algorithm was trained on data from two massively popular competitive fighting games; ”Street Fighter V”, and ”Super Smash Bros. Ultimate”. By using the animation and frame data from these existing games I attempted to train the algorithm to recognize and classify the three parts of attack animations for future games. After training the algorithm, I tested its applicability on another popular fighting game ”Tekken 7”. Before diving further, I must quickly elaborate on what ”frame data” means in the context of this paper.

Frame Data

To quickly explain what frame data is, frame data is what makes up the attacks or moves in a video game. For the purposes of this project if can be further explained as it applies to a more niche scope of games. To explain an analogous real life version of how frame data works imagine you are throwing a really strong punch. Before you punch you have a brief period where your body is essentially getting ready to throw it and your muscles are moving to give it more strength. This is the first part of frame data, essentially charging up before doing a move. The middle part of the punch is the part where you can actually damage or hit something with it. In terms of frame data this is the part of the attack where you would actually hit the enemy and your ”hitboxes” would be active. After your punch there is a window of time where you are vulnerable to a counter attack. Since you need to reset after your punch you are briefly vulnerable. this is the last part of frame data, often called recovery frames, as your character needs to recover from doing the attack. Therefore, frame data can usually be defined by three parts; Start-up Frames, Active Frames, and Recovery Frames. The actually names of these are called slightly different things from game to game, but follow the same logic. For the purposes of this project I will be using these names.

Use Cases and Problem Overview

For example, if someone was creating a 2D fighting game in the same vain as ”Street Fighter V”, ”Tekken 7”, ”Super Smash Bros. Ultimate”, et cetera, the algorithm could be used to interpret animation data for the game and generate frame data for the attack. On paper, this would seem a pointless task as this technology solves an incredibly difficult problem that is extremely simple for humans to handle. However, frame data is incredibly vital to fighting games. It is arguably the most important part of not only the competitive viability, but the ability for a fighting game to gain a following at all. This fact is demonstrated by what games are played in world class e-sports tournaments like EVO. EVO could be considered by many as the fighting game equivalent to a Baseball World Series. Having your game played at EVO means it is competitively viable enough that people are willing to create massive prize pools. Granted, as of 2021 there has been some drama surrounding the inclusion of certain games, however, this primarily caused by the developers and publishers not the games themselves. Regardless, there are two examples from EVO that perfectly demonstrate the importance of good frame data.

Firstly, is the omission of the game ”Jump Force”. It was released onto the fighting game scene in 2018 to moderate acclaim and notably included fighters from incredibly popular Japanese Shōnen Jump Anime and Manga series. However, the following year it was not played at EVO, but ”Dragon Ball FighterZ” was. Ironically, characters from ”Dragon Ball FighterZ” Appear in ”Jump Force” and they are even published by the same company. While ”Dragon Ball FighterZ” was praised for its gameplay and mechanical depth, ”Jump Force” was criticized for it. As a result, this is likely why the game was omitted from tournament play at EVO, despite announcements of new characters happening at the event in 2019.

Secondly, to demonstrate why great frame data is vital I would like to highlight a moment from EVO 2004. Dubbed ”Offical EVO Moment 37” by the tournament organizers (Webb (2019)), it refers to a moment during a ”Street Fighter III: 3rd Strike” semifinal match between Daigo Umehara and Justin Wong. During the first round Justin had brought Daigo’s health down to point where one hit would win Justin the round. While trying to win the match

Justin goes on the offensive, however, Daigo is prepared. Daigo blocks his attack with a parry, a move in the game that blocks incoming damage, to stop the attack. Incredibly, Daigo manages to parry fifteen kicks in a row! Daigo has about 1/10 of a second to perform this for each move. After the fifteenth consecutive parry, Justin switches his attack pattern. This ultimately leads to a punishment from Daigo, bringing Justin from about half health to no health. Daigo ultimately wins the round with what is probably the lowest possible health value possible before you would be K.O.'d. This moment perfectly highlights the importance of great frame data. If the frame data for characters, or even the parry was imperfect, this moment of pure skill and raw talent is no longer possible.

I genuinely believe this algorithm could have potential use cases. If properly trained on better raw data, it could have great applications for developers of fighting games. Likewise, if the accuracy is raised to a comparable level, more in-depth analysis could lead to more incredible moments at tournaments. It could also increase the baseline of future competitive games, allowing for a wider variety of games being considered competitively or tournament viable. This may allow developers to spend development time in different ways, and speed up QA time for projects. To help solve this problem, I first have the algorithm be able to interpret incoming fighting game animation data. Next the algorithm can train to accurately estimate the frame data of the action and output the results.

