Um Modelo de Aprendizagem Profunda para a classificação multi-instância, multi-rótulos em vídeos de jogos digitais

Etienne da Silva Julia

Universidade Federal de Uberlândia
Faculdade de Computação
Programa de Pós-Graduação em Ciência da Computação

Uberlândia
2022

Etienne da Silva Julia

Um Modelo de Aprendizagem Profunda para a classificação multi-instância, multi-rótulos em vídeos de jogos digitais

Dissertação de mestrado apresentada ao Programa de Pós-graduação da Faculdade de Computação da Universidade Federal de Uberlândia como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

Área de concentração: Inteligência Artificial

Orientador: Marcelo Zanchetta do Nascimento

Uberlândia
2022

Ficha Catalográfica Online do Sistema de Bibliotecas da UFU
com dados informados pelo(a) próprio(a) autor(a).

J94
2022

Julia, Etienne da Silva, 1996-
A deep learning system to perform Multi-Instance Multi-Label event classification in video game footage [recurso eletrônico] / Etienne da Silva Julia. - 2022.

Orientador: Marcelo Zanchetta do Nascimento.
Dissertação (Mestrado) - Universidade Federal de Uberlândia, Pós-graduação em Ciência da Computação.
Modo de acesso: Internet.
Disponível em: http://doi.org/10.14393/ufu.di.2022.562
Inclui bibliografia.
Inclui ilustrações.

1. Computação. I. Nascimento, Marcelo Zanchetta do, 1976-, (Orient.). II. Universidade Federal de Uberlândia. Pós-graduação em Ciência da Computação. III. Título.

CDU: 681.3

Bibliotecários responsáveis pela estrutura de acordo com o AACR2:
Gizele Cristine Nunes do Couto - CRB6/2091
Nelson Marcos Ferreira - CRB6/3074

Eu dedico este trabalho à minha família, meu orientador e meus amigos, todos que, das mais diferentes maneiras, ajudaram a torná-lo realidade.

Agradecimentos

Eu gostaria de agradecer ao meu orientador, o professor Marcelo Zanchetta do Nascimento, que, com muita sabedoria e paciência, me ajudou a superar inúmeros desafios, tanto profissionais, quanto pessoais. Eu pude contar com sua orientação e seu apoio ao longo de toda essa jornada e eu não poderia ter escolhido alguém melhor para me acompanhar nela.

A minha família, que nunca parou de me motivar a ser forte.

A meu colega e amigo, Matheus Prandini, cujo caráter e ética de trabalho sempre foram uma inspiração para mim al longo da minha carreira acadêmica.

Ao meu grupo de pesquisa, que me acompanharam ao longo dos últimos dois anos.

Gostaria de agradecer à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior -(CAPES)- Pelo apoio financeiro e ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) (Projeto 11404/2021-9) e à Fundação de Amparo à Pesquisa do Estado de Minas Gerais (FAPEMIG) (Projeto APQ-00578-18).

Por fim, gostaria de agradecer à Universidade Federal de Uberlândia (UFU), que foi uma segunda casa para mim nos últimos 7 anos, onde aprendi muito, fiz amigos para o resto da vida e descobri muito sobre mim mesmo e sobre o papel que eu gostaria de exercer nessa mundo.

"My thanks, Ashen One. With this will I paint a world... Twill be a cold, dark, and very gentle place. And one day, it will make someone a goodly home."

(The Painter)

Resumo

Jogos digitais, além de serem um setor da indústria de entretenimento extremamente relevante, são muito utilizados como objetos de estudo na inteligência artificial, uma vez que representam um cenário de alta complexidade. Nesses estudos, destaca-se a investigação de abordagens para capacitar agentes jogadores com a habilidade de recuperar informações relevantes, já que isso é muito útil para maximizar a capacidade de aprendizagem desses agentes. Esse trabalho é dividido em duas partes, a primeira propõe e analiza novos modelos profundos de aprendizagem para identificar eventos em vídeos do jogo Super Mario Bros. Esses modelos são compostos por uma rede neural convolucional (CNN), responsável pela extração de características, e uma rede neural artificial (NN) para a classificação. O objetivo da CNN é produzir uma nova representação para as cenas de jogo que maximize a performance da rede classificadora na tarefa de identificar eventos de jogo. A principal contribuiçaõ dessa primeira parte é a demonstração de uma performance superior obtida por modelos que utilizam de uma representação dos dados por *chunks* combinados com os recursos de uma rede neural recorrente (RNN) para a classificação. A segunda parte apresenta dois modelos de aprendizagem profunda (DL) desenvolvidos para tratar com a classificação de eventos multi-instâncias multi-rótulos (MIML) em vídeos de jogo. A arquitetura desses modelos é baseada em um script para a geração de dados, em uma rede neural convolucional (CNN), em um extrator de características e em uma rede classificadora. As principais contribuições dessa segunda parte são: 1) a implemetação de um gerador de dados automático para produzir e rotular frames a partir de videos de jogos; 2) A construção de um de datasets balanceados para o treinamento dos modelos; 3) a implementação de uma MobileNetV2 refinada para tratar especificamente de vídeos de jogos; 4) a implementação de modelos de aprendizagem profunda para a realização de classificação de eventos em cenários MIML.

Palavras-chave: Vídeo de gameplay, eventos de jogo, extração de características, classificação, Super Mario Bros, CNN, RNN, quadros, chunks, MIML. classificação de eventos, multi-rótulo, Deep MIML, classificação multi-rótulo, classificação multi-instância, classificação em vídeo.

A Deep Learning system to perform Multi-Instance Multi-Label event classification in video game footage

Etienne da Silva Julia

Universidade Federal de Uberlândia
Faculdade de Computação
Programa de Pós-Graduação em Ciência da Computação

Uberlândia
2022

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Coordenação do Programa de Pós-Graduação em Ciência da Computação

Av. João Naves de Ávila, nº 2121, Bloco 1A, Sala 243 - Bairro Santa Mônica, Uberlândia-MG, CEP 38400-902

Telefone: (34) 3239-4470 - www.ppgco.facom.ufu.br - cpgfacom@ufu.br

ATA DE DEFESA - PÓS-GRADUAÇÃO

Programa de Pós-Graduação em:	Ciência da Computação				
Defesa de:	Dissertação de Mestrado 13/2022, PPGCO				
Data:	17 de agosto de 2022	Hora de início:	14:00	Hora de encerramento:	15:49
Matrícula do Discente:	12012CCP002				
Nome do Discente:	Etienne da Silva Julia				
Título do Trabalho:	A Deep Learning-based system to perform multi-instance multi-label event classification in video game footage				
Área de concentração:	Ciência da Computação				
Linha de pesquisa:	Ciência de Dados				
Projeto de Pesquisa de vinculação:	-				

Reuniu-se, por videoconferência, a Banca Examinadora, designada pelo Colegiado do Programa de Pós-graduação em Ciência da Computação, assim composta: Professores Doutores: Paulo Henrique Ribeiro Gabriel - FACOM/UFU, Marcelo Costa Oliveira - UFAL e Marcelo Zanchetta do Nascimento - FACOM/UFU, orientador do candidato.

Os examinadores participaram desde as seguintes localidades: Paulo Henrique Gabriel - Uberlândia/MG; Marcelo Costa Oliveira - Maceió/AL e Marcelo Zanchetta do Nascimento - Uberlândia/MG. O discente participou da cidade de Uberlândia/MG.

Iniciando os trabalhos o presidente da mesa, Prof. Dr. Marcelo Zanchetta do Nascimento, apresentou a Comissão Examinadora e o candidato, agradeceu a presença do público, e concedeu ao Discente a palavra para a exposição do seu trabalho. A duração da apresentação do Discente e o tempo de arguição e resposta foram conforme as normas do Programa.

A seguir o senhor presidente concedeu a palavra, pela ordem sucessivamente, aos examinadores, que passaram a arguir o candidato. Ultimada a arguição, que se desenvolveu dentro dos termos regimentais, a Banca, em sessão secreta, atribuiu o resultado final, considerando o candidato:

Aprovado

Esta defesa faz parte dos requisitos necessários à obtenção do título de Mestre.

O competente diploma será expedido após cumprimento dos demais requisitos, conforme as normas do Programa, a legislação pertinente e a regulamentação interna da UFU.

Nada mais havendo a tratar foram encerrados os trabalhos. Foi lavrada a presente ata que após lida e achada conforme foi assinada pela Banca Examinadora.

Referência: Processo n° 23117.059846/2022-68 SEI n° 3844416

Abstract

Video games, in addition to representing an extremely relevant field of entertainment and market, have been widely used as a case study in artificial intelligence for representing a problem with a high degree of complexity. In such studies, the investigation of approaches that endow player agents with the ability to retrieve relevant information from game scenes stands out, since such information can be very useful to improve their learning ability. This work is divided into two parts, the first proposes and analyses new deep learning-based models to identify game events occurring in Super Mario Bros gameplay footage. These models are composed of a feature extractor convolutional neural network (CNN) and a classifier neural network (NN). The extracting CNN aims to produce a feature-based representation for game scenes and submit it to the classifier so that the latter can identify the game event present in each scene. The main contribution of this first part is to demonstrate the greater performance reached by the models that associate chunk representation of the data with the resources of the classifier recurrent neural networks (RNN). The second part of the study presents two deep learning (DL) models designed to deal with multi-instance multi-labels (MIML) event classification in gameplay footage. The architecture of these models is based on a data generator script, a convolutional neural network (CNN) feature extractor, and a deep classifier neural network. The main contributions of this second part are: 1) implementation of an automatic data generator script to produce the frames from the game footage; 2) construction of a frame-based and a chunk-based pre-processed/balanced datasets to train the models; 3) generating a fine-tuned MobileNetV2, from the standard MobileNetV2, specialized in dealing with gameplay footage; 4) implementation of the DL models to perform MIML event classification in gameplay footage.

Keywords: Gameplay footage, game events, feature extraction, classification, Super Mario Bros, CNN, RNN, frames, chunks, MIML, event classification, multi-label, Deep MIML Network, multi-label classification, multi-instance classification, video classifica-

tion.

List of Figures

List of Tables

Acronyms list

AI	*artificial intelligence*
CV	*computer vision*
CNN	*convolutional neural network*
DL	*deep learning*
LSTM	long short term memory
ML	machine learning
NN	Neural Network
OF	optical flow
RNN	*recurrent neural network*
VG	*video games*

Contents

Chapter 1 Introduction

Since their inception, over 40 years ago, video games have been constantly evolving and gaining in popularity. In the last 10 years, they have been one of the fastest growing economic sectors and, according to *Global Data*, the video games industry is expected to reach an overall value of 300 billion dollars by 2025 (Global Data, 2021). One of the main reasons for this growth is the huge amount of people that enjoy playing for entertainment. The last report from *Entertainment Software Association* indicates that over 214 million people, in the United States alone, play at least one hour a week (Entertainment Software Association, 2020).

One of the more common research goals in games is the development of intelligent agents with a high level of play, that are able to defeat even the best human players, like the *AlphaZero* system created to master the games of chess, shogi and go (SILVER et al., 2018). However, video games have also been used as case studies for machine learning (ML) research for a long time, since their high degree of complexity and variability makes them great benchmarks for the state of the art of *artificial intelligence* (AI) algorithms. A lot of games can be used to emulate practical real world situations, but in a controlled, easily scalable and adjustable way (LEE; KIM; SUH, 2017). Games are also great case studies in areas such as education, health and psychology, a fact that has motivated the development of various kinds of research in AI whose objectives are focused on such domains (KOZLOV; JOHANSEN, 2010; BOYLE; CONNOLLY; HAINEY, 2011; Janarthanan, 2012; ANNETTA, 2008).

Despite all of these applications, there is still a great lack of research that focus on the retrieval of relevant information from video game environments (LUO et al., 2018). This presents a problem since the ability of autonomous agents to obtain relevant information about the environment they operate in, and use this information to improve their decision making is one of the main requisites for good performance in any machine learning application.

One of the main reasons for this lack of focus on retrieving information from game scenarios is that, traditionally, video game research assumes access to the *Game Engine*, which is essentially the game's source code. The game engine allows for access to certain system flags, called game logs, which register what the players are doing, as well as what is happening in the environment. However, retrieving this information is not a trivial task, for the following reasons: firstly, game engines are usually inaccessible due to the companies' privacy concerns (LUO; GUZDIAL; RIEDL, 2019); secondly, even considering game platforms that make the game logs available, it is not possible to count on them in situations where information must be retrieved from gameplay footage. This second reason is particularly interesting because, nowadays, with the growth in popularity of video and streaming platforms for video games (VG) like *Youtube Gaming*, *Twitch* and *Facebook Gaming*. In this scenario, thousands of gameplay videos are posted every single day. This corresponds to a giant pool of data that can be explored only through analyzing gameplay footage.

In order to work on information retrieval from games, it is first necessary to understand that VG are essentially interactive videos that run in real-time, where the player has the ability to make decisions that trigger the next game scenario. Considering this, the three main components that describe a game scene are: the *Environment Objects*, the *Game Actions*, and the *Game Events*.

The *Environment Objects* are all the visible items and characters present in a game scene. Identifying these is important because it allows for an understanding of all the different elements with which the player can interact. Considering that object detection in videos and images is one of the main research subjects in CV, there is an abundance of techniques that can be used to address it (MURTHY et al., 2020), (Zhao et al., 2019).

The *Game Actions* consists of all the different actions that are available for the player to take in a certain game scene. These actions are the tools that a player has to interact with the environment.

Although the definition of events in AI is very broad, as shown in (Ravanbakhsh et al., 2017), (ATEFEH; KHREICH, 2015) and (YU et al., 2020), in the context of this thesis, the *Game Events* represent the effects that *Game Actions* have on the *Environment Objects*. In a broader sense, when considering dynamic environments, such as video games, the events represent the dynamic changes and interactions that happen in the environment.

So considering the fact that: 1) there are a lot of existing techniques that can be used to detect *Environment Objects* and; 2) since *Game Actions* trigger the *Game Events*, the events that occur can be used to infer the actions; this project proposes different *deep learning* (DL) techniques to automatically retrieve *Game Events* from gameplay footage. As a case study, the game *Super Mario Bros* was used, since it is a classic and very famous game that has been widely used in scientific research.

1.1 Goals and Challenges

The goal of this work is to compare DL and *computer vision* (CV) techniques, more specifically *recurrent neural network* (RNN),*convolutional neural network* (CNN), and Deep Multi-Instance Multi-Label Networks, to create an effective framework for extracting events from gameplay videos of *Super Mario Bros.* Considering this, the main challenges investigated in this work were:

a) Generating reliable datasets that could be used to train the DL models;

b) Pre-processing the data in a way that would maximize the effectiveness of the models;

c) Working in a scenario where multiple events could happen simultaneously (Multi-Label);

d) Having to adapt ML techniques that were designed for real-world images and videos to the graphical representation of *Super Mario Bros*, due to the lack of works designed to classify events in video games;

e) Having to work in an online context, where the frames have to be processed in real-time.

1.2 Hypothesis

The research question this work seeks to answer is: is it possible, through DL and CV techniques, to generate a viable framework for extracting relevant information from video games that can replace the need to access the *Game Engine*? The secondary questions that are also tackled are: i) can methods designed for real-world scenarios be adapted to work in-game scenes? ii) if so, among a large number of different methods in the literature, which ones are more suited for identifying events in-game scenes? iii) what is the best strategy to deal with multi-labeled frames? Considering these questions, this work was developed assuming that it is possible to generate an effective event classification system by adapting already existing, state of the art, DL and CV techniques to work on video games.

1.3 Contributions

The contributions made in this thesis are:

a) A new approach to generate labeled data from gameplay automatically based on the *Mario AI Framework* (KARAKOVSKIY; TOGELIUS, 2012);

b) A balanced Super Mario Bros dataset specifically designed to contain multi-label and single-label examples;

c) A proposed representation for clustering frames that boosts classification performance in gameplay footage;

d) An evaluation among four state-of-the-art convolutional neural networks (CNN) (ResNet50V2, MobileNetV2, VGG16 and AlexNet) for feature-extraction in *Super Mario* video frames;

e) An investigation among three state-of-the-art neural networks (Long Short Term Memory Network, Gated Recurrent Unit and Deep Multi-Instance Multi-Label Network) for classifying events from *Super Mario Frames*;

f) A fine-tuning strategy to significantly boost the performance of CNNs pre-trained on real-world data when applied to video games;

g) A strategy to fit games video footage into the Multi-Instance Multi-Label framework (ZHOU et al., 2012), in order to deal with the multi-label problem.

