**BACHELOR’S THESIS**

**TETRIS NEURAL NET PLAYING IN THE NINTENDO SWITCH**

**Joan Dot Sastre**

**Degree in informatic engineering**

**Faculty of engineering**

**Academic Year 2020-21**

Key words:

word 1, word 2, word 3, …

*Thesis Supervisor’s Name Jose María Buades*

*Tutor’s Name (if applicable) Jose María Buades*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| The University is hereby authorized to include this project in its institutional repository for its open consultation and online dissemination, for academic and research purposes only. | Author | | Supervisor | |
| Yes | No | Yes | No |
|  |  |  |  |

**Abstract**

Contenido

[1. Introduction 4](#_Toc73038526)

[1.1. Project summary 4](#_Toc73038527)

[1.2. Artificial intelligence in videogames 4](#_Toc73038528)

[1.3 Objectives 5](#_Toc73038529)

[2. System 6](#_Toc73038530)

[3. Switch-PC interface 6](#_Toc73038531)

[4. Information capture 6](#_Toc73038532)

[5. Deep learning module 6](#_Toc73038533)

[6. Decision making 6](#_Toc73038534)

[7. Results 7](#_Toc73038535)

[8. Conclusions 7](#_Toc73038536)

1. Introduction

1.1. Project summary

The basis of this project consists in achieving an AI capable of playing the game “Tetris 99” in the console known as “Switch”, manufactured by the famous game company Nintendo. In order to reach our goal, we have to tackle the problems one by one. Thus, the means by which the results in the project have been obtained consists of dividing it into four different modules:

* Switch-PC interface: The way in which the pc is able to communicate with the console.
* Information capture: How the console’s information is sent to the pc and then processed for use by the neural net.
* Machine learning: How the AI was able to learn. Includes the training environment explanation, the heuristic used and how it was chosen.
* Decision making: Defines how the information extracted by the information capture module is treated right before it is finally ready to be sent to the net. It also explains how the output is adapted and transferred to the console correctly.

The aforementioned modules will be further explained later, in their corresponding section in the document.

1.2. Artificial intelligence in videogames

Artificial intelligence has been present in videogames since the very beginning.

Its purpose has always been to improve the players experience and the methods that have been used to implement such behaviours are vast, ranging from finite state machines and increasingly more complex enemy movement patterns tied to the game difficulty/level, to combining different advanced methods like pathfinding and decision trees. Other techniques related to machine learning such as reinforcement learning can also currently be found in some games. All these methods are mostly used for NPCs (non-playable characters) and the information they perceive from the environment can be given in two different ways, via sensors, which provide a limited vision of the game world, or via the game’s own stored information e.g., the player’s exact location.

(<https://en.wikipedia.org/wiki/Artificial_intelligence_in_video_games>)

Due to an increasing interest in artificial intelligence in recent years, people have started to try and beat their favourite games with it. When taking this approach, we must first consider how the agent\* is going to perceive the game, having the same two options we talked about before. This time we usually encounter a major inconvenience, we do not have direct access to the game information due to us not being the game developers, although thanks to some APIs (such as OpenAI Gym) we can access the game and thus base our agent’s information on it. Unfortunately, those APIs mostly feature older games, which limits us to the ones provided by it. Hence comes the need for image processing tools to extract data, though this may not necessarily be done by us, as will be shown later.

Once we have discussed about how data can be collected, we can introduce the next step, agent building. Due to videogames, many different methods have arisen, and with the increasing difficulty of the games beaten has also come an increase in agent intricacy, leading to the drop of simpler techniques in favour of reinforcement learning, which ended up performing much better in highly complex environments.

Due to the increasingly more difficult games being beaten has also come a need for more intricate agents, leading to the drop of simpler techniques in favour of reinforcement learning (many times paired with those old techniques in order to provide the agent with basic behavioural guidance), which has ended up providing much better results in highly complex environments.

1.3 Objectives

The overall goal of the project and how it can be achieved has already been discussed but what will be called a success has not yet been defined.

Building an AI capable of playing Tetris has already been done many times before with great success, though the challenge trying to be taken has a few major and minor hindrances.

First of all, as a minor inconvenient, the Tetris version we are building our AI on features de SRS (Standard Rotation System) \*, which is a modern rotation system with some unconventional situational rotations. No implementation that can be used has been found so an entire game replicating Tetris 99 must be built from scratch to train our model.

Secondly, there is not a standardized way to access the consoles input from the PC, so a reliable workaround must be built and adapted. This is probably the biggest setback.