Partners: None.

2 Literature Review

Introduction

The two key components of this project pose an interesting issue. In essence, I am using two genetic algorithms to solve parts of the problem. One is used for image processing, the other is for pattern recognition. That is roughly how the algorithm will be composed, so understanding how to improve one likely can improve the other. In fact, it may be worth examining whether or not the image processing issue can be solved in other ways. Fortunately, examining methods like Principal Components Algorithm (PCA), Interpersonal Image Difference Classifier (IIDC), and Elastic Bunch Graph Matching (EBGM) has been done before. However, as previously mentioned there is a second genetic algorithm involved. This algorithm is responsible for generating viable frame data. While streamlining the image processing is greatly beneficial to the overall efficiency of this project, it is only useful to the initial issue of image recognition. The goal of the project is primarily accuracy based. As such, the pattern recognition aspect is primarily focused on avoiding issues like infinite combos. Therefore, research into this will be more focused on AI looking for infinite combos in games, competitive fighting game balance, dynamic AI opponents, etc. With inherit difficulty stemming from the definition and goal of "Good Frame Data", this needs to be defined. Examining what makes good or bad frame data will greatly help with this.

Genetic Algorithms

Before we move into the average results of comparing image recognition algorithms, it is important to define this term that is frequently used. A Genetic Algorithm is based on a similar concept to evolution. It is an adaptive heuristic search algorithm that looks at what may evolutionary be called fitness. To put it plainly in an analogy, it creates a bunch of bots that try to solve a problem. Those bots are tested to solve a problem and often start by randomly guessing. Those with the highest accuracy get to move on and "evolve". Over time these bots slowly adapt to overcome the problem similar to how fauna on earth evolve and adapt due to natural selection.

Image Recognition Algorithm Research

The most popular use in the public zeitgeist is currently focused on Facial Recognition. iPhones feature a security feature called FaceID, DeepFakes are being used as a tool to create comedic videos as well as for deeply disturbing videos, Google can now create folders based on faces in your stored photos. A great overview of these was done on various image recognition applications and found a common trend. In examining some of the applications the performance of the task can vary widely (Barrett (1997)). Of the use cases tested, simple verification of objects performed much better than facial recognition (Barrett (1997)). One of the biggest issues that causes this is when image recognition runs into is the large number of co-variate factors such as age and gender (Mahmud et al. (2014)).

Researchers have attempted to solve for this problem in various ways; Combining Neural Networks with Genetic Algorithms, Modifications on PCA, et cetera. While PCA is still the de facto best solution (Mahmud et al. (2014)), it is by no means the only one being studied. As of writing there is no "Perfect Algorithm" publicly known with 100% accuracy. Intuitively that would make sense, but in my research the highest I have found is rarely even in the 90% accuracy range (Givens et al. (2004)). On a brief aside, I seriously doubt the existence of one maybe even ever. The United States Government can not even make one. A paper on the topic was able to hit an average of about 90% accuracy, but could barely even gain an additional 1% in accuracy when increasing the population size from 160000 to 320000 (Grother et al. (2010)). But I digress, variations in the algorithms themselves often comes down to two trade offs—computational time versus better recognition rate (Givens et al. (2004)).

Looking at some of the comparisons between algorithms, it becomes apparent that improvements between them are really transient and nebulous. Studying the differences between PCA and IIDC found that IIDC was better at recognizing Asian faces, however, PCA was better at identifying people with glasses and closed eyes (Givens et al. (2004)). Ultimately, they seem to equal out in being better or worse at one thing or the other. Likewise, with PCA and EBGM the racial complexion created some variation in the results, with both algorithms sometimes being better or somethings performing worse on non-white subjects (Givens et al. (2004)). The only real advantage that seemed to be found between PCA, EBGM, and IIDC was the amount of training required. EBGM does not have a training phase, while both PCA and IIDC do require training data. All in all, I think PCA would still technically be the best. Given it's prevalence, I think the likelihood of it being easier to apply combined with the low statistical and empirical advantage from similar algorithms results in it being the most useful for this project.