All the code developed for the experiments conducted in this thesis are posted in the following repositories:

- ❏ https://github.com/Intn21/DL-Models-for-Performing-Multi-Instance-Multi-Label-Event-Classification-in-Gameplay-footage
- ❏ https://github.com/matheusprandini/Mario-AI-Framework-Generate-Dataset
- ❏ https://github.com/matheusprandini/Data-Wrangling-Mario-Dataset
- ❏ https://github.com/matheusprandini/dnns-game-events

1.4 Contextualization of the proposal

Finally, this work was developed within the scope of a broader project whose main objective is to produce an automatic player agent endowed with the ability to map cognitive and/or motor weaknesses of patients with some types of syndrome (such as Down syndrome) and, in possession of this information, adapt its decision-making engine in order to lead the game to situations that stimulate the cognitive development of its users. In this way, the objectives achieved through the models developed here meet part of the requirements required to execute the first part of this project, that is, to use the events identified in game footage as an instrument to perceive the actions performed by the game users and map their possible cognitive weaknesses.

1.5 Thesis Organization

This chapter presented the goals, challenges, contributions, proposed methodology and motivation for this work. The next chapters will be organized in the following manner:

- ❏ **Chapter 2 - Theoretical Foundation:** Presents the fundamental theoretical concepts that relate to this work, more specifically: Neural Networks, CNN, recurrent neural network (RNN), Deep MIML Network, MIML, CV, Video Games, Video Games as Learning Platforms and Evaluation Metrics for Multi-Label Classification;
- ❏ **Chapter 3 - Related Works:** Presents an overview of the literature related to this work, especially in the context of: machine learning in video games and event and action classification in videos;
- ❏ **Chapter 4 - Investigating the Performance of Various Deep Neural Networks-based Approaches Designed to Identify Game Events in Gameplay Footage:** This chapter presents both the methodology and the experiments regarding the first part of this work;
- ❏ **Chapter 5 - Investigating the Performance of Deep Neural Network-based Approaches Designed to Identify Game Events in Gameplay Footage:** This chapter presents both the methodology and the experiments regarding the second part of this work. This is a followup to the work presented in Chapter 4.
- ❏ **Chapter 6 - Conclusion:** Presents the conclusions that were reached through this work, its contributions, limitations and future works.

Chapter **2**

Theoretical Foundation

This section will introduce the main theoretical concepts that were necessary during the development of this work.

2.1 Deep Learning

Deep learning (DL) is a field of machine learning that consists on the usage of computational models with multiple internal processing layers to detect patterns and complex relations within data. DL algorithms find these patterns in large datasets by gradually adjusting their internal parameters. These parameters determine how each internal layer interprets the received data (LECUN BENGIO, 2015).

The most common learning framework for DL is supervised learning (LECUN BENGIO, 2015). Supervised learning is characterized by the usage of pre-labeled data during the training process. This implies that, in a supervised framework, the data has to contain two components: the *Features*, which correspond to the descriptive parameters of the data, and the labels, that indicate the correct class of the data. In this way, the general training dynamic in a DL process is: i) the model has all it's internal parameters set randomly; ii) the training data is delivered to the model; iii) the model processes the data according to it's own internal parameters and returns an array where each position corresponds to the probability of that instance of data belonging to one of the possible classes; iv) the model compares the position with the highest probability with the correct label; v) in case of a correct prediction, the model maintains the values of it's internal parameters, if not, the model adjusts these parameters based a learning function (LF).

Learning functions measure the distance between a prediction and the correct label according to an error function. From that, the LF indicates how the internal parameters have to be readjusted in order to minimize the error. The internal parameters of networks are numerical values called *weights*, which determine the output of the network for each data input. A traditional DL model can contain up to hundreds of millions of parameters and, because of that, the learning process through weight readjustment corresponds to the highest computational cost in all the applications of DL algorithms. The way these learning rules and weight readjustments work on different DL models is presented in sections 2.1.1, 2.1.2 and 2.1.3.

DL models first appeared as an alternative for linear classifiers because of the ladder's inability to classify data that wasn't linearly separable (LECUN BENGIO, 2015) (see figure 1). Because of this limitation, linear classifiers are dependent on a feature extractor to make sure the data is structured in a linear way. This problem does not affect DL models, because they are more effective in dealing with complex multi-dimensional data (see Figure 2).

Nowadays, DL has considerably improved the performance of the state of the art models in the fields of: voice recognition, computer vision and genomics (LECUN BENGIO, 2015).

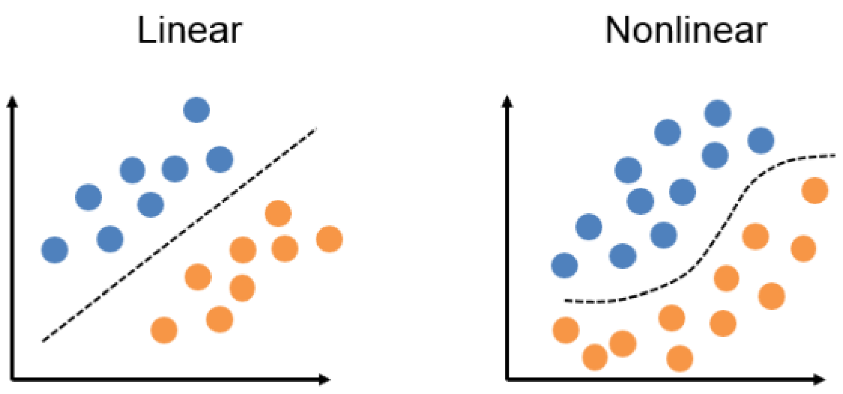

Figure 1 – Representation of a linear problem versus a non-linear problem. From <https://github.com/jtsulliv/ML-from-scratch/blob/master/Neural-Networks/perceptron.ipynb>

Some of the main DL models in the literature are: *neural networks* (NN), more specifically, multilayer perceptrons, **convolutional neural networks** (CNN) and *recurrent neural networks* (RNN). Even though neural networks are some of the most studied architectures in the field of AI, they still present certain limitations, as presented in (MARCUS, 2018). Some of these limitations are:

- The high amount of data required for an adequate training process. Since the weight readjustment in DL is done by progressively observing new data, the higher the number of internal layers in a model, the higher the demand for large quantities of data in order for the network to converge to a good solution.

- The difficulty in identifying errors in the models, especially in complex systems. Due to the huge amount of parameters in DL models, often reaching millions, it is almost impossible to have a solid idea of what each layer of the network is doing to solve the problem. For this reason, NNs are often used as a sort of "black box", where there is a lack of understanding of the process that is being used to solve a problem (SAMEK; WIEGAND; MüLLER, 2017).

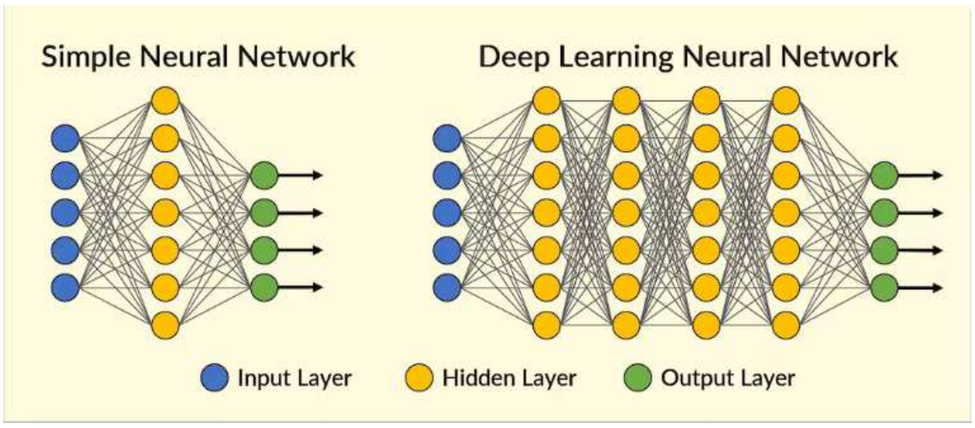

Figure 2 – Representation of a linear classifier (left) versus deep classifier (right). From <https://www.securityinfowatch.com/video-surveillance/video-analytics/article/21069937/deep-learning-to-the-rescue>

2.1.1 Neural Networks

Neural Network (NN) is an artificial intelligence model inspired by the biological structure of the human brain. They consist of a set of elements, called neurons, that are connected to one another, storing and transmitting the information. The two essential components of a NN are neurons and connectors. Figure 3 shows the structure of a NN, with an input layer, hidden layers (internal layers), and output layers. The neuron is the fundamental processing structure in a NN, they have the function of processing and sending a signal to other neurons through the connections. The connectors connect two different neurons and have a weight value attributed to them. This value multiplies the exit signal from a neuron before it reaches the next neuron. Next, the layout of a NN is explained:

- *Input layer*: As the first neuron layer, the input layer is responsible for receiving the input data. This layer will have a number of neurons corresponding to the size of the data representation that is being used. For example, considering the challenge of recognizing handwritten digits presented in the MNIST dataset (Lecun et al., 1998), where each image is represented by 784X784 pixels in *greyscale*, a NN will need to have 614,656 total neurons in order to receive each input (one neuron per pixel);
- *Hidden layer*: Neuron layers that are in between the input, and output layers. These are responsible for processing information and for the general learning process of the NNs;

❏ *Output layer*: In addition to being the last neuron layer, it is also responsible for outputting prediction values. In a classification problem, for example, the output layer returns, for each input, percentage values for each possible label.

The learning process in NNs consists of adjusting the weight values in the connectors to gradually improve performance. This process is called backpropagation and can be done with multiple learning rules. Some of the main ones are: the Delta Rule, the Generalized Delta Rule, and the Hebb Rule. These rules are iterative learning processes based on mathematical functions that determine how much each weight readjustment will impact the final result of the network. Due to the large number of parameters traditionally associated with NNs, and the fact that NNs take multiple iterations to converge, backpropagation has a very high computational cost.

Some of the applications for NNs are: function estimation, process management, pattern classification, data clustering, system change predictions, function optimization and object detection in images and videos (WIDROW; RUMELHART; LEHR, 1994), (DILLON; NIEBUR, 1996), (UDO, 1992). This work will focus primarily on DL models specialized in CV, which are presented in section 2.1.2.

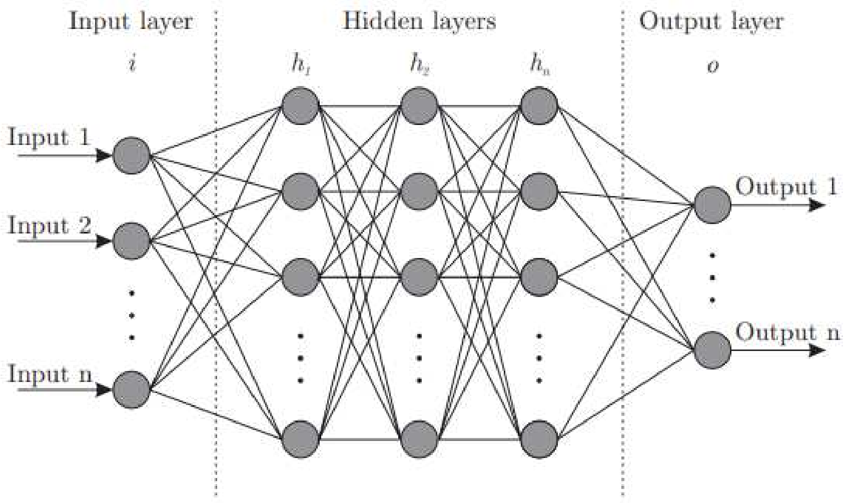

Figure 3 – Representation of a NN, from (BRE; GIMENEZ; FACHINOTTI, 2017)

2.1.2 Convolutional Neural Networks

Like NNs, CNNs also attempt to imitate the learning process of the human brain (RAWAT; WANG, 2017). However, they are specialized in CV problems, like image classification and object trajectory analysis in videos. A CNN is composed of a series of image kernels (or filters) designed to perform an automatic selection of features that allow for a concise and clear representation of images presented at the input of the network (Aloysius; Geetha, 2017). This model is made of sequences of pairs composed of pooling and convolution layers (figure 5).

The convolution layers apply multiple kernels to an image, where each kernel aims to retrieve a set of relevant features to represent it. On the other hand, the pooling layers consist of a function that shrinks the dimensions of the image. This pooling process lowers the total number of parameters in the network, consequently decreasing the computational complexity of the network and controlling overfitting (Aloysius; Geetha, 2017). Figure 4 shows an example of a pooling operation called *max pooling* with stride 2. In this case, the input is divided into 2x2 quadrants, where the distance from the first pixel of a quadrant, position $[1, 1]$, to the next quadrant will be equal to the stride. From this, each region is converted to a single position, whose value corresponds to the higher value pixel of the quadrant. The combination of multiple of these layers generates a smaller representation of the input that contains only the features that better represent it.

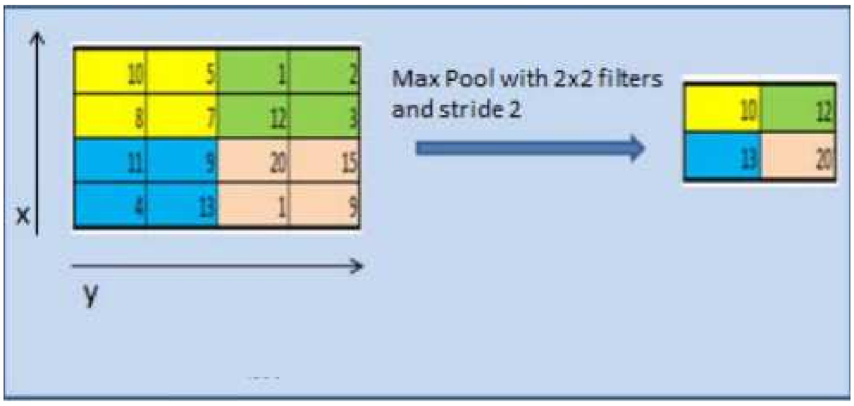

Figure 4 – Example of a max pooling operation in a 4x4 input.

After going through multiple convolutions and poolings, the input image becomes a more abstract representation of itself. This representation is then passed on to the final part of the network, the *fully connected layer*, which basically has the structure of a traditional NN. Considering a classification problem, this layer will then output the class probability for each label. Some examples of recent CNN models are: *LeNet* (EL-SAWY; EL-BAKRY; LOEY, 2017), *AlexNet* (KRIZHEVSKY; SUTSKEVER; HINTON, 2012) and *ResNet* (HE et al., 2016).

Despite the fact that CNNs suffer from a lot of the same problems as traditional NNs (section 2.1.1), especially regarding the high volume of data necessary for training, they have had a lot of success in solving CV problems, as stated in (Aloysius; Geetha, 2017).

2.1.3 Recurrent Neural Networks

Even though the traditional NN structure is an extremely important model with applications across various fields, it has difficulty classifying instances that are related to previous events. For example, in an object trajectory identification problem, traditional NNs will receive various images of an object in different positions and try to infer it's the

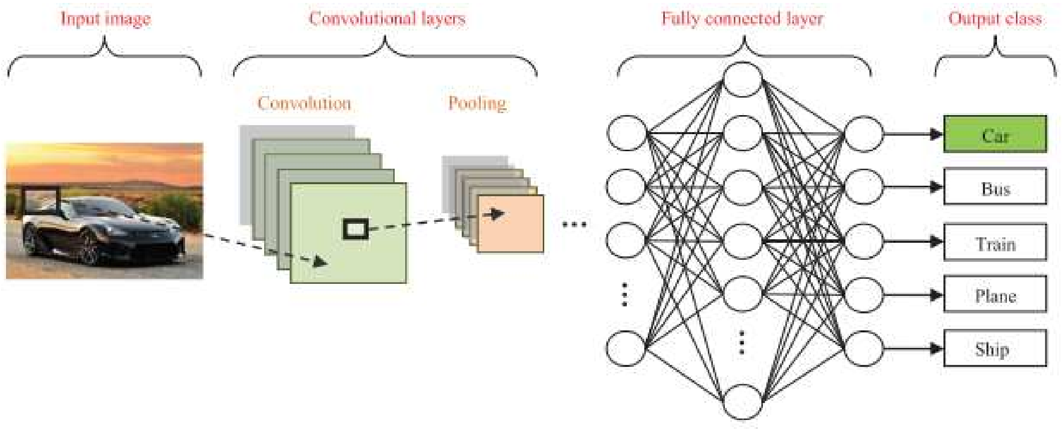

Figure 5 – Representation of a CNN, from (Aloysius; Geetha, 2017)

corresponding trajectory from each of these images. However, this becomes a very difficult task when considering only one position of the object at a time, since different trajectories may overlap at certain points. In this example, being able to take into consideration positions that were previously processed by the network is a very helpful feature.