Lastly, and as a result of having to intercommunicate both devices, some extra delay, that we hope will not heavily interfere, will occur when bringing everything together.

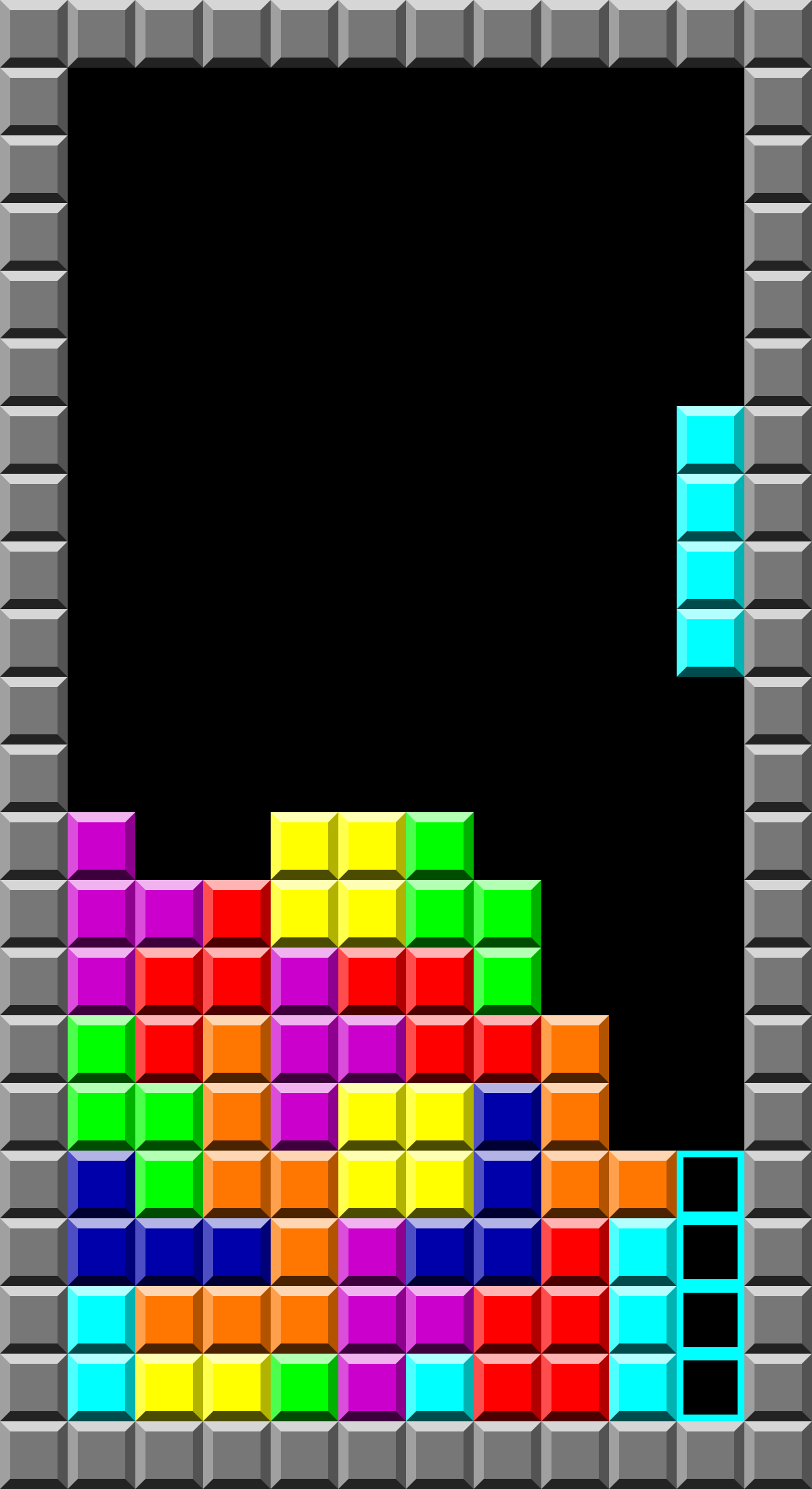
Taking this into account, we expect our agent to work swiftly and, if we manage to pass the neural net’s output accurately to the console, be able to perform well when under low gravity.