Examining/Analyzing Good and Bad Frame Data

A similar but completely different problem was addressed by Deep Blue and AlphaGo. In solving how to evaluate this problem you could say each frame would have relative accuracy. In this case it would be which part of the three phases of the attack would it be. Is this frame a start-up frame, active frame, or recovery frame? I think it may be appropriate to quickly compare this problem to some chess algorithms. We can evaluate some of the techniques employed by previous chess algorithms to potentially learn how these algorithms make high accuracy evaluations of information. Looking into this, the main method deployed was based on a general reinforcement learning algorithm (Silver et al. (2017)). Some of the previous methods used by other AI—namely Monte Carlo Tree Search (MCTS) and Alpha-Beta (Which was notably used by Deep Blue)—have success when it comes to properly evaluating a task. However, AlphaGo Zero is not only more accurate, but performed better than the leading two AI in chess and shogi by around 5k-10k more evaluations(Silver et al. (2017)). Another paper on research done in the way of pattern recognition and classification using a genetic algorithm shows how the algorithm may have created results that were in fact better classifications than originally anticipated (Khoo and Suganthan (2003)). For the purposes of my project, I am likely to explore more of the features highlighted in the paper to avoid this. Since frame data is mostly based on how it feels in movement, several of my test and training methods rely on a level of currently available data. As such I would like to avoid finding "new and better" frame data for preexisting data, and rather have the algorithm create relatively good frame data. This may seem counter-productive, but a consideration into what makes good frame data is human ability.

An important part of fighting game balance comes as a result of every playable character in the game. It is not uncommon for a character to generally have a bad match up against a another character, but if certain characters are just statistically better in every way then you have an issue. Notorious examples of this issue normally manifest as certain characters being outright banned from competitive play, meaning they are so strong that they are not allowed to be played in tournaments. This is why we are not trying to create the best possible frame data. In a similar vain, game balancing for humans has actually been an area of research that has previously been explored. One of the most notable examples used a Hidden Markov Model to attempt to find infinite combos in fighting games (Zuin and Macedo (2015)). The results indicated that on a small scale an algorithm could learn to find smaller combos, but struggled to find longer combos (Zuin and Macedo (2015)). One of the key issues with the algorithm, however, was that it became over specialized (Zuin and Macedo (2015)). Other similar solutions tried using genetic algorithms, reinforcement learning and Monte Carlo tree searches to create an AI player capable of simply playing the game (Thuan et al. (2019)). Ultimately, the solution was thought to be a hybrid of those algorithms (Thuan et al. (2019)).

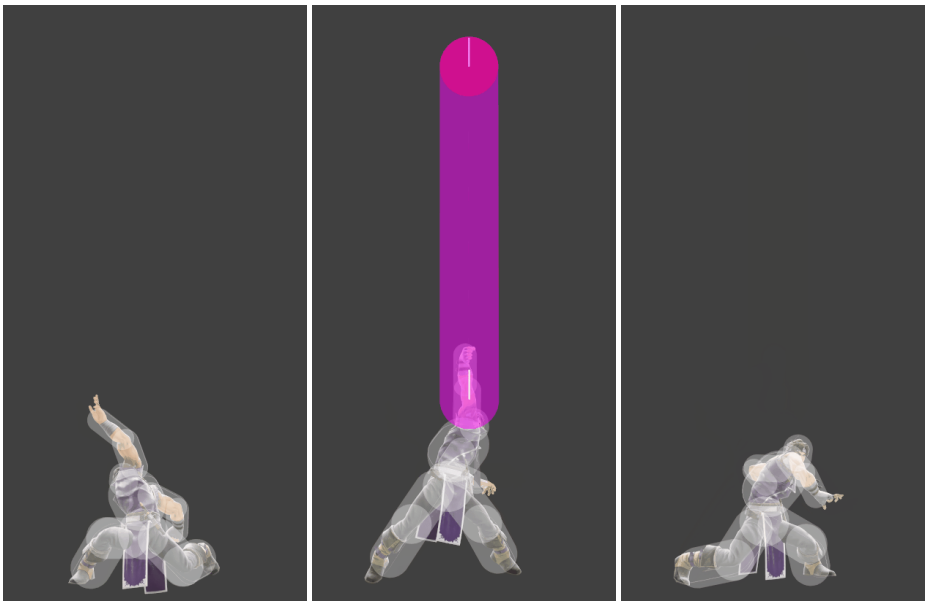
Literature Review Conclusion

Given this information we can see a bit of a trend in similar problems. Initially they may start with algorithms like Monte Carlo tree searches, but as time has passed genetic algorithms seem to have overall given better results. For image recognition this has resulted in PCA having a prevalence. In Analysis and evaluation, we can see the shift from MTCS to genetic algorithms from Deep Blue to Alpha Go Zero. In the end this leads me to the conclusion that using a genetic algorithm, or a variation of one, will yield the best results.