Both previously presented NN models have a *feedfoward* flow of information, which implies that the network connections all follow the same direction, flowing sequentially from the input layer, towards the output layer (figure 6). RNNs are models that embody temporal, and sequentially notions in the learning process. They have specific connections between neurons, called *feedback* connections, that allow for the persistence of important information in the network thought the processing of multiple instances, by feeding back certain outputs as new inputs (see figure 7). This makes RNNs extremely effective in scenarios where temporal dependencies between the data are relevant.

The first proposed RNN models, (JORDAN, 1986) and (HOPFIELD, 1982), had good initial results, but presented a problem called gradient loss in situations where the temporal dependencies were stretched among too many instances (HOCHREITER, 1998). This essentially meant that towards the later layers in *backpropagation*, the gradient of the loss function would approach zero, and the network would only be trained effectively up to a certain layer. As a solution for this problem, *Hochreiter* and *Schmidhuber* proposed, in 1997, the long short term memory (LSTM) (HOCHREITER; SCHMIDHUBER, 1997) model, where each neuron in a hidden layer was replaced by a memory cell. Each one of these cells had a connection that allowed for a direct transfer of the local gradient to future processes, contrary to the previous RNNs that would lose this information. This solution allowed for relevant information to persist in the network for more iterations. Even though the LSTM is a model designed over 20 years ago, it is still, to this day, one of the most used RNNs in the literature (LIPTON, 2015).

Another, more recent, proposal that also handles the gradient loss problem is the gated recurrent unit (GRU) model, which was developed by *Kyunghyun Cho* in 2014 (CHO et al., 2014). The GRU works in a very similar way to the LSTM, but with an optimized architecture that allows for lower memory consumption and faster training.

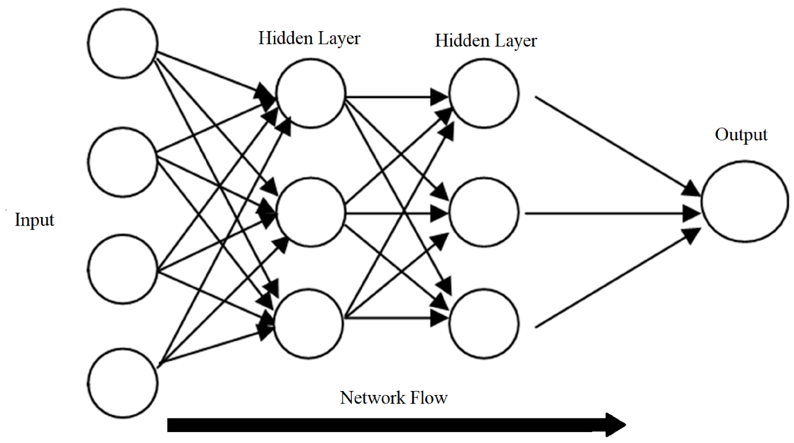

Figure 6 – *Feedfoward* flow

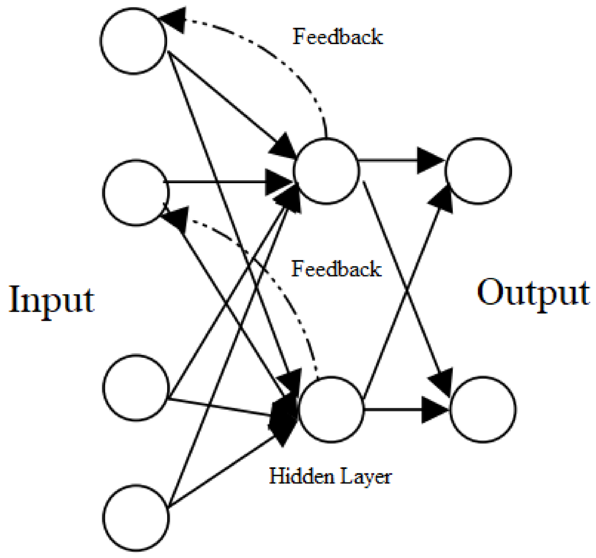

Figure 7 – *Feedback* flow

2.2 Multi-Instance Multi-Label Framework

The multi-instance multi-label (MIML) framework, proposed in (ZHOU et al., 2012), is defined by each data example being comprised of multiple instances, associated with multiple labels. Traditionally in ML, each instance of data is associated with a single label, or even, each instance with multiple labels (all these frameworks are shown in figure 8).

However, for problems involving complicated examples with multiple semantic meanings, using more than one instance to represent them can allow for additional inherent patterns in the data to become more clear. Then, in these situations, the MIML framework can be a more natural and appropriate way to represent the data.

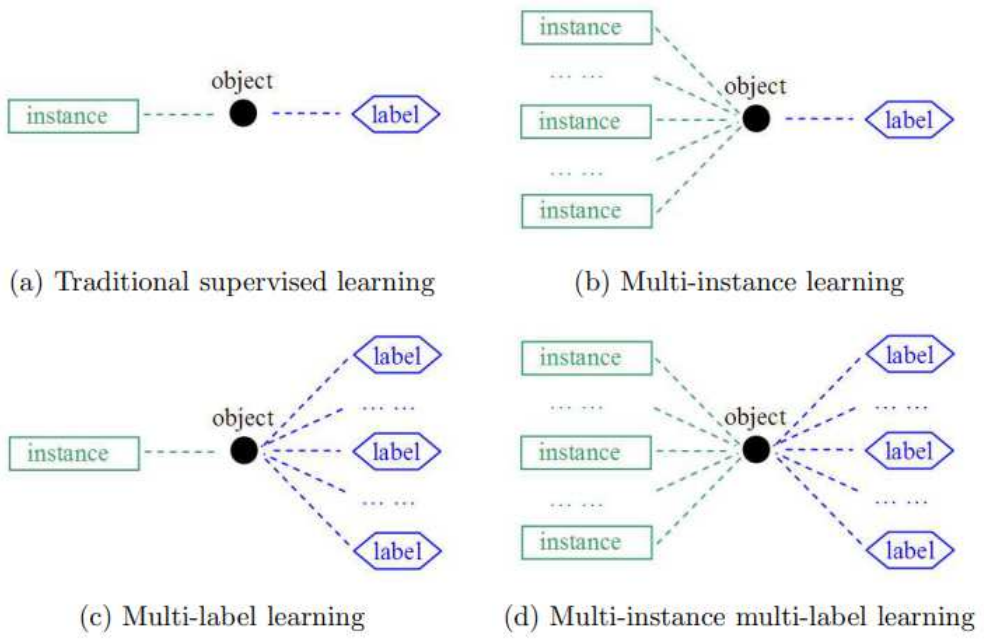
instance
object
label
(a) Traditional supervised learning
(b) Multi-instance learning
(c) Multi-label learning
(d) Multi-instance multi-label learning

Figure 8 – Four different learning frameworks, image from (ZHOU et al., 2012)

An important idea in multi-instance learning is the sub-concept. When considering broad concepts like "*Africa*", it can hardly be described by only one aspect. Usually, a group of cultural and environmental identifiers is required. So the concept "*Africa*" could be identified from, for example, an image of a grassland environment, coupled with lions and trees. These low-level concepts that aren't necessarily enough on their own, but when combined are capable of describing complex and broad concepts, are the sub-concepts.

2.2.1 Deep MIML Network

As previously mentioned in section 2.2, a lot of real-world applications greatly benefit from being represented in a multi-instance multi-label context. The Deep MIML Network (FENG; ZHOU, 2017) is a model that adds an automatic instance generator and a sub-concept learning structure to the traditional neural network formation. The structure of the Deep MIML Network (Figure 9) is an instance generator module (usually a CNN network), followed by a *sub-concept layer*, which is essentially a classifier that matches the scores between instances and sub-concepts for every label. This approach allows for an instance-label relation discovery that works very well in a MIML framework.

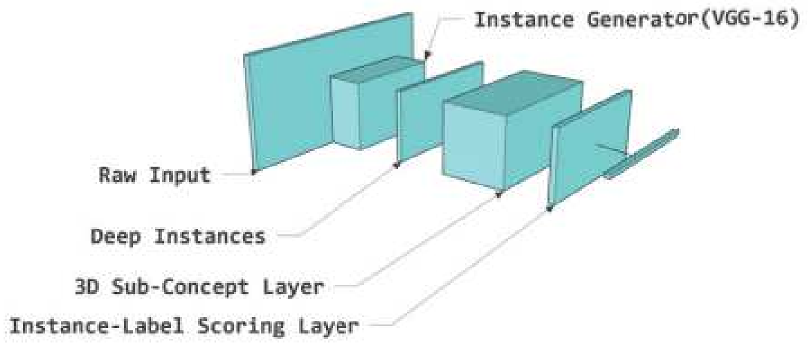

Figure 9 – Representation of a Deep MIML Network, image from (FENG; ZHOU, 2017)

2.2.2 Multi-Label Evaluation Metrics

In traditional supervised learning, as mentioned in the previous subsection, a single instance is associated with a single label. This allows for an *accuracy* metric, the percentage of examples correctly classified, to often be a good enough indicator of performance (ZHOU et al., 2012). However, in multi-label situations, the goal is not to identify a single label, but rather to correctly classify the highest amount possible from a group of labels. For example, it is better for the model to guess 4 out of 5 labels correctly than to get 2 out of the same 5 labels. This has to be taken into account by a multi-label evaluation metric for it to be an effective performance indicator.

The evaluation metrics relevant to this work are explained below:

- Hamming loss: a loss metric that represents the fraction of labels that are incorrectly predicted, be it a correct label that is missed, or a wrong label that is predicted. Considering the function $hamming_loss(h) = 0$, the lower the value of $hamming_loss(h)$ the better the performance of h.

- Mean average precision: the average precision (AP) is a relation between the class precision and class recall. Precision (P) indicates how many predictions were correct and recall (R) indicates how many of all instances of the class were predicted by a model. The average precision corresponds to the area under the graph $precisionXrecall$. Since the average precision corresponds to a single class, the mean average precision (mAP) corresponds to the average APs over all classes.

- F1 score: the F1 score, also is a relation between class precision and class recall. However, it corresponds to the harmonic mean between the two, so: $F1 = 2 * \frac{P*R}{P+R}$. Like mAP, the F1 score also references a single class, so in a multi-label context, this metric corresponds to the average F1 score of all classes.

2.3 Computer Vision

Computer vision (CV), is an interdisciplinary field whose primary objective is to replicate the abilities of human vision to interpret and recognize images and videos. More specifically, CV essentially works with image analysis (SZELISKI, 2010b). The field first appeared in the 70's, and it's proposal, compared to the already existing field of digital image processing, was to interpret tri-dimensional structures from real-world images, in such a way as to allow for a better understanding of full scenarios. In the last 10 years, advances in deep learning (section 2.1) have contributed to CV, in image recognition, as well as filtering, noise removal, etc (SZELISKI, 2010b). The main steps of image analysis in CV are: image formation, object segmentation, feature extraction, and lastly, image recognition.

Image Synthesis is the process of producing an image from geometrical shapes, light, and textures. These shapes consist of dots, lines, and planes, which are the basic components in the description of tri-dimensional spaces. The light includes all the lighting aspects in a scenario, the reflections, and shadows. Finally, the textures correspond to spatial variations in pixel intensity, that determine the different regions of an image. In another stage, object segmentation aims to detect the pixel groupings that correspond to each of the objects in an image. Considering, for example, an image of a busy street, a segmentation process can be used to identify the different vehicles, pedestrians, obstacles, etc. Feature extraction is the process of highlighting the relevant features of an image. These may correspond to specific geometrical structures, movement-related aspects, and objects. Finally, image recognition aims to classify objects, people, animals, and pretty much any component that can be found in an image. This is the hardest among all aforementioned visual tasks, as stated in (SZELISKI, 2010a).

Despite many advances in CV and digital image processing, the accuracy of state-of-the-art techniques in object classification is still far inferior to the average human. The main difficulty in this task is the fact that the same objects can present themselves in many different ways in a scenario. A person can make different poses, use different clothes and change their facial expression. For a human, identifying all these differences is a completely effortless process, but for a machine, figuring out which one of these variations is relevant, or not, during a classification process is an extremely difficult task.

In addition to digital images, another kind of input that is studied in CV is digital videos. Even though they can be interpreted as sequences of images, they can represent a much more detailed description of actions and human movement (AGGARWAL; CAI, 1999). Because of this, videos are extremely valuable for research that involves the detection of actions and trajectories. In the context of DL, a lot of techniques were adapted to work better in this context. One of them is the usage of the optical flow (OF) as an additional dimension to the data (BEAUCHEMIN; BARRON, 1995).

The OF is a representation of object movement across multiple frames from a video that allows for a combined spatial-temporal analysis of videos. This can improve object tracking in video models for action detection (see Figure 10).

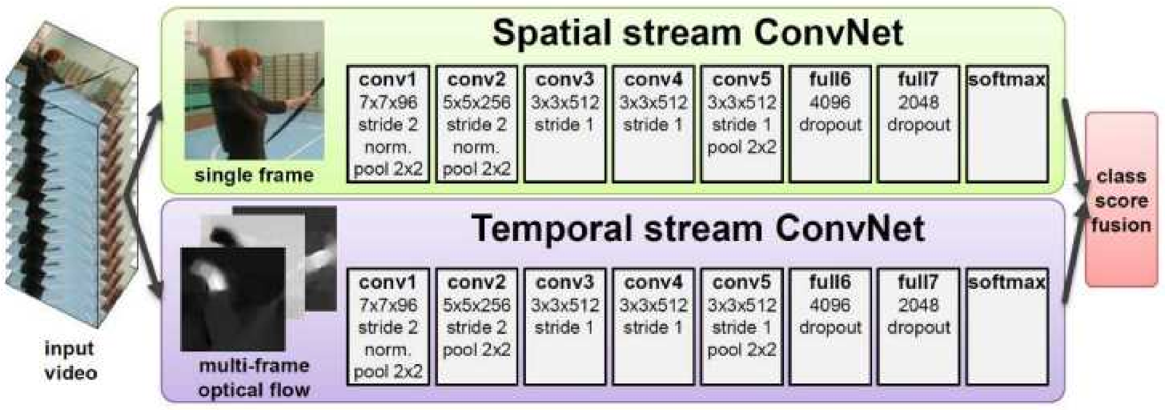

Figure 10 – Representation of a DL model processing the *optical flow*, from (SZELISKI, 2010a)

Among the many current applications of CV, some of the most relevant are:

- Action and activity detection; is the process of automatically detecting human actions, usually in videos. It is a very relevant problem since it allows for a better understanding of the physical aspect of how humans traverse different situations (XU et al., 2018);
- Interpretation and highlighting of medical images; since many medical procedures, like ultrasound and magnetic resonance imaging, return their results in images, the application of CV methods allow for improved image quality, which positively helps the specialist analysis (BISWAS et al., 2019);
- Autonomous vehicle guidance; autonomous driving in ground vehicles is a feature that has become progressively more popular with advances in camera, sensor, and processor technologies (JANAI et al., 2020). Even though most recent applications don't rely on DL, advances in the field have shown the potential for certain applications (LEE; KIM; SUH, 2017).

2.4 Video Games

Since their appearance over 40 years ago, *video games* (VG) have been constantly evolving and becoming more popular. In the context of academic research, in addition to being rich and diverse simulation scenarios (TOGELIUS, 2017), they allow for use of the own player as a study object. VGs are defined as an entertainment medium that promotes constant interaction with the user, which allows for a different gameplay experience for each player, both in a mechanical aspect, as well as in an emotional one.