2. Tetris 99 and game system

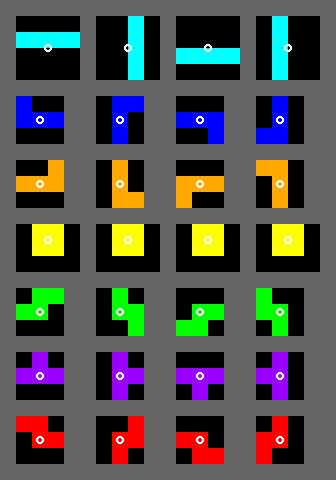
Tetris is a long running game series that has been going on since 1984, when Alexey Pajitnov invented it. Ever since it was created, many iterations of the game have been made, with each one of those somewhat altering the rules or adding new mechanics to spice things up. As previously mentioned, the project is being made under the Tetris 99 version, which implements the SRS.

2.1 Game basics

As many people already know, Tetris is a puzzle game consisting in trying to stack pieces up and clear lines on a 10x20 grid. Whenever a line is filled to its maximum capacity it gets cleared and the blocks above it drop as many lines as were cleared, four lines for the following case:



There is a total of 7 different pieces, each on of those having an associated colour that is usually maintained through all Tetris versions. Their names are I, J, L O, S, T and Z and they look as follows:



As we can see in the image above, each piece has four different orientations which can be accessed sequentially back and forth in the order shown, the small circle indicating the axis the piece rotates in.

The I and O pieces are a special case given that they do not use an actual block as their anchor point to rotate, making the first one shift one block up or down depending on the current position and the second one not rotate at all.

Now that we know their shapes, we see that the maximum number of lines that can be cleared at once is four. This is important because the score we obtain does not increase linearly, netting us higher scores the more lines we clear in one go.

The actual formula which determines how many points we get is:

|  |  |
| --- | --- |
| Single | 100 × level difficulty |
| Double | 300 × level difficulty |
| Triple | 500 × level difficulty |
| Tetris (Quadruple) | 800 × level difficulty |

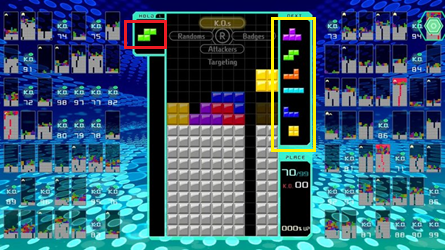
More ways of obtaining points are soft drops (moving the piece down one cell), hard drops (letting the piece fall to the bottom) and combos (chaining line clears with different pieces):

|  |  |
| --- | --- |
| Combo | 50 × combo count × level |
| [Soft drop](https://tetris.wiki/Drop#Soft_drop) | 1 per cell |
| [Hard drop](https://tetris.wiki/Drop#Hard_drop) | 2 per cell |

Finally, there is T-spins, which is a mechanic that will be spoken about at the end of this block once all the information surrounding the SRS has been laid out.

2.2 UI and specifics

An actual game of Tetris 99 will look like this:



There are many things that need to be analyzed to fully understand all the game aspects of this game version.

The first thing that should be mentioned is that we can see the upcoming 6 pieces that will have to be placed on the board (highlighted by the yellow square). Those are chosen from a bag containing all seven pieces, therefore making the game much more predictable as luck will not be impacting the game as much as with a fully random selector.

Then, there is the piece storage block (encapsulated by the red square), which allows us to save the current piece and draw the next one or, if there already is a stored one, to swap it out.

Last and more importantly, the background shows many more smaller Tetris boards. Those boards belong to other players who are playing concurrently and against each other, including you. The name Tetris 99 comes from this particularity, being that the main game mode is an online competition to see who wins out of 99 players.

- Explicar el juego Tetris y el sistema de movimiento de piezas universal (tiene un nombre pero no me acuerdo)

- Visión general del sistema, compuesto por 4 módulos

\* Interfaz ordenador - Switch

\* Sistema de captura de información

\* Aprendizaje previo del sistema de toma de decisiones

\* Sistema de toma/aplicación de decisiones

3. Switch-PC interface

Aquí hay que explicar el emulador del mando hecho con Arduino, recibe comandos del ordenador y se los pasa a la Switch

Aunque no lo hayas hecho se puede describir su funcionamiento.

4. Information capture

OpenCV, cómo se detecta la escena, como se "lee" la pantalla

5. Deep learning module

Aquí hablar del juego Tetris implementado, y que la red aprende sobre ello

6. Decision making

Explicar que este módulo recoge la información del módulo de captura de información, la procesa con el sistema generado en el

módulo de aprendizaje, encola las órdendes, y las manda emulador del mando

7. Results

Pequeña estadística de partidas jugadas y resultados obtenidos

8. Conclusions