3 Approach

Methodology

Using resources and projects like *fullmeter.com* and *ultimateframedata.com*, I was able to compile excellent data of highly popular fighting games frame data. Likewise, training a genetic algorithm on predetermined fighting tournament viable data allows for the algorithm to be trained on extremely visually diverse characters. For training purposes the algorithm was trained only 2D video/animation data. I would have liked to incorporate a way to train it for 3D fighting games like "Soul Caliber 6", but figured I limit my ambitions slightly. Afterwards, it was then challenged to generate new framer data based on data it was fed. The images below demonstrates the average look of the data it will be trained and used on.



(From *Super Smash Bros' Ultimate*)

Each of these images represents a phase of the attack's frame data. From left to right they show a frame from the Start-up Frames, Active Frames, and Recovery Frames from Richter's attack respectively. The algorithm will be trained to correctly identify which phase of the attack each one of these images would fall into, and classify the frame as either a Start-up Frames, Active Frames, or Recovery Frames.

Tools

The primary tool I used for this project is TensorFlow. TensorFlow is a robust machine learning platform owned by Google. TensorFlow gave a great baseline for solving the image recognition aspect of the problem, and the simplicity of library made coding in python relatively painless. What was not painless was how I converted this videos. In combination with TensorFlow, I spent several hours in Adobe Photoshop 2021. I had to use Photoshop to convert .gif's and .mp4's into .png's for the algorithm. At the time and in retrospect I know this was a horribly inefficient way to convert .gif's to folders of .png's, but I did it anyway because I think there is something wrong

with my brain. I seriously wish I was kidding when I say that I probably wasted around 10-20 hours of my limited human existence manually converting .gif's and .mp4's into .png's. Dear god, it was it not worth it.

4 Experiment Design

Benchmarking

Given the goal of the project, runtime, memory, etc. will not be a good metric for analyzing the results. Instead, the easiest way would be to examine the quality of the results based on the current data of the games. Once the algorithm is trained, we can compare the output of the algorithm to the actual frame data of the games. After reaching a relatively high accuracy on those games, it was then tested on other games. By asking it to blindly try on another game, we can insure it is not just memorizing the results. For a game it was not trained on, we would expect to see around 33% accuracy if it was in-fact blindly guessing. That in combination with a high accuracy in during training would likely imply that it has not learnt how to solve the problem, and may have just memorized the results. Should it score higher than 33% on the blind testing, it likely would imply it has at least partially grasped been able to learn to correctly identify frames. As a result the former, more naive benchmarking method gives a good baseline to gauge the algorithms initial accuracy. Meanwhile, the latter more precisely tests how accurate the algorithm is at making estimates on new data. Likewise, it ensures that it is actually able to generate new data, and is not simply memorizing current frame data. I quantified the results using the percent error formula shown below:

$$\%\ \text{error} = \left|\frac{\#\,\text{experimental} - \#\,\text{actual}}{\#\,\text{actual}}\right| \times 100$$

For most of the second step of benchmarking, anything below or around a 33% can be interpreted as blind guessing. This is due to there being a 1/3 chance of correctly classifying images by randomly guessing. If the bot is actually able to generate new frame data it will need to be much higher than 33% when blindly tested on new data.

Timeline

Goal One: Create an Algorithm that can interpret the frame data using the aforementioned API's and software.
Goal Two: Have The Algorithm Generate what I think is the correct data, and export it's expected results.

Week of 4/11: Studied and Tested viable API's and Software and determined TensorFlow as one of the best options. Compile data set of frame data and attack animations from "Street Fighter V", "Super Smash Bros. Ultimate".

Week of 4/18: Planned on coding, but the majority of hours were spent converting files to .pngs. Set up a TensorFlow test algorithm to dip my feet into using it.

Week of 4/25: Started work on project algorithm and began to try and train it.