As a research subject, games can be divided into recreational and serious games. The main aspects of recreational ones are: the story, the art, and the software (ZYDA, 2005). The story of a game consists of its different plots and NPCs (*Non-Playable Characters*. The art involves all the visual and sound expressions in a game, the character models, the scenarios, the color pallet, the sound effects, the soundtrack, etc. The software refers to the game engine, which is essentially the source code used in the development of the game. In general terms, the game engine is responsible for functionalities related to graphics rendering, physics simulation, sound triggers, animations, memory management and general processing. Besides the previously presented characteristics, the main difference between VG and other entertainment mediums, like television or movies, is the game mechanics. These are defined by all the possible interactions the player can have with the game. In the game Super Mario bros, for example, the game mechanics are: jumping, swimming, running. collecting items, etc.

In the case of serious games, need to have all of the previously mentioned characteristics, but also need to have a pedagogical agenda included, in order to allow for the completion of learning goals inside the game (ZYDA, 2005). The differences between entertainment and serious games are shown in Figure 11.

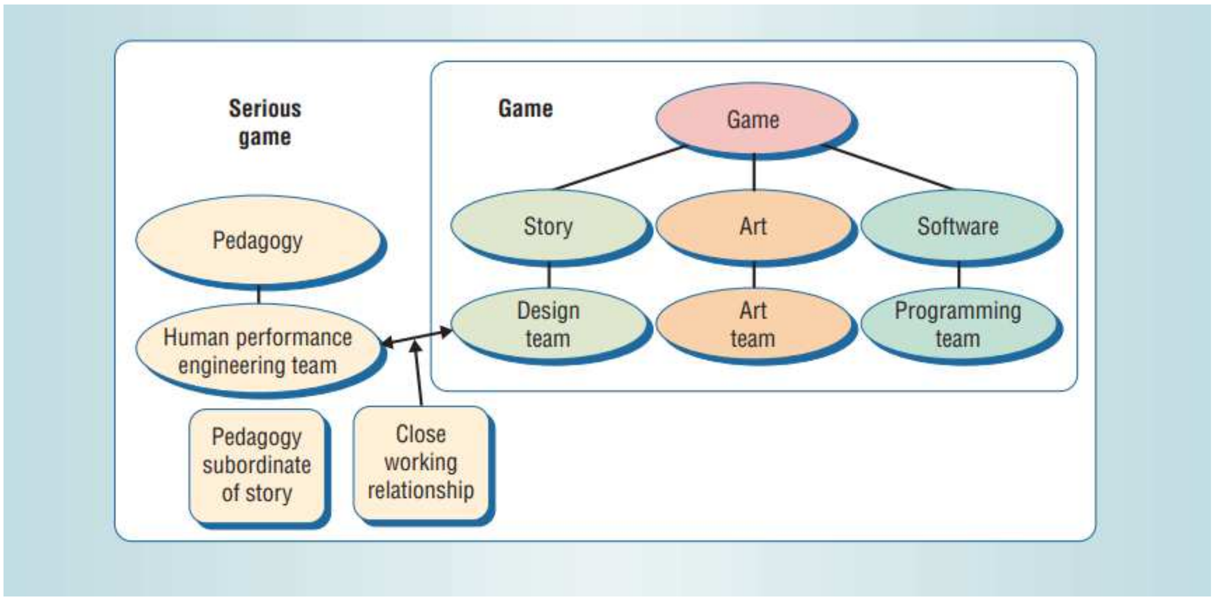
Serious game
Game
Game
Pedagogy
Story
Art
Software
Human performance engineering team
Design team
Art team
Programming team
Pedagogy subordinate of story
Close working relationship

Figure 11 – Serious Game versus Entertainment Game, from (ZYDA, 2005)

Another fundamental component of VGs is the game genre. It defines the kind of experience each game is expected to provide, and each genre stimulates the players differently. A general division of game genres, proposed in (APPERLEY, 2006), can be seen below:

- ❑ Simulation: consists of all games that represent activities such as sports, driving, flying, etc. The proposal of these games is to try to simulate certain real-world experiences inside of a virtual environment. An example of this genre is the franchise *Truck Simulator*, which simulates the player driving a truck and making deliveries in different parts of the world.

❑ Strategy: these are games that prioritize strategies and game tactics. Some of the different sub-genre of games that fit in this category are: *real-time strategy* (RTS) and *multiplayer online battle arena* (MOBA), both types of games involve the control of multiple units and structures in order to complete various objectives. In the research literature, a game that is widely studied is *Starcraft 2* (VINYALS et al., 2017a), a multi-player RTS game where the player controls different alien civilizations in order to manage their units and resources to destroy the enemy civilizations as fast as possible.

❑ Action: this genre can be separated into two main sub-genres: *first person shooters* (FPS), and *third person games* (TPG). The difference between the two is the perspective provided for the player, in FPS the camera simulates the vision of an in-game character, while in TPG, the player observes the in-game character from a top view. Action games are usually fast paced and present complex actions that can be executed. The franchise *Call of Duty*, is a good example of the action genre, it puts the player in a simulated warfare situation, where he needs to move quickly in order to kill enemies and fulfill game objectives.

❑ Role-Playing Games (RPGs): RPGs are adventure games traditionally associated with the fantasy genre. Their goal is to provide the player with a role-playing experience as if he was an in-game character. These games usually involve decision-making, a lot of NPC dialog and a great focus on story-telling.

❑ Platform Games: This genre is extremely well known, and represents some of the first digital games ever created. These games usually have a 2D (two-dimensional) perspective and are separated into multiple levels. In order to beat each level, the player has to control a character to cross multiple obstacles through a sequence of precisely timed actions. One classic example of the genre, which is also widely researched in the literature, is *Super Mario Bros*.

VGs have a great stylistic variety, which is why they appeal to many different audiences. Because of this, they can be rich and customizable testing scenarios for many research fields, such as AI and medicine.

2.4.1 Video Games as Learning Platforms

AI investigates complex interactions between agents and environments in various different contexts. VGs can be seen as ideal platforms for studies in this field since they are a controlled, safe, fast, and easily adjustable environment for a lot of different types of experiments. In this context, as presented in Figure 12, game scenarios work as environments that agents navigate through with different game actions and, based on the success or failure of these actions, the agent is either rewarded or punished.

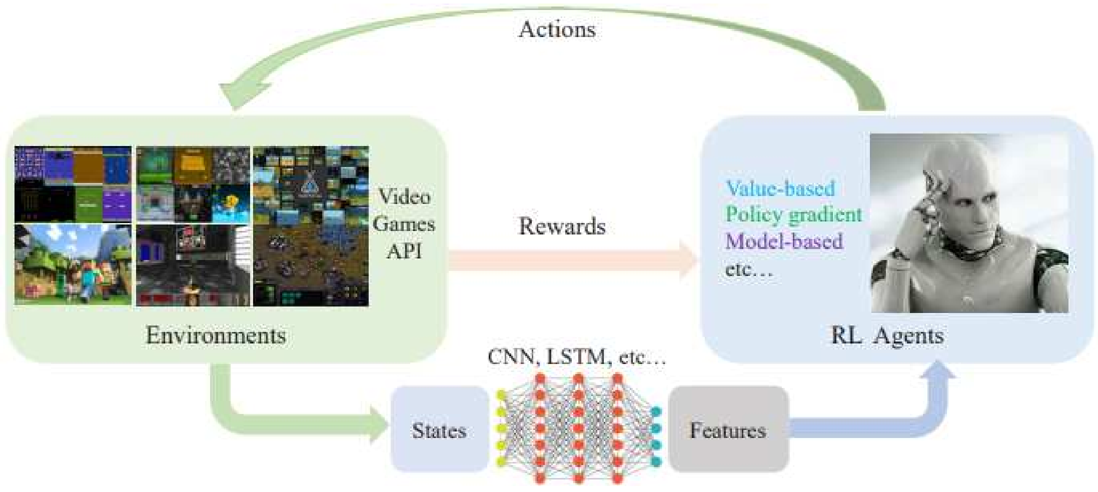

Figure 12 – Representation of the decision-making process of an agent inside a video game, from (SHAO et al., 2019)

These game scenarios, according to (SHAO et al., 2019), can be divided into two types of platforms: *general*, or *specific*. General platforms consist of groups of tasks and challenges from different games, specifically designed for scientific research. The *Arcade Learning Environment* (ALE) (BELLEMARE et al., 2012), and *Open AI Gym* (BROCKMAN et al., 2016), are both examples of general platforms. On the other hand, specific platforms are based on a single game or game adaptation. Some examples present in the literature are: *TorchCraft* (SYNNAEVE et al., 2016), based on the game *Starcraft*, and the *Starcraft II learning environment* (VINYALS et al., 2017b), which is based on the game *Starcraft II*.

Even though video games still have a lot of unexplored possible applications within AI, a lot of interesting work already exists in the literature, like the creation of automatic player agents and the generation of testing scenarios for experiments. The creation of automatic agents involves the use of AI and DL techniques in order to train these agents to be able to overcome game challenges (see Figure 12), compete against other player agents, or even against humans. A current example of this type of work is the *AlphaZero* system (SILVER et al., 2018), an automatic player developed by the company *DeepMind* to master the games of chess, shogi and go. The generation of testing scenarios is a field that focuses on the use of video games to simulate real-world scenarios that are hard to replicate in real life (LEE; KIM; SUH, 2017). This is particularly valuable for DL since the algorithms in this field require a large amount of training data, and the collection of this data can be a very challenging task. So being able to rapidly simulate various different scenarios in a game can allow for an effective way to generate data.

The application of DL in video games is a recent research field that has been showing great potential to improve AI as a whole. Even though a lot of interesting work has already been conducted in the field, there is still a lack of new *frameworks* that help with the process of working with complex scenarios inside of video games, in a way to facilitate and incentive new research to be developed in the field (SHAO et al., 2019).

2.4.2 Super Mario Bros

Super Mario Bros is a platform game (first released in 1985) in which the player has a side view of the character he is controlling (Mario) and his goal is to traverse a series of game levels, each with a different set of obstacles, be it enemies, projectiles, or even difficult terrain that requires a set of precise actions to be traversed. In order to overcome these obstacles, the player relies on a fixed set of actions that Mario can execute, which are: walking, running, jumping, and throwing fire. All these actions trigger specific *Game Events*, which can be interpreted as the impact that Mario's actions have on the environment. A few examples of these events include killing an enemy with fire, breaking a block by jumping underneath it, and getting damaged by an enemy. A more comprehensive view of these game events will be presented in section 4.1.3. From a programming standpoint, the game is executed from what is called a *Game Engine*. Every action and event that occur is stored as a *Game Log* in a table of the game engine.

In Super Mario Bros, the graphics are very simple, being composed of a few pixelated sprites that represent different character models, objects, or background scenery. In addition to that, the game has a 2D (two-dimensional) perspective, which, contrary to a 3D one, does not give depth to the objects (ROETTL; TERLUTTER, 2018), as shown in Figure 13. Visually, the game consists of sequences of frames refreshing very fast on a screen (usually 30 to 60 frames per second (FPS)), which can generate a very large amount of data in a short span of time. In this way, lowering the frame rate at which the frames are being captured, or clustering sequential frames into groups called *chunks*, both allow for a more condensed representation of the game scenarios.

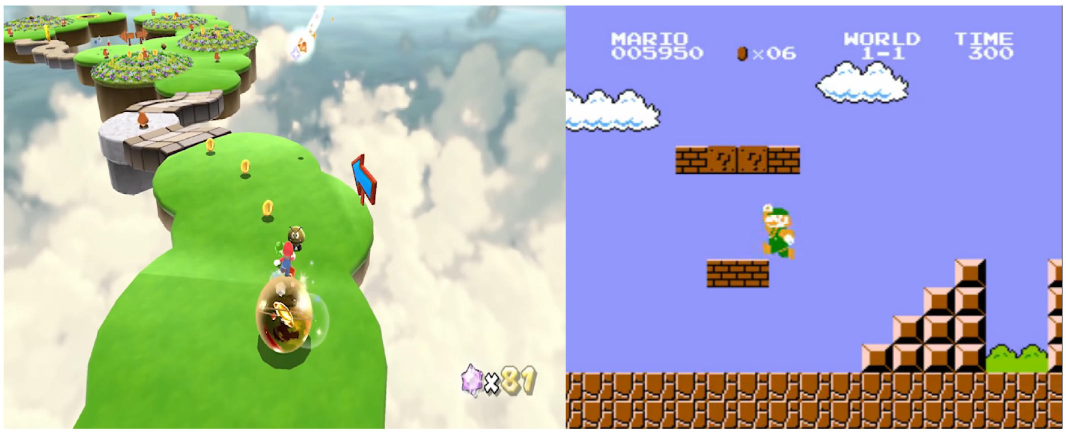

Figure 13 – In-game footage of the games Super Mario Galaxy on the left (3D) and Super Mario Bros on the right (2D).

Chapter 3

Related Works

This chapter will present the main works in the literature that relate to scope of this thesis.

3.1 Action Detection in Videos

A lot of works have been done around the task of action detection, especially since it's one of the main research topics of CV (Section 2.3). Most of these works can be separated into two approaches: offline action detection and online action detection. The main difference between both is that the offline methods observe the entire video before returning predictions, while the online ones have to make their predictions in real-time, as soon as the action happens. Since this work involves online detection, this section will present the relevant works around this topic in the current literature.

In (GEEST et al., 2016), Roeland de Geest and Tinne Tuytelaars tackled the problem of online action detection in videos. Their goal was to find a way to detect the action as fast as possible, even before it finished happening. This was very challenging because a lot of predictions had to be done after observing a very small fraction of an action. In order to solve this, the authors proposed three different approaches: i) one based on the combination of *improved trajectories* (WANG; SCHMID, 2013) with *Support Vector Machines* that had inputs obtained through *Fisher Vectors*; ii) one based on the VGG-16 (SIMONYAN; ZISSERMAN, 2014) convolutional network; and iii) one based on the LSTM (HOCHREITER; SCHMIDHUBER, 1997), which is the most popular RNN in the literature. In order to conduct the experiments, the authors generated a dataset from 16 hours of footage from famous television shows, containing a total of 30 class labels. It is important to point out that, since the authors wanted to detect the actions as fast as possible, each model received only a fraction of each action during the tests. In their experiments, they observed that among the proposed approaches, none had obtained a satisfactory result, and concluded that early action detection in an online setting was still an open problem to be addressed.

In a continuation of this work, the same authors proposed a new architecture specifically designed for the problem of online action detection (GEEST; TUYTELAARS, 2018). In this new approach, the authors assumed that, in an online action detection problem, there were two main tasks to be taken into account: the interpretation of the input frames, and the co-occurrence relation between certain actions (some actions have a tendency to occur new one another, or even in a specific sequence, for example, in a soccer game, the actions "ball pass" and "ball received" almost always happen in a sequence). From this, the authors proposed the usage of an LSTM network, since it is a model specialized in keeping temporal dependencies across multiple instances. However, since they had already shown in their previous work (GEEST et al., 2016), a traditional LSTM model did not provide good enough results. Because of this, in this work, they proposed a new *two-stream network*, where an LSTM was used to classify input frames, and another LSTM was used solely to keep track of co-occurrence relations between multiple actions. The final output of the networks was a combination of the prediction of both these models. After testing their architecture on two well-known datasets, *TVSeries* and *Breakfast*, the authors concluded that their two-stream LSTM provided noticeable improvements in comparison to a traditional LSTM.

Still, in the context of RNNs, Migze Xu and Mingfei Gao proposed a new framework for online action detection called *temporal recurrent network* (TRN) (XU et al., 2018). The model relied on a future event prediction module, combined with a traditional prediction approach to make action predictions. This model took into account not only current and past information but also predicted future information. TRNs can be seen as an extension of RNNs since they also rely on the persistence of important information in order to make predictions. The proposed architecture is composed of two parts, first, a feature-extraction module, that processes a sequence of frames to generate a new feature representation, and an optical flow (Section 2.3). Following this, the second part of the model, called *TRN cell*, consists of a recursive processing unit that receives the output from the feature-extraction module and returns a probability array corresponding to all the possible labels. This cell is the main processing structure in the TRN, and is responsible, among other things, for controlling the internal flow of information. The TRN cell is composed of three separate structures: a temporal decoder, a future gate, and a spatiotemporal accumulator (STA). The temporal decoder is the unit responsible for representing the future action predictions sequentially, the future gate takes this representation and restructures it to represent the future context in a more optimized way. Following this, the STA takes into account both the feature-representation of the input, as well as the future gate future action prediction, and returns the probabilities for all possible labels. The authors validated this framework in three different datasets: the HDD (Honda Driving Dataset), TVSeries, and Thumos14. Through their experiments, they showed that the TRN obtained results superior to the previous state of the art methods, both in early action detection, as well as in future action prediction.

3.2 Video Games in Artificial Intelligence

Video Gamess are excellent testing scenarios for AI. An interesting application of this was presented by Kangwook Lee, Hoon Kim and Changho Suh in (LEE; KIM; SUH, 2017). This work investigated the game *Grand Theft Auto V* (GTA V) as a platform for training DL models designed to predict vehicle collisions. The primary motivation for this work was that, due to the lack of real-world data on vehicle collision, DL models could not be trained effectively to deal with the problem. So the authors used the game GTA V to generate a vehicle collision dataset and used it to train three CNN models, an AlexNet, a VGG16, and a ResNet50. All these models had the primary goals of: i) detecting collisions before they happened; and ii) identifying sources of danger for collision. In the end, the authors concluded that video games have big potential as a data source for scientific research. In addition to that, they showed that their proposed approach can be very useful to increase data quantity and allow DL models to be used in real-world problems where the amount of real-world data available for training is limited since it's possible to use transfer learning techniques to pre-train a network and lower the overall data required for the model to converge.

The work proposed by (LUO et al., 2018) presents three approaches to classify game events in video game footage of Gwario, which is a Super Mario Bros clone. Its main goal is to provide an easy way for researchers to utilize video games as testing and learning platforms. The first proposed approach consists of training an AlexNet (KRIZHEVSKY; SUTSKEVER; HINTON, 2012) model from scratch with a manually labeled Gwario game dataset. Their second approach utilizes a *student-teacher* technique (WONG; GALES, 2016) to lower the amount of necessary training data and time processing. In the third approach, the work applied the *student-teacher* method combined with a pre-trained network to the UCF 101 dataset (SOOMRO; ZAMIR; SHAH, 2012) related to the game Skyrim. In such an approach, the game events come from the UCF 101 dataset. Even though the first two approaches did not perform very well, the third showed very promising results. From this, the authors concluded that a transfer learning process from a real-world dataset can allow for good results in an action classification task, in addition to lowering the data amount required and speeding up the training process.

3.3 Final Considerations

Even though online action detection in videos is a well-researched topic in the literature, there are still a lot of challenges regarding applying these techniques in the context of video games. A common trend that has wielded excellent results in real-world online action detection problems is the use of CNNs for feature extraction combined with RNNs for classification and label co-relation analysis. However, in the context of video game research, most works still use simple CNN models, that don't take into account data label co-relations, which, as shown in real-world video classification research, is a very important thing to consider. Because of this, there are still a lot of research opportunities on the problem of online action and event detection for video games.

CHAPTER 4

Investigating the Performance of Deep Neural Networks-based Approaches Designed to Identify Game Events in Gameplay Footage

4.1 Proposed Approach

This section presents the architecture of the models investigated herein to identify game events in Super Mario Bros gameplay footage. Such architecture is composed of the following modules: a *feature extractor* and a *classifier*, as shown in Figure 14.

The models differ from each other by the following factors: types of CNNs that perform the feature extraction and the game event classification; and the feature extractor input type (chunk or frame). The CNN extractor aims to automatically define a set of features that allows for producing a compact and adequate representation of the game scenes. Such feature-based representation is then presented at the classifier input so that it identifies the game event occurring in the analyzed scenes.

4.1.1 Capturing Frames and Chunks from the Gameplay Footage

The required training data samples for this work (frames and chunks) are extracted from 135 Super Mario gameplay videos with the Mario AI Framework (KARAKOVSKIY; TOGELIUS, 2012) (as shown in subsections 4.1.1.1 and 4.1.1.2). These data are then preprocessed (subsection 4.1.2) in order to generate the datasets necessary for the execution of the experiments.

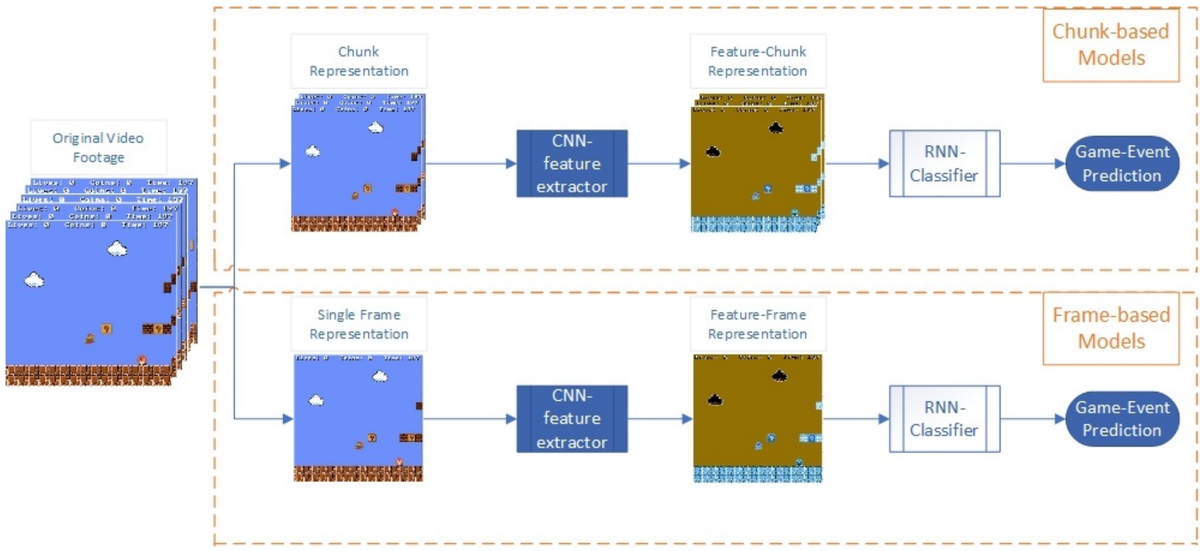

Figure 14 – General architecture of the chunk-based and frame-based models

4.1.1.1 Frame capture

The samples that make up the frame-based dataset consists of individual frames containing a single active game event (that is, frames devoid of events or otherwise containing multiple events are discarded).

4.1.1.2 Chunk capture

Inspired by work proposed by Xu et al. (XU et al., 2018), the chunks are made up of six frames and are generated from the following three-step algorithm that runs sequentially through all frames of the videos:

1. If the current frame has more than one event or no event, go to the next frame;
2. Else, produce a cluster composed of the current frame, the two frames that precede it, and the three frames that follow it;
3. Label the whole chunk (cluster) with the middle frame's label (that is, the current frame's label).

This way, a chunk is always labeled with the game event occurring at its middle frame, regardless of other events that might occur on its remaining frames. Because of this, for all chunk-based models, each chunk corresponds to a single event.

4.1.2 Frame and Chunk Pre-Processing

In order to improve the feature extraction process, before being inserted into the training datasets, both the individual frames and the frames that make up the chunks are submitted to a pre-processing that consists of the following two phases: firstly, with the purpose of reducing the feature extraction runtime, every frame is resized from its original size to 224×224 pixels, maintaining the RGB color channels; secondly, in order to allow that the CNN retrieves the most relevant features to represent the game scene, every resized frame is submitted to a pixel-wise normalization, where the values range in the interval [0,1].

4.1.3 Game Events

The Super Mario game events investigated in this work are defined by the Mario Al Framework itself as being fundamental to modeling the dynamics of a match. The game situations corresponding to these 12 game events are briefly described next:

- *EventBump:* Mario jumps having a block above its head;
- *EventCollect:* Mario collects a coin at the current level;
- *EventFallKill:* An enemy dies by falling out of the scene;
- *EventFireKill:* Mario kills an enemy by shooting fire at it;
- *EventHurt:* Mario takes damage by hitting an enemy;
- *EventJump:* Mario performs the jump action;
- *EventKick:* Mario kicks a Koopa shell;
- *EventLand:* Mario lands on the ground after having jumped;
- *EventLose:* Mario loses the game level (which happens either when this character dies or when the time to complete the current level is over);
- *EventShellKill:* Mario kills an enemy by throwing a Koopa shell at it;
- *EventStompKill:* Mario kills an enemy by jumping over it;
- *EventWin:* Mario wins the game (that is, the character successfully completed the current level).

4.1.4 The CNN Models

In order to pursue its objectives, this work firstly implements 25 distinct models to perform game event detection in Super Mario game footage. Among these models, 24 correspond to novel approaches proposed in this work and only one counts on an architecture that has already been used with the purpose of performing game event detection in-game footage of Gwario, which is a clone of Super Mario Bros. More specifically, such architecture was proposed in (LUO et al., 2018) (section 3.2).

These models are summarized in Table 1 and briefly introduced below (more details will be discussed in the following sections), where: FE represents the feature extractor (MobileNetV2, ResNet50V2, VGG16 or AlexNet); C represents the classifier (LSTM, GRU or FCL); and IR represents the game scene representation input type (frame or chunk).

Table 1 – Summary of the proposed CNN models to evaluate in this work

Model	**FE**	**C**	**IR**	**Input and Hidden Size**	**Output Size**	**Number of Parameters**
1	MobileNetV2	LSTM	Chunk	1280	12	13.127.692
2	MobileNetV2	LSTM	Frame	1280	12	13.127.692
3	MobileNetV2	GRU	Chunk	1280	12	9.849.612
4	MobileNetV2	GRU	Frame	1280	12	9.849.612
5	MobileNetV2	FCL	Chunk	1280	12	52.388.468
6	MobileNetV2	FCL	Frame	1280	12	26.174.068
7	ResNet50V2	LSTM	Chunk	2048	12	33.587.212
8	ResNet50V2	LSTM	Frame	2048	12	33.587.212
9	ResNet50V2	GRU	Chunk	2048	12	25.196.556
10	ResNet50V2	GRU	Frame	2048	12	25.196.556
11	ResNet50V2	FCL	Chunk	2048	12	71.262.836
12	ResNet50V2	FCL	Frame	2048	12	29.319.796
13	VGG16	LSTM	Chunk	4096	12	134.283.276
14	VGG16	LSTM	Frame	4096	12	134.283.276
15	VGG16	GRU	Chunk	4096	12	100.724.748
16	VGG16	GRU	Frame	4096	12	100.724.748
17	VGG16	FCL	Chunk	4096	12	121.594.484
18	VGG16	FCL	Frame	4096	12	37.708.404
19	E-AlexNet	LSTM	Chunk	4096	12	134.283.276
20	E-AlexNet	LSTM	Frame	4096	12	134.283.276
21	E-AlexNet	GRU	Chunk	4096	12	100.724.748
22	E-AlexNet	GRU	Frame	4096	12	100.724.748
23	E-AlexNet	FCL	Chunk	4096	12	121.594.484
24	E-AlexNet	FCL	Frame	4096	12	37.708.404
25	EC-AlexNet		Frame	-	12	60.689.804

The research focused on exploring the following approaches:

- New proposed models: in these unpublished models, the feature extraction and the classification are performed by distinct NNs (corresponding to the first 24 models of Table 1). More specifically, the feature extraction is carried out by a pre-trained CNN and the classification is executed by an RNN (LSTM or GRU) or an FCL, which must be trained from scratch. These models will be referred to here as *FE+C+IR*. In all these models in which the pre-trained feature extractor FE corresponds to the AlexNet, such CNN will be referred to as *E-AlexNet* (where *E* indicates that, in these models, the pre-trained AlexNet only copes with feature extraction). The design of these models were inspired by (XU et al., 2018) and by empirical tests.

- State of Art-based Model: in this model inspired by (LUO et al., 2018), the feature extractor and the classifier modules are combined into a single CNN that must be trained from scratch (corresponding to the last model of Table 1). Such an approach only uses individual frames as game scenery representation type. This model will be referred to here as *EC-AlexNet+frame*, where *EC-AlexNet* corresponds to an AlexNet that must be trained from scratch (in this case, *EC* indicates that, in such model, the training from scratch AlexNet copes with both feature extraction and classification).

4.1.5 Training Datasets

This subsection presents the datasets built to train the NNs of the models, that is, the Frame Dataset, the Chunk Dataset, the Frame-Feature Dataset, and the Chunk-Feature Dataset.

All of them are made up of 11,000 examples (obtained from 135 videos, as commented in section 4.1.1) involving the set of 12 game events presented at subsection 4.1.3. In order to optimize the NNs' training, the frames and chunks that make up such datasets undergo the pre-processing described in subsection 4.1.2.

- Frame Dataset: The data is composed of pairs ($frame_i$, $event_j$), where $event_j$ is the only game event occurring at the single frame $frame_i$. It is used to train from the scratch the AlexNet that makes up the model *EC-AlexNet+frame* and to generate the *Frame-Feature Dataset* used to train the classifier C of all models *FE+C+frame*.

- Chunk Dataset: Is made up of pairs ($chunk_i$,$event_j$), where $event_j$ is the main game event (or midle frame's event) occurring in the six frames that compose the chunk $chunk_i$ (as shown in subsection 4.1.1.2). This dataset is used to generate the *Chunk-Feature Dataset* to be used to train the classifier C of all models *FE+C+chunk*.

- Frame-Feature Dataset and Chunk-Feature Dataset: The Frame-Feature and Chunk-Feature datasets are built as follows: firstly, all training examples of both the Frame Dataset and the Chunk Dataset are presented one time to each pre-trained extractor CNN used herein (MobileNetV2, ResNet50V2, VGG16 and AlexNet). Next, the feature-based representations produced by all these CNNs from the Frame Dataset are stored in the Frame-Feature Database. In the same way, the feature-based representations produced by all these CNNs from the Chunk Dataset are stored in the Chunk-Feature Database. It is interesting to note that this *fully processing* strategy in which a set of frames that compose a chunk is presented at the feature extractors' inputs (particularly here, a set of 6 frames) is adequate in problems in which the images that represent the scenes are not complex (XU et al., 2018), as in Super Mario Bros (as presented in section 2.4.2). In fact, the simplicity of the images of Super Mario allows for obtaining very satisfactory run-times in the chunk-based models, even keeping all the frames that make up a chunk at the feature extractor input. The Frame-Feature Dataset and the Chunk-Feature Dataset are used to train the classifier C of all models *FE+C+frame* and *FE+C+chunk*, respectively. Both datasets will allow for significantly reducing the global training time of the classifier NNs in the experiments since the feature-based representation of every example (be it frame or chunk) to be presented at the classifier input can be directly retrieved from them (instead of being obtained by repeatedly submitting the example to the CNN extractor).

4.1.6 Feature Extractors

As aforementioned, this study investigates the use of five distinct CNN Feature Extractors: the pre-trained MobileNetV2, ResNet50V2, VGG16, and AlexNet, as well as the trained from scratch AlexNet (trained from the Frame-Dataset in the frame-based models). Each one of these CNN extractors has the same number of parameters and the same output size in both the frame-based and chunk-based models in which it is used. Table 2 resumes the configurations of these CNNs. In this table and from this point in this work, the pre-trained AlexNet that operates on the models only as a feature extractor and the one trained from scratch that operates on them as a feature extractor and a classifier will be referenced as E-AlexNet and EC-AlexNet, respectively.

Noteworthy here is that MobileNetV2 is a consolidated state-of-the-art CNN alternative to operate in resource-constrained environments (SANDLER et al., 2018). Further, the four remaining feature extractors used here proved to be effective in selecting features from real-world video for classification purposes (XU et al., 2018). In short (more details are shown in Table 2):

❏ *MobileNetV2:* this CNN presents a very compact architecture compared to other existing feature extractors, both in terms of the number of layers and the dimension of the weight vector. This allows for faster processing of the input data (SANDLER et al., 2018).

❏ *ResNet50V2:* it is one of the main CNN extractors used in modern computer vision methods (HE et al., 2016) due to the following fact: it includes residual blocks (composed by skip connections) that improve its performance and training convergence by mitigating the vanishing gradient problem.

❏ *VGG16:* this model is also a largely used feature extractor in modern research. Its primary characteristic is the use of very small convolutional kernels in a deep structure with a lot of layers (SIMONYAN; ZISSERMAN, 2014). As shown in Table 2, VGG16 presents the largest architecture among the feature extractors used herein.