Weekdays of 5/2: Kept revising my spaghetti code and trying to train as much as possible. Encountered weird bug resulting from how my .pngs were saved and had to restructure my testing data. Started getting final results of training, then my computer kept crashing while doing tests. Apparently, at some point my monkey brain thought it would be a good idea to export the results. Exporting the results was fine, but I also essentially created around a terabyte of copys of .pngs files after I changed how data was stored.

Weekend of 5/2: Feel mildly depressed at how awful the results were. The second phase of the testing for the algorithm against new data is now completely redundant, but I performed it anyway. Compiled all my results and finished drafting the project paper.

Explaining the Timeline

There was an incredibly naive timeline section of my project when I initially started to create a draft of this paper. I replaced that as I went along which lead to the current included timeline. The included timeline is a quick glimpse into how my time was spent on this project. As you can likely tell by the last few entries in the list, things did not go according to plan. I managed to complete goal one using the TensorFlow library, but goal two was a completely different can of worms. Here's a highlight of some of the bugs I encountered while attempted to complete goal two.

- Bug: Training on the wrong data (It got very good at identifying pictures with flowers in them).
- Bug: It would only train on Banjo and Kazooie's attacks... and it still didn't get them right!
- Bug: Algorithm couldn't properly travel though files where the data was stored so I created a function to copy the data making it easier to access (Foreshadowing).
- Bug: Ubuntu/Python crashing mid test/training... Then my OS and computer started crashing!
- Bug: Algorithm would create a copy of training set so it could properly go through it all... It then never deleted the copy and my computer started crashing because it would generate 10+ gigabytes of data every time it ran.

In the end there was an algorithm that was able to identify and classify data like I wanted. It was just incredibly inaccurate. In my hubris I severely overestimated the viability of finishing such a project in such a short amount of time, but I chalk this mostly up to my inexperience.

5 Results and Analysis

To reiterate a point I brought up earlier, using the percentage error formula we should expect an accuracy of about 33% if the algorithm is simply randomly guessing. It might be obvious why this number is being reiterated here.

Super Smash Bros. Ultimate Results:		
	Loss	Accuracy
Round 1	0.4946	0.3158
Round 2	0.4857	0.3169
Round 3	0.4779	0.3225
Round 4	0.4642	0.3263
Average	0.4807	0.3204

Street Fighter V Results:		
	Loss	Accuracy
Round 1	0.4898	0.3542
Round 2	0.4852	0.3644
Round 3	0.4825	0.3635
Round 4	0.4776	0.3683
Average	0.4838	0.3603

Tekken 7 Results:		
	Loss	Accuracy
Round 1	0.5762	0.2827
Round 2	0.5523	0.2953
Round 3	0.5496	0.3013
Round 4	0.5144	0.3024
Average	0.5482	0.3008

Total Average Loss $\approx 50.42\%$
Total Average Accuracy $\approx 32.72\%$

Percent Error

As previously stated, I used the percent error formula to help gauge the overall accuracy of the results. I was hoping for experimental value to be significantly higher, but reality is often disappointing. Plugging it into the equation results in this.

$$1.8\% = \left| \frac{0.3272 - 0.3333}{0.3333} \right| \times 100$$

This percent error tells us how big errors in the experiment have effected the results. A small percent errors means that we are close to the accepted value, and or results have not been eaaffected much by error. For example, a value less than 5% likely means your results are relatively accurate, while a value of 50% would imply there is serious

issues with your results. The percent error for this experiment is around 2%, which is good. However, this is when we are comparing it to the results of random guessing. A simpler way to put it would be that there is around a 2% chance the algorithm was not just randomly guessing. This is seriously not a good sign, but the data of the final test rounds does seem to support this.

Understanding Results

Comparing the results from Table 1 and Table 2 there is a weird discrepancy. It would see they should all have a relatively similar accuracy percentage, but Table 2 is about 4% higher than Table 1. This is likely caused by the amount of data that the algorithm was able to be tested on. "Super Smash Bros. Ultimate" has 79 unique fighters (80 if you count Pyra and Myrtha as different fighters), whereas "Street Fighter V" only has around 40 fighters. Therefore, I would probably have to chalk this discrepancy up to a lack of balance between the data. I believe the results of the "Street Fighter V" test are slightly higher as the bot was a bit luckier, and with less tests to perform overall, it essentially did not average out like the "Super Smash Bros. Ultimate" ones did. The results from "Tekken 7" tests in Table 3 are about what I predicted when I was nearing the end of this project. It initially seems to actually be trying to correctly identify frames, it's just not very good at it. As it does this it trends up towards the 33% threshold, likely due to algorithm randomly guessed on top of whatever it was basing digestions off of being more effective.