- *AlexNet:* it is one of the main architectures responsible for popularizing CNNs in computer vision research (Aloysius; Geetha, 2017), having improved the CNN learning ability through strategies to optimize parameters and training. Table 2 resumes the configurations of the feature extractor layers of AlexNet (E-AlexNet) used in this work.

Table 2 – Summary of CNN Models

	Pre-Trained Weights	**Number of Parameters**	**Number of Features (Output Size)**
MobileNetV2	ImageNet	2.257.984	1280
ResNet50V2	ImageNet	23.564.800	2048
VGG16	ImageNet	134.260.544	4096
E-AlexNet	ImageNet	43.859.328	4096
EC-AlexNet	None	60.689.804	-

It is important to point out that the four pre-trained CNNs used to perform feature extraction in the 24 first models presented in Table 2 were trained from the ImageNet dataset (the pre-trained weights of these CNNs are all available in PyTorch[1]). Each one uses the number of features presented at the fourth column of the table to produce the feature-based game scene representations stored in the Frame-Feature and in the Chunk-Feature datasets, as explained in subsection 4.1.5.

On the other hand, the CNN EC-AlexNet used in the last model of Table 2 is trained from scratch from Frame Dataset (section 4.1.5). As both the feature extraction and the classification modules of such CNN are trained together, they will be more detailed in section 5.1.5. Further, the data presented in Table 2 refer exclusively to the feature extractor layers of such CNN after being trained (the data related to the classifier layers are presented in Table 1).

4.1.7 Classifiers

This stage of research investigates the performance of RNNs, FCL and AlexNet in the classification process. The RNNs used herein are LSTM (HOCHREITER; SCHMIDHUBER, 1997) and GRU (CHO et al., 2014), being both composed of two fully-connected layers. The activation function of the input and output layers of both RNNs are ReLU and Softmax, respectively.

The FCL-based classifier used in the experiments is composed of four layers (being then similar to the classifier layers used in AlexNet (KRIZHEVSKY; SUTSKEVER; HINTON,

[1] <https://pytorch.org/vision/stable/models.html>

2012)). The activation function used in the hidden (three) and output layers are ReLU and Softmax, respectively. The activation signals to be presented at the classifier input corresponds to the output signals produced by the feature extractor CNN (that is, the game scene representation produced by such CNN). Then, the number of neurons of the input and hidden layers of the classifiers is equal to the number of features produced by the feature extractor (as shown in Tables 2 and 1). Still, the number of neurons of the classifier output layer corresponds to the quantity of game events taken into consideration, which here is equal to 12 (as presented in Table 1).

It is interesting to point out that, as EC-AlexNet used in the 25th model (inspired by (LUO et al., 2018)) has to perform both feature extraction and classification, it must be trained from scratch so as to be able to perform both processes. The training of the model *(EC-AlexNet) + Frame* is performed from examples provided by *Frame Dataset* (section 4.1.5). On the other hand, in the unpublished models *FE+C+IR* proposed here (subsection 4.1.4), in which the feature extraction is performed by pre-trained CNNs, only the classifier NNs must be trained. More specifically, the training of the models *FE+C+frame* and *FE+C+Chunk* are performed from examples retrieved, respectively, from *Frame-Feature Dataset* and from *Chunk-Feature Dataset* (section 4.1.5).

The optimizer and loss parameters used in all training sessions performed in this work were Adam (KINGMA; BA, 2015) with learning rate equal to 0.0005 (the same proposed in (XU et al., 2018)) and Categorical Cross Entropy, respectively. The training of all NNs was limited to a maximum number of 100 epochs (the final parameters correspond to those produced in the epoch that generated the weights which presented the best results in terms of test loss). However, it is important to note that each training session was interrupted whenever the test loss did not improve for five consecutive epochs.

4.2 Experiments and Results

The purpose of the experiments is to evaluate the performance of the models to identify game events in gameplay footages depicting Super Mario Bros runs (through Mario AI Framework (KARAKOVSKIY; TOGELIUS, 2012)).

For this, the experiments are carried out in 3 distinct test scenarios: 1) The first one evaluates the individual performance of all 24 *FE+C+IR* models and uses the obtained results to perform a statistical comparison among them (for this, the models are grouped according to the specifics of their individual architectures); 2) The second test scenario has as its purpose to compare the performance of the following two trained from scratch *CNN+frame*-based models: the *AlexNet+frame* inspired by (LUO et al., 2018) (25-th model) and the *Winner_CNN+frame* (26-th model), where *Winner_CNN* corresponds to the feature extractor *FE* that composes the best *FE+C+IR* model found in the first test scenario; 3) Finally, the third test scenario aims to compare the performance between the best *FE+C+IR* and *CNN+frame* models obtained in the first and second test scenarios, respectively.

In order to maintain fairness in the comparison among the 26 models, they were all trained and tested from the same datasets and configurations. Such comparison is carried out through the following statistical results: accuracy mean standard deviation (σ) and loss values calculated from the 5-Fold Cross-Validation method, in which the data set is divided into five folds (each one composed of 2200 random examples). In each iteration, four folds (approximately 8800 examples) are used to train the models and one fold (about 2200 examples) to test them. The parameters used to evaluate the performance of each model are: the training runtime and the classification success rate (accuracy). In order to carry out the comparative analysis among the models, they are divided into groups based on the type of classifier, the game scene representation (chunks or frames) and weather they are based on pre-trained CNNs or not.

All experiments were executed on the graphics processing units provided by Google Colab platform[2] (Tesla K80 with 12GB RAM) and the source code related to this work is available in Github[3].

4.2.1 Test Scenario 1

In order to evaluate the *FE+C+IR* models, in this first test scenario they are divided into the following groups (according to the specifics of their respective architectures):

- ❏ **Group 1:** CNN + RNN + Chunk (models *1*, *3*, *7*, *9*, *13*, *15*, *19* and *21*);
- ❏ **Group 2:** CNN + FCL + Chunk (models *5*, *11*, *17* and *23*);
- ❏ **Group 3:** CNN + RNN + Frame (models *2*, *4*, *8*, *10*, *14*, *16*, *20* and *22*);
- ❏ **Group 4:** CNN + FCL + Frame (models *6*, *12*, *18* and *24*).

The results of this first test scenario are presented in Table 3.

As demonstrated in such table, the eight models of Group 1 (*FE+RNN+chunk*) presented the best accuracy results. It can be explained by the combination of the following factors: firstly, in Super Mario, the game events are often spread throughout multiple frames; secondly, the RNNs have the ability to keep temporal and spatial information about the environment on the network throughout the processing of successive instances (section 2.4.2); finally, in the chunk-based models, the classifier input corresponds to a combination of multiple frames which represents recent game scenes.

Consequently, the aforementioned ability of an RNN to interrelate information received over time makes it very suitable to identify the main game event occurring in the multiple

[2] <https://colab.research.google.com/>
[3] <https://github.com/matheusprandini/dnns-game-events>

Table 3 – First Scenario Results

Group	**Model**	**Mean Accuracy**	**Std. Dev. Accuracy**	**Mean Loss**	**Std. Dev. Loss**	**Training Time (per epoch)**
6***1**	1	93.10%	0.5%	0.2367	0.239	6s
	3	92.13%	0.5%	0.2516	0.252	6s
	7	91.49%	0.4%	0.2959	0.296	10s
	9	90.67%	0.7%	0.3151	0.315	6s
	13	74.65%	7.0%	0.8313	0.198	34s
	15	70.25%	4.0%	0.8641	0.093	28s
	19	48.56%	0.6%	1.2754	0.260	42s
	21	43.10%	0.4%	1.5088	0.228	37s
3***2**	5	86.1%	0.2%	0.4765	0.477	7s
	11	87.73%	1.0%	0.4141	0.029	8s
	17	36.96%	0.3%	1.5359	0.009	11s
	23	42.09%	0.9%	1.4852	0.015	10s
6***3**	2	73.87%	0.7%	0.7147	0.019	5s
	4	73.67%	0.6%	0.7287	0.010	5s
	8	71.59%	0.7%	0.7997	0.800	6s
	10	70.70%	0.6%	0.8115	0.811	6s
	14	63.39%	0.8%	1.0363	0.018	34s
	16	63.67%	0.9%	1.0343	0.012	17s
	20	39.87%	0.8%	1.4823	0.005	35s
	22	39.40%	0.6%	1.4908	0.010	19s
3***4**	6	68.47%	0.2%	0.9100	0.021	5s
	12	69.92%	1.0%	0.8398	0.022	6s
	18	43.12%	3.0%	1.5085	0.012	6s
	24	39.30%	0.2%	1.5028	0.011	7s

frames represented at its input. Further, the best results in terms of accuracy obtained by Group 1 compared to those of Group 2 prove the superiority of RNNs over FCLs as classifier tools in the problem addressed (since the models of both groups differ from each other only for the type of the classifier used).

Comparing now the accuracy results between Groups 1 and 3, and between Groups 2 and 4, it is possible to conclude that the chunk-based models present the best performance to identify game events in Super Mario scenes than the frame-based ones (since the models of Groups 1 and 3, as well as those of Groups 2 and 4, differ from each other only in terms of the game scene representation).

With respect to the training run-time, Table 3 corroborates the following theoretical expectation: the simplicity of the GRU architecture in relation to the LSTM results in a shorter training time and a slight lower accuracy of the GRU-based models compared to the LSTM-based models. In fact, as GRU has a more concise architecture than LSTM, it tends to be less accurate than this later. However, its structural simplicity is very adequate to deal with problems involving images with a moderate level of complexity (such as the sprite-based images of Super Mario), since it produces classification results with very satisfactory accuracy in a much shorter execution time.

Concerning the feature extractors, VGG16 and E-AlexNet presented the worst performance results both in terms of training time and accuracy. This is due to the fact that these CNNs produce a set of features larger than those produced by the other ones, which substantially increases the size of the model, causing overfitting (since there is a greater number of parameters to be adjusted in the training process).

Finally, the global analysis of the results presented here confirms that the first model of Group 1, that is, *MobileNetV2+LSTM+Chunk*, presents the best performance among the 24 new *FE+C+IR* models proposed in this work, achieving, approximately, a mean accuracy of 93.1% and a mean loss of 0.2367. This model will then be used in the comparative analysis to be carried out in *Test Scenario 3*.

4.2.2 Test Scenario 2

As provided in the beginning of section 4.2, considering that the first test scenario proved the performance superiority of model *MobileNetV2+LSTM+Chunk* among the 24 *FE+C+IR* models investigated, the *Winner-CNN* is *MobileNetV2*. Then, in this second test scenario, the model *EC-MobileNetV2+frame* (26-th one) is implemented and trained from scratch from the same *Frame Dataset*. The trained model is then confronted with the trained from scratch *EC-AlexNet+frame* state-of-art model (LUO et al., 2018) (25-th model). In function of their architectures, both models define then the following fifth and last group to be investigated in this work:

- ❑ **Group 5:** composed by the trained from scratch *EC-CNN+frame* models (models 25 and 26).

Table 4 shows the results of these models. They indicate a slight performance superiority of the *EC-AlexNet+frame* model over the *EC-MobileNetV2+frame* model. Although *EC-MobileNet* proved to be about twice faster than *EC-AlexNet* in terms of training time, the latter stands out for achieving an accuracy rate approximately 4% superior to the former. These results are theoretically coherent, since the number of parameters of EC-AlexNet and EC-MobileNet are 60.689.804 and 2.273.356, respectively.

Finally, comparing the accuracy results between the models of Group 1 that uses MobileNet as *FE* and *EC-MobileNetV2+frame* investigated in this second test scenario, it is possible to conclude that GRU and LSTM (which, together with MobileNet, make up the models of Group 1) enormously contribute to increase the ability of the MobileNet-based models to identify game events in Super Mario Broth gameplay footage.

Table 4 – Second Scenario Results

Group	**Model**	**Mean Accuracy**	**Std. Dev Accuracy**	**Mean Loss**	**Std. Dev Loss**	**Training Time (per epoch)**
2*5	25	84.2%	1.0%	0.5405	0.540	390s
	26	80.4%	0.2%	1.0454	0.108	169s

4.2.3 Test Scenario 3

The purpose of the third test scenario is to compare the best *FE+C+IR* model proposed in this work (*MobileNetV2+LSTM+Chunk*, or model 1, evaluated in *Test Scenario 1*), with the best *EC-CNN+frame* model (*EC-AlexNet+frame*, or model 25, evaluated in *Test Scenario 2*).

Table 5 shows the performance results - in terms of accuracy and training runtime - obtained by each model, as well as the statistical results (mean and standard deviation) that allow for making a comparison between them.

Table 5 – Third Scenario Results

Group	**Model**	**Mean Accuracy**	**Std. Dev Accuracy**	**Training Time (per epoch)**	**Total Training Time**	**Inference Time**
1***1**	1	93.1%	0.5%	6s	240s	2s
1***5**	25	84.2%	1.0%	390s	3900s	17s

The results prove the significant superiority of the performance both in terms of training time and accuracy of the best model proposed in this work (model 1) compared to that conceived in the state-of-the-art (model 25). In fact, in addition to the accuracy of this best new model being 10% greater than that of the model proposed in (LUO et al., 2018), its training time (per epoch), total training time, and total inference time (corresponding to the mean inference time of the folds used for testing) were, respectively, 6500%, 1625% and 850% faster than that observed in such related work.

These results can be explained by the fact that this latter has to train both the feature extraction and the classification layers of the EC-AlexNet, whereas in the *MobileNetV2+LSTM+Chunk* model, only the LSTM has to be trained (since MobileNetV2 is a pre-trained CNN). It is interesting to note that each epoch of the EC-AlexNet training requires about 390 seconds (or 6 minutes and 30 seconds) to be executed, whereas each epoch of the LSTM training spends only six seconds to be concluded. Besides this, the inference time proves that the best model of this work is more suitable considering an online setting.

To compare the models *MobileNetV2+LSTM+Chunk* and *EC-AlexNet+frame*, Test Scenario 3 also evaluates their performance in detecting each event individually, as well as in the detection of events that belong to a certain class. Table 6 presents the classification accuracy of both models to identify each one of the 12 game events considered herein individually.

In Table 6 is shows that the accuracy of *MobileNetV2+LSTM+Chunk* surpasses, ties and falls short the accuracy of *EC-AlexNet+frame* for 7, 4 and 1 of the events, respectively. Further, considering that these events, in function of their peculiarities, can be divided into classes, it is possible to calculate - from the accuracy results presented in Table 6 - the following success rates of the model *MobileNetV2+LSTM+Chunk* compared to the model *EC-AlexNet+frame* (the success rate related to a given class indicates the percentage of events of that class for which model 1 presented accuracy at least equal to that of model 2):

- Class of the more frequent events: *Bump*, *Jump*, *Land*, *StompKill*: *75%*
- Class of the less frequent events: *Collect*, *Fallkill*, *Firekill*, *Hurt*, *kick*, *Lose*, *Shellkill*, *Win*: 100%
- Class of events independent of Mario: *Fallkill*: 100%
- Class of events that can be originated from more than one sequence of actions: *Collect*, *Fallkill*, *Firekill*, *Hurt*, *kick*, *Lose*, *Shellkill*, *Win*, *Fallkill*: 100%

Table 6 – Acuracies of models 1 and 25 for each game event

	Model 1	Model 25
Bump	**95.24%**	68.78%
Collect	**86.01%**	72.72%
FallKill	**51.72%**	13.80%
FireKill	**88.24%**	61.54%
Hurt	**89.13%**	31.91%
Jump	**96.64%**	83.11%
Kick	**50.00%**	**50.00%**
Land	**96.21%**	94.01%
Lose	**33.33%**	**33.33%**
ShellKill	**42.86%**	**42.86%**
StompKill	89.82%	**91.67%**
Win	**100%**	**100%**

Both analysis performed in *Test Scenario 3* confirms the superiority of the best model *MobileNetV2+LSTM+Chunk* produced in *Test Scenario 1* compared to the best model *EC-AlexNet+frame* produced in *Test Scenario 2*. Then, *MobileNetV2+LSTM+Chunk* proved to be the best model investigated herein to identify game events occurring in Super Mario Bros gameplay footage.

Finally, it should be noted that in the state-of-the-art work (LUO et al., 2018) that inspired the implementation of the *EC-AlexNet+frame* model in this work, the authors report having obtained an accuracy of approximately 94% (different then from the mean accuracy of 84% shown in Table 4 for the model *EC-AlexNet+frame*). It can be explained by the fact that, here, the model was trained to play a different game (Mario, instead of Gwario), from datasets that, distinctly from (LUO et al., 2018), do not include frames devoid from events.

4.3 Final Chapter Considerations

This part of the work investigated 26 DL-based models to identify game events occurring in Super Mario Bros gameplay footage. Among them, 24 correspond to novel approaches in which: a pre-trained CNN (MobileNetV2, ResNet50V2, VGG16, or AlexNet) performs feature extraction; an FCL or an RNN (LSTM or GRU) executes the game event classification; and the game scenes are represented either by frames or by chunks. In the other 2 models, a trained from scratch CNN (AlexNet or MobileNetV2) deals with both feature extraction and classification, and the game scenes are represented by frames. In these last two models, such CNNs were selected due to the following facts: *AlexNet* had already been tested in (LUO et al., 2018) to detect events in gameplay footage of Gwario, and *MobileNetV2* makes up the model which obtained better performance among the first 24 *FE+CNN+IR* models initially evaluated. At the end of all test scenarios, the new model *MobileNetV2+LSTM+chunk* proposed herein proved to be the best among all the 26 models investigated.

CHAPTER 5

DL Models for Performing Multi-Instance Multi-Label Event Classification in Game Footage

5.1 Proposed Approach

This chapter details the models investigated in this study to tackle multi-label game event classification in game-play footage of *Super Mario Bros.* These models are composed of a CNN and a deep classifier NN, the first of which aims to automatically extract the most relevant features to be used to represent the game scenes, while the last is used to identify the events happening in such scenes.

The architecture of these models was created taking into consideration the following results obtained in preliminary studies: 1) In the state-of-art work (Chapter 4), where various models were investigated to perform single event detection in game-play footage of *Super Mario Bros*, and the winner model was made of a feature extractor MobileNetV2 associated with a classifier LSTM (FARIA et al., 2022); 2) In DeepMIML the authors proposed the DeepMIML 2D Sub-concept Layer to perform multi-label classification in images. Then, the architecture of the models proposed herein is composed of the following modules:

- ❏ An automatic data generator: the data generated represent the game scenes and were retrieved from the Mario AI Framework; (KARAKOVSKIY; TOGELIUS, 2012);
- ❏ A feature extractor module: the CNN used to automatically performs feature extraction is a fine-tuned MobileNetV2 produced in this work, which corresponds to the original pre-trained version of MobileNetV2 (SANDLER et al., 2018) retrained from a dataset composed of frames produced from the samples produced by the automatic data generator.

- ❏ A multi-label classification module: two distinct deep classifier NNs were investigated: 1) LSTM (HOCHREITER; SCHMIDHUBER, 1997); 2) the classifier sub-concept Layer of the DeepMIML NN (FENG; ZHOU, 2017).

Briefly, the multi-label game event classification process carried out in this study can be resumed as follows: firstly, the Mario AI Framework is used to capture gameplay footage from games involving human and automatic player agents. The structure of such footage consists of frames containing *at least one* event retrieved from the videos which compose the footage, and a set of CSV files storing the labels corresponding to these frames.

After that, all frames are submitted to a pre-processing and added to a dataset, which is then balanced for the purpose of treating the class imbalance problem. The frames of such dataset are used as follows: those containing just *one* event are grouped into a dataset named *Frame Dataset*, which will be used to train and produce a fine-tuned version of the original feature extractor MobileNetV2; each frame containing *at least one* event is clustered with their surrounding frames into structures named *chunks*, which are stored in a dataset referred here as *Chunk Dataset*. In this way, each chunk represents, in fact, a set of game scenes.

In order to train the multi-label game event classifier deep NN, each training chunk selected in the *Chunk Dataset* is first submitted to the fine-tuned extractor MobileNetV2, and the feature-based representation produced by this extractor for that chunk is then presented at the input of the classifier deep NN to be trained. It is interesting to point out that this chunk-based representation for the game scenes was also inspired by the state-of-art work (FARIA et al., 2022), in which such representation produced much better results than the frame-based one. All the steps of these processes are explained in detail in the following sub-sections.

5.1.1 Data Generation

In order to cope with the objectives of this work, the authors in (KARAKOVSKIY; TOGELIUS, 2012) had to implement a script, based on the Mario AI Framework, to automatically and easily generate, from gameplay footage, adequate data to train the proposed models. This need arose from the following facts: firstly, as such objectives involve working with a multi-label classification problem, it was necessary to count on a significant number of data instances (frames representing game scenes) containing, at least, two or more game events (or labels). Secondly, as Mario AI Framework generates about 270 variables (such as game events, coordinates, and others) to describe each frame, the script has to be able to select, among them, those which are relevant to label the frames.

In this way, the implemented script allows for control over both the *framerate* of the footage, which represents how many times the game scene refreshes in the game in a second, and which events end up being represented in the data. Further, the footage from which the script produced the data to be pre-processed and used as training instances is composed of 75 videos, which record the set of the same 15 levels of Super Mario bros played by 5 distinct players, among which 4 are humans and one is the automatic A* algorithm - based agent available in the Mario AI Framework.

Moreover, the footage from which the script produced the data to be pre-processed and used as training instances is composed of 75 videos, which record the set of the same 15 levels of Super Mario bros played by 5 distinct players, among which 4 are humans and one is the automatic A* algorithm - based agent available in the Mario AI Framework. The footage is captured at 30 frames per second, and the following events were included in the data:

- *EventBump:* Mario bumps his head on a block after jumping;
- *EventCollect:* Mario collects a coin;
- *EventFallKill:* an enemy dies by falling out of the scene;
- *EventFireKill:* Mario kills an enemy by shooting fire at it;
- *EventHurt:* Mario takes damage after being hit by an enemy;
- *EventJump:* Mario performs a jump;
- *EventKick:* Mario kicks a Koopa shell;
- *EventLand:* Mario lands on the ground after having jumped;
- *EventLose:* Mario loses the game level, can be caused by the player dying or the time running out;
- *EventShellKill:* Mario kills an enemy by kicking a Koopa shell at it;
- *EventStompKill:* Mario kills an enemy by jumping on top of it;
- *EventWin:* The player Completes the current level.

5.1.2 Data Pre-processing

The pre-processing process has as its purpose to refine the row data with *at least one event* produced by the generator script in such a way as to make them more suitable to improve the training quality of the proposed models. In short, the following three pre-processing proceedings were implemented in this stage:

- **Retrieval of the relevant variables**: as the Mario AI Framework uses about 270 variables to describe each frame, the first proceeding has to select from them only the 12 that correspond to the events used to label the training data;

- **Definition of a correct capture window**: Super Mario AI Framework annotates each event occurrence in the frame associated with the exact moment it starts. However, for some kinds of events, such frames are not adequate, since, visually, the effect of triggering that event will only be noticed in a later frame. Due to this, the second implemented proceeding has to define a new adequate custom capture window for each kind of event label. For example, specifically for the event *EventBump*, even though the system annotates its occurrence in a given frame f, the second proceeding will annotate it in the frame that succeeds f.

- **Frame resizing**: Then, in order to make the dimension of the frames suitable to the deep NNs that were used, the third proceeding resized each frame from its original size to 224x224 pixels, and a pixel-wise normalization was applied. All three RGB color channels were kept.

5.1.3 Training Datasets

After being generated by the generator script (Sub-section 5.1.1) all labeled row frames with *at least one* event were pre-processed (Section 5.1.2 and stored in a dataset. Next, as the dataset was originally very unbalanced due to the fact that some events naturally occur much more than others in the footage from which the data were retrieved, it had to be submitted to an adequate balancing strategy. As, due to intrinsic characteristics of the Super Mario bros game, about 98% of the frames with *at least one event* are single labeled instances (that is, present just *one* event), it was possible to use a simple oversampling balancing in which the frames belonging to the under represented classes were randomly duplicated, until a more even distribution was reached. In the end, this pre-processed and balanced dataset was composed of $10,000$ examples, among which only 218 were multi-labeled frames (that is, presented more than one event).

It means that the multi-label challenge faced by this work arises from the fact that the data that will be processed consist of frame groupings (due to the chunk-based game scene representation that is used), which, indeed, are multi-labeled instances. The construction of both training datasets (*Frame Dataset* and *Chunk Dataset* is presented in the following
.

5.1.3.1 Frame Dataset

In order to cope with the objective of improving the feature extraction process, here the original MobileNetV2 pre-trained from ImageNet (SANDLER et al., 2018), had to be retrained, in such a way as to produce a CNN fine-tuned extractor. To speed up such retraining, it was performed through the *Frame Dataset*, which is composed exclusively of the 9, 782 frames with a single label present in the pre-processed and balanced dataset.

5.1.3.2 Chunk Dataset

This is the main dataset used for training and testing the models proposed in this work. It differs from the pre-processed and balanced dataset for having as instances, instead of single frames, groups of sequential frames, called **chunks**. Each chunk C_f of the *Chunk Dataset* is generated through the following algorithm that is applied to every frame f of the pre-processed and balanced dataset:

1. Take a frame f from the dataset;
2. Find f in the sequence of row frames produced by the generator script; in the same sequence, take the 3 frames that precede f and the 3 frames that follow it (it is interesting to note that such frames can be devoid of the event);
3. Apply the same pre-processing described in 5.1.2 to all the frames taken in the previous step that presents no event (which had not yet been pre-processed, since they do not belong to the pre-processed and balanced dataset). Note that all the remaining row frames taken in the previous step (that is, those with at least one event) have already been pre-processed since they appear in the pre-processed and balanced dataset;
4. Build the chunk C_f (containing 7 frames) by grouping, in the same order they appear in the sequence produced by the generator script, the 3 frames that precede f, f itself, and the 3 frames that follow f (all of them in their pre-processed format);
5. Label the chunk C_f with the set of labels of its frames.

Then, a chunk is a group of sequential pre-processed frames labeled with the set of labels corresponding to these frames. As previously mentioned, such a structure was used because it produced better accuracy in performing label classification in footage of Super Mario bros than the frame-based one (FARIA et al., 2022).

In addition to that, since the chunks receive the labels of their multiple frames, among the 10, 000 chunks that make up the Chunk Dataset, 3, 791 correspond to multi-labeled examples (with two or more labels), which represents a significant increase in the number of multi-labeled instances available to train the models proposed in this work (compared to the few quantities of multi-labeled instances present in the pre-processed frame-based dataset (which is equal to 217).

5.1.4 Feature Extraction

Concerning the feature extraction process, as aforementioned, the main objective here is to produce a fine-tuned version of the original MobileNetV2 pre-trained on the ImageNet dataset (SANDLER et al., 2018). The following reasons motivated to select MobileNetV2 as the basis of the feature extraction process: firstly, it presents a very compact architecture compared to other existing feature extractors, both in terms of the number of layers and dimension of the weight vector, which allows for faster processing of the input data; secondly, MobileNetV2 has proven its ability to perform feature extraction in the winning model obtained in the preliminary studies carried out in (FARIA et al., 2022)), which performs single event classification in Super Mario bros game-play footage.

In this way, here this original pre-trained MobileNetV2 was retrained from the *Frame Dataset* presented in Subsection 5.1.3. Such a strategy is based on the following fact: the ImageNet dataset, from which the original MobileNetV2 was trained, is composed of real-world images that differ a lot from the visual aspects of Super Mario's game scenes. Then, the retraining of MobileNetV2 from the *Frame Dataset*, whose instances correspond to frames reporting the real visual aspects of the game scenes, will allow for producing a fine-tuned feature extractor able to improve the performance of the proposed models.

For the sake of clarity, as explained in the introduction of Section 5.1, still inspired by the best model obtained in (FARIA et al., 2022), in this study the training of the feature extractor and the classifier deep NNs was performed separately, as follows: initially, it is performed the training that produces the fine-tuned MobileNetV2; next, in order to train the multi-label game event classifier deep NN, each chunk selected in the *Chunk Dataset* is submitted to the fine-tuned extractor MobileNetV2, and the feature-based representation produced by this extractor, for that chunk, is then presented at the input of the deep classifier NN.

For this, in order to be used to train the deep NN classifier, the architecture of the trained fine-tuned MobileNetV2 was shortened as follows: its fully connected layer, responsible for classification, was removed (in such a way as to keep only the extractor layers intact in the architecture). This way, such shortened architecture will be able to produce, at its extractor output layer, the feature-based representation of each chunk that must be presented at the input layer of the deep classifier NN that must be trained.

5.1.5 Classifiers

In order to find an adequate NN to perform MIML classification in game-play footage, this study investigates two distinct deep classifiers NN: the sub-Concept layer of the Deep-MIML NN (FENG; ZHOU, 2017) and the LSTM (HOCHREITER; SCHMIDHUBER, 1997).

The reasons for the such choice are the following: concerning the Sub-Concept layer of the DeepMIML network, as resumed in Subsection 2.2.1, it corresponds to the classifier portion of the DeepMIML NN architecture, which was conceived to operate with MIML problems. The ability of such a classifier layer to automatically generate the sub-concepts that will help to identify each label makes it quite suitable for dealing with MIML problems. Therefore, taking into consideration that the chunks - used here as the basis for representing the game scenes - are composed of multiple frames with multiple labels, it allows for including the challenge faced in this work in the class of problems MIML (ZHOU et al., 2012).

With respect to the LSTM, it was selected for the following main reasons: firstly, because it proved to be very effective as a classifier NN in the winner model obtained in the preliminary studies related to single event classification in Super Mario bros gameplay footage carried out in (FARIA et al., 2022). Such a good result is due to the fact that, in LSTM, the intermediary neurons are replaced with memory cells that allow for keeping the persistence of relevant information in the NN across the processing of different instances. Secondly, in the work that proposed the Deep MIML (FENG; ZHOU, 2017), by way of evaluating it, the authors compared its performance with that of LSTM, using as a case study a multi-label image classification problem. By a small margin, the LSTM performed better than the Deep-MIML. Finally, the LSTM is one of the most successful RNNs used in the literature until today.

5.2 Experiments and Results

The purpose of this section is to present the experiments carried out to build and evaluate the performance of the models proposed in this paper with the purpose of performing MIML classification in-game footage, as well as the obtained results.

For this, the following subsections present the experiments and results related to the construction of the fine-tuned feature extractor and of the classifier modules, respectively.

5.2.1 Evaluations with the Feature-Extractors

This subsection has as its purpose to present the creation of the shortened fine-tuned MobileNetV2 conceived with the purpose of operating as a feature extractor in the models proposed in this paper (Section 5.1.4), as well as to perform a performance comparison between the such fine-tuned extractor and the original pre-trained MobileNetV2 from which the former was built up.

Such shortened fine-tuned extractor NN was created and evaluated according to the following steps: 1) Firstly, a complete architecture (feature extractor layers + classifier layers) of this fine-tune extractor NN was built from the re-training, on the *Frame Dataset* presented in Section 5.1.3.1, of the complete architecture (feature extractor layers + classifier layers) of the original MobileNetV2 pre-trained on the ImageNet dataset; 2) In order to perform a performance evaluation between the fine-tuned MobileNetV2 obtained in the *step 1* and the original MobileNetV2, both were submitted to a validation test from *Frame DataSet*; 3) Finally, the classifier layers of the fine-tuned MobileNetV2 obtained in the *step 1* were removed, in such a way as to keep only the extractor layers, which correspond to the shortened fine-tuned MobileNetV2 to be used as feature extractor in the models proposed herein. It is interesting to point out that this shortened fine-tuned MobileNetV2 has the same number of features (at the output extract layer) and parameters as the extractor part of the original pre-trained MobileNetV2 (SANDLER et al., 2018), that is: 1.280 features and 2,257,984 parameters, as shown in Table 7.

It is also interesting to note that, in steps *1* and *2*, both NNs were implemented on Keras, and the fine-tuning training, as well as the validation, were performed with a 5-fold cross-validation method for up to 100 epochs, or until there was no improvement on the loss for five consecutive epochs. The optimizer used was Adam (KINGMA; BA, 2015), the loss parameter was categorical cross-entropy and the learning rate was equal to 0.0005, following what was used in (FARIA et al., 2022).