It's very difficult to understand what the algorithm was really doing by the end of this project, but by my estimate it was likely just randomly guessing. The results from table 1 and 2 along with the calculated percent error seem to confirm this as well. However, the weirdly low initial accuracy from Table 3 gives me a bit of hope. In the end it did average back out to around 33%, but it having such an oddly low accuracy implies it may not have been randomly guessing exclusively. "Tekken 7" has around 50 fighters, but I was only fed the data for 40 of them. The results should have been more similar to "Street Fighter V", but they are significantly worse.

Light Speculation: Drawing a bit of a optimistic conclusion from this, it seems like about 50% of the bots may have been trying to actually solve the problem, but 50% would just guess randomly. This lines up with the loss percentages of most of the tables, but I think especially on "Tekken 7". The algorithm was trained to the best of my ability to that point, so I could see it argued that about half of the bots randomly guessed and half of them were trying. Furthermore, a quick sidenote on the loss percentages: I have absolutely no clue how accurate those are. I am using them as evidence in the more speculative analysis of my findings because I could not tell you were those number were being generated from. There's a high likelihood it may simply be irrelevant inherited code from a TensorFlow tutorial. However, I felt I would at least highlight something that stood out to me as being a plausible explanation for what the algorithm might have been doing.

6 Conclusion

To put it plainly, more time is needed. It is my belief that the algorithm is currently starting to learn how to properly sort frames of the animations, however, not enough time was spent allowing it to do so. Likewise, this belief is based on some light speculation. In conclusion, I believe that this algorithm will eventually work. I may even attempt to train it further. However, in its current state it shows nothing more than potential. The statistical likelihood that is currently basing its classification on random guessing is relatively high, and as a result I was unable to train it to a point where it would no longer be encouraged to do so. Do I believe it could have been achieved, yes, but I do not think I could have more effectively been able to work on this algorithm without the knowledge I know now. Critical mistakes were made, and while they did not directly effect the results, they did take up significant amounts of time to address.

While not the results I had hoped for, it is nonetheless greatly illuminating into my understanding of AI development. I wish I could have made a better genetic algorithm, but circumstances have left me with an algorithm/bot that can at least guess and be able to classify frames. I suppose that in itself is an achievement, the issue is that it's just not very good at the second part.

References

Barrett, W. (1997). A survey of face recognition algorithms and testing results. In Conference Record of the Thirty-First Asilomar Conference on Signals, Systems and Computers (Cat. No.97CB36136), volume 1, pages 301–305 vol.1.

Givens, G., Beveridge, J., Draper, B., Grother, P., and Phillips, P. (2004). How features of the human face affect recognition: a statistical comparison of three face recognition algorithms. In Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004., volume 2, pages II–II.

Grother, P., Quinn, G., and Phillips, P. (2010). Report on the evaluation of 2d still-image face recognition algorithms.

Khoo, K. G. and Suganthan, P. N. (2003). Structural pattern recognition using genetic algorithms with specialized operators. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 33(1):156–165.

Mahmud, F., Haque, M. E., Zuhori, S. T., and Pal, B. (2014). Human face recognition using pca based genetic algorithm. In 2014 International Conference on Electrical Engineering and Information Communication Technology, pages 1–5.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T. P., Simonyan, K., and Hassabis, D. (2017). Mastering chess and shogi by self-play with a general reinforcement learning algorithm. CoRR, abs/1712.01815.

Thuan, L. G., Logofătu, D., and Badică, C. (2019). A hybrid approach for the fighting game ai challenge: Balancing case analysis and monte carlo tree search for the ultimate performance in unknown environment. In Macintyre, J., Iliadis, L., Maglogiannis, I., and Jayne, C., editors, Engineering Applications of Neural Networks, pages 139–150, Cham. Springer International Publishing.

Webb, K. (2019). Watch the most iconic moment in street fighter history from a brand new angle.

Zuin, G. L. and Macedo, Y. P. A. (2015). Attempting to discover infinite combos in fighting games using hidden markov models. In 2015 14th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames), pages 80–88.