Table 7 – Feature-Extractor Models Overview

	Pre-Trained Weights	Training Dataset	Output Size
MobileNetV2	ImageNet	-	1280
MobileNetV2-FineTuned	ImageNet	Frame Dataset	1280

Table 8 shows the results obtained in step *2*, which prove that, in fact, the fine-tuned MobileNetV2 proposed in this work (accuracy 90.81% and Mean Loss $0,2831\%$) performs much better than the original MobileNetV2 (accuracy 74.69% and Mean Loss $0,7021\%$). This makes sense, considering that ImageNet, from which the original MobileNetV2 was trained, is a dataset comprised of real-world images, whereas the *Frame Dataset* used to train the fine-tuned MobileNetV2 is composed of images related to the problem faced here. Then, the fine-tuning definitely proved to be a good approach to provide a feature representation more suited to the desired game aesthetic.

Table 8 – Mean Accuracy and Loss Results

	Mean Accuracy	Mean Loss
MobileNetV2	74.69%	0.7021
MobileNetV2-FineTuned	90.81%	0.2831

Taking into consideration these favorable results, the shortened and fine-tuned MobileNetV2 - from this point on, referred to just as *Fine-tunned MobileNetV2* - was defined as a feature extractor NN of the proposed models.

5.2.2 Investigation with Classifiers

As explained before, this work proposes two distinct models to perform MIML event classification in-game footage: 1) a first one, combining the fine-tunned MobileNetV2 as a feature extractor and the Sub-concept layer of the DeepMIML as a classifier, named *fine-tunned-MobileNetV2 + DeepMIML-SubConcept-Layer*; 2) and a second model combining the Fine-tunned MobileNetV2 as a feature extractor and LSTM as a classifier, named *Fine-tunned-MobileNetV2 + LSTM.*

Then, concerning the first model, it is interesting to point out that the fine-tunned MobileNetV2 plays the role of the feature-based-instance generator module mentioned in Section 2.2.1. This way, in both models, the Fine-tunned MobileNetV2 provides the feature-based representation for every chunk from the *Chunk Dataset* that will be used to train the classifiers, as mentioned in Section 5.1.4.

Table 9 resumes the information related to both classifiers. Obviously, the 12 labels mentioned in the such table refer to the 12 events described in Section 5.1.1. Again, both networks were implemented on Keras, and a 5-Fold Cross-Validation method, with a 0.0005 learning rate, was used for training and testing. The chosen optimizer was *dadelta* (ZEILER, 2012). As this works copes with MIML setting, the metrics used in the experiments for evaluating the performance of both models were: *Mean Average Precision* (mAP), *F1-Score* (F1) and *Hamming Loss.* The results obtained in the experiments are presented in Table 10.

Table 9 – Classifiers Models Overview

	Pre-Trained Weights	Training Dataset	Labels
DeepMIML Sub-Concept Layer	-	Chunk Dataset	12
LSTM	-	Chunk Dataset	12

Table 10 – Evaluation of the Investigated Classifiers

	mAP	Hamming Loss	F1
DeepMIML Sub-Concept Layer	87.92%	0.014	0.91
LSTM	89.33%	0.011	0.93

The results show that LSTM (Table 10) performs a little better than MIML in all evaluated metrics. In fact, the *Mean Average Precision*, the *Hamming Loss* and the *F1-Score* produced by the former and the latter were, respectively: 89.33%, 0.011%, 0.93, and 87.92%, 0.014%, 0.91.

It is interesting to note that the same slight performance superiority of the LSTM-based model over the MIML-based model observed here, dealing with MIML classification in game footage, was also observed in (FENG; ZHOU, 2017), where the authors investigated both LSTM and MIML-based models dealing with image object detection in the Common Objects in Context (COCO) dataset.

5.3 Chapter Considerations

In order to extend the current state of art in the domain of performing MIML event detection in gameplay footage, this stage of this study proposed two distinct DL models, in which the instances are represented by chunks and the feature extraction is done by a fine-tuned MobileNetV2. With respect to the deep classifier NN, the first model uses the sub-concept layer of the DeepMIML, whereas the second one uses the LSTM. The Super Mario Bros game was used as a case study. The results proved that both models succeeded in the task they dealt with, even though the *Fine-tunned-MobileNetV2 + LSTM* model performed a little better than the *Fine-tunned-MobileNetV2 + DeepMIML-SubConcept-Layer* model.

Chapter 6

Conclusion

As stated in section 1.2, this work sought to answer the following questions: i) is it possible, through DL and CV techniques, to generate a viable framework for extracting relevant information from video games that can replace the need to access the *Game Engine*? ii) can methods designed for real-world scenarios be adapted to work in-game scenes?; iii) if so, among a large number of different methods in the literature, which ones are more suited for identifying events in-game scenes? iv) what is the best strategy to deal with multi-labeled frames?

This work proposed two main contributions related to the problem of classifying game events from gameplay footage. The first one was an investigation of 26 different DL-based models to identify game events in the game Super Mario Bros, structured in such a way as to separate a DL model into two networks, a CNN for feature-extraction, and an RNN for classification, in addition to applying a frame clustering approach in the dataset to improve overall performance.

The second one focused on structuring the game-event classification problem into a MIML framework, using a new proposed fine-tuning strategy to train a CNN to perform feature-extraction, and then using both a LSTM and a Deep MIML network to perform multi-label classification.

These contributions mentioned above showed positive answers to both questions i and ii, that were presented in the beginning of this section. In regards to questions iii and iv, it is not possible to determine a overall best method for identifying game events, and deal with multi-labeled frames, since different video games present very distinct characteristics, both in terms of graphics, as well as gameplay. However, this work proposed solid alternatives based on the state of the art in real-world video classification that are believed to be a good baseline and starting point to any future research that requires action or event classification in games.

6.1 Main Contributions

a) A new approach to generate labeled data from gameplay automatically based on the *Mario AI Framework* (KARAKOVSKIY; TOGELIUS, 2012);

b) A balanced Super Mario Bros dataset specifically designed to contain multi-label and single-label examples;

c) A proposed representation for clustering frames that boosts classification performance in gameplay footage;

d) An evaluation among four state-of-the-art convolutional neural networks (CNN) (ResNet50V2, MobileNetV2, VGG16 and AlexNet) for feature-extraction in *Super Mario* video frames;

e) An investigation among three state-of-the-art neural networks (Long Short Term Memory Network, Gated Recurrent Unit and Deep Multi-Instance Multi-Label Network) for classifying events from *Super Mario Frames*;

f) A fine-tuning strategy to significantly boost the performance of CNNs pre-trained on real-world data when applied to video games;

g) A strategy to fit games video footage into the Multi-Instance Multi-Label framework (ZHOU et al., 2012), in order to deal with the multi-label problem.

6.2 Future Works

In the context of event classification from gameplay footage, the future works that extend the contributions presented in this thesis are as follows:

a) Investigate how the proposed approaches perform in 3D games with various graphic styles;

b) Include the process classifying frames containing no events or actions;

c) Generate a diverse dataset focused on feature-extraction for all kinds of video games, in a was as to allow for a more specific pre-training process for DL models;

d) Investigate automatic labeling techniques for video footage, in order to simplify the data generation process;

e) Apply this classification framework in a task of analyzing the psychological profile of different players. The hypothesis is that gameplay analysis can be an effective tool to gather important information regarding behavior and cognitive abilities.

6.3 Published Papers and Submissions

- ❏ Matheus Prado Prandini Faria, Etienne Silva Julia, Marcelo Zanchetta do Nascimento, and Rita Maria Silva Julia. 2022. Investigating the Performance of Various Deep Neural Networks-based Approaches Designed to Identify Game Events in Gameplay Footage. ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (May 2022), 17 pages. https://doi.org/10.1145/3522624

- ❏ Etienne Silva Julia, Marcelo Zanchetta do Nascimento, Rita Maria Silva Julia, Matheus Prado Prandini Faria, and Rodrigo Zamboni. Deep Learning-based Models for Performing Multi-Instance Multi-Label Event Classification in Gameplay Footage. Submitted to the 34th IEEE International Conference on Tools with Artificial Intelligence (ICTAI) on the 07/08/2022 and awaiting review.

Bibliography

AGGARWAL, J.; CAI, Q. Human motion analysis: A review. **Computer Vision and Image Understanding**, v. 73, n. 3, p. 428–440, 1999. ISSN 1077-3142. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1077314298907445>.

Aloysius, N.; Geetha, M. A review on deep convolutional neural networks. In: **2017 International Conference on Communication and Signal Processing (ICCSP)**. [S.l.: s.n.], 2017. p. 0588–0592.

ANNETTA, L. A. Video games in education: Why they should be used and how they are being used. **Theory Into Practice**, Routledge, 2008.

APPERLEY, T. Genre and game studies: Toward a critical approach to video game genres. **Simulation Gaming - Simulat Gaming**, v. 37, p. 6–23, 03 2006.

ATEFEH, F.; KHREICH, W. A survey of techniques for event detection in twitter. **Computational Intelligence**, v. 31, n. 1, p. 132–164, 2015.

BEAUCHEMIN, S. S.; BARRON, J. L. The computation of optical flow. **ACM Comput. Surv.**, Association for Computing Machinery, New York, NY, USA, v. 27, n. 3, p. 433–466, set. 1995. ISSN 0360-0300. Disponível em: <https://doi.org/10.1145/212094.212141>.

BELLEMARE, M. G. et al. The arcade learning environment: An evaluation platform for general agents. **CoRR**, abs/1207.4708, 2012. Disponível em: <http://arxiv.org/abs/1207.4708>.

BISWAS, M. et al. State-of-the-art review on deep learning in medical imaging. **Frontiers in bioscience (Landmark edition)**, v. 24, p. 392—426, January 2019. ISSN 1093-4715. Disponível em: <https://doi.org/10.2741/4725>.

BOYLE, E.; CONNOLLY, T. M.; HAINEY, T. The role of psychology in understanding the impact of computer games. **Entertainment Computing**, v. 2, n. 2, p. 69–74, 2011. ISSN 1875-9521. Serious Games Development and Applications. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1875952110000200>.

BRE, F.; GIMENEZ, J.; FACHINOTTI, V. Prediction of wind pressure coefficients on building surfaces using artificial neural networks. **Energy and Buildings**, v. 158, 11 2017.

BROCKMAN, G. et al. **OpenAI Gym**. 2016.

CHO, K. et al. **Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation**. 2014.

DILLON, T. S.; NIEBUR, D. **Neural networks applications in power systems**. 1996.

EL-SAWY, A.; EL-BAKRY, H.; LOEY, M. Cnn for handwritten arabic digits recognition based on lenet-5. In: HASSANIEN, A. E. et al. (Ed.). **Proceedings of the International Conference on Advanced Intelligent Systems and Informatics 2016**. Cham: Springer International Publishing, 2017. p. 566–575. ISBN 978-3-319-48308-5.

Entertainment Software Association. **2020 Essential Facts About the Video Game Industry**. 2020. <https://www.theesa.com/esa-research/2020-essential-facts-about-the-video-game-industry>.

FARIA, M. P. P. et al. Investigating the performance of various deep neural networks-based approaches designed to identify game events in gameplay footage. **Proc. ACM Comput. Graph. Interact. Tech.**, Association for Computing Machinery, New York, NY, USA, v. 5, n. 1, may 2022. Disponível em: <https://doi.org/10.1145/3522624>.

FENG, J.; ZHOU, Z.-H. Deep miml network. In: **AAAI**. [S.l.: s.n.], 2017.

GEEST, R. D. et al. Online action detection. **CoRR**, abs/1604.06506, 2016. Disponível em: <http://arxiv.org/abs/1604.06506>.

GEEST, R. D.; TUYTELAARS, T. Modeling temporal structure with lstm for online action detection. In: **2018 IEEE Winter Conference on Applications of Computer Vision (WACV)**. [S.l.: s.n.], 2018. p. 1549–1557.

Global Data. **Video games market set to become a 300bn-plus industry by 2025**. 2021. <https://www.globaldata.com/video-games-market-set-to-become-a-300bn-plus-industry-by-2025>.

HE, K. et al. Deep residual learning for image recognition. In: **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2016.

HOCHREITER, S. The vanishing gradient problem during learning recurrent neural nets and problem solutions. **International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems**, v. 6, p. 107–116, 04 1998.

HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. **Neural computation**, v. 9, p. 1735–80, 12 1997.

HOPFIELD, J. J. Neural networks and physical systems with emergent collective computational abilities. **Proceedings of the National Academy of Sciences**, National Academy of Sciences, v. 79, n. 8, p. 2554–2558, 1982. ISSN 0027-8424. Disponível em: <https://www.pnas.org/content/79/8/2554>.

JANAI, J. et al. Computer vision for autonomous vehicles: Problems, datasets and state of the art. **Foundations and Trends® in Computer Graphics and Vision**, v. 12, n. 1–3, p. 1–308, 2020. ISSN 1572-2740. Disponível em: <http://dx.doi.org/10.1561/0600000079>.

Janarthanan, V. Serious video games: Games for education and health. In: **2012 Ninth International Conference on Information Technology - New Generations**. [S.l.: s.n.], 2012.

JORDAN, M. I. Serial order: a parallel distributed processing approach. technical report, june 1985-march 1986. 1986.

KARAKOVSKIY, S.; TOGELIUS, J. The mario ai benchmark and competitions. **IEEE Transactions on Computational Intelligence and AI in Games**, v. 4, n. 1, p. 55–67, 2012.

KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization. In: **3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings**. [S.l.: s.n.], 2015.

KOZLOV, M. D.; JOHANSEN, M. K. Real behavior in virtual environments: Psychology experiments in a simple virtual-reality paradigm using video games. **Cyberpsychology, Behavior, and Social Networking**, 2010.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. In: PEREIRA, F. et al. (Ed.). **Advances in Neural Information Processing Systems 25**. Curran Associates, Inc., 2012. p. 1097–1105. Disponível em: <http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf>.

LECUN BENGIO, H. Deep learning. **Nature**, p. 436–444, 05 2015.

Lecun, Y. et al. Gradient-based learning applied to document recognition. **Proceedings of the IEEE**, v. 86, n. 11, p. 2278–2324, 1998.

LEE, K.; KIM, H.; SUH, C. Crash to not crash: Playing video games to predict vehicle collisions. In: **ICML 2017**. [S.l.: s.n.], 2017.

LIPTON, Z. C. A critical review of recurrent neural networks for sequence learning. **CoRR**, abs/1506.00019, 2015. Disponível em: <http://arxiv.org/abs/1506.00019>.

LUO, Z. et al. Player experience extraction from gameplay video. **CoRR**, abs/1809.06201, 2018.

LUO, Z.; GUZDIAL, M.; RIEDL, M. Making cnns for video parsing accessible. **CoRR**, abs/1906.11877, 2019.

MARCUS, G. Deep learning: A critical appraisal. 01 2018.

MURTHY, C. B. et al. Investigations of object detection in images/videos using various deep learning techniques and embedded platforms—a comprehensive review. **Applied Sciences**, v. 10, n. 9, 2020. ISSN 2076-3417.

Ravanbakhsh, M. et al. Abnormal event detection in videos using generative adversarial nets. In: **2017 IEEE International Conference on Image Processing (ICIP)**. [S.l.: s.n.], 2017. p. 1577–1581.

RAWAT, W.; WANG, Z. Deep convolutional neural networks for image classification: A comprehensive review. **Neural Computation**, v. 29, n. 9, p. 2352–2449, 2017.

ROETTL, J.; TERLUTTER, R. The same video game in 2d, 3d or virtual reality – how does technology impact game evaluation and brand placements? **PLOS ONE**, Public Library of Science, v. 13, n. 7, p. 1–24, 07 2018. Disponível em: <https://doi.org/10.1371/journal.pone.0200724>.

SAMEK, W.; WIEGAND, T.; MüLLER, K.-R. Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models. **ITU Journal: ICT Discoveries - Special Issue 1 - The Impact of Artificial Intelligence (AI) on Communication Networks and Services**, v. 1, p. 1–10, 10 2017.

SANDLER, M. et al. Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation. **CoRR**, abs/1801.04381, 2018.

SHAO, K. et al. A survey of deep reinforcement learning in video games. **CoRR**, abs/1912.10944, 2019. Disponível em: <http://arxiv.org/abs/1912.10944>.

SILVER, D. et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. **Science**, American Association for the Advancement of Science, v. 362, n. 6419, p. 1140–1144, 2018. ISSN 0036-8075. Disponível em: <https://science.sciencemag.org/content/362/6419/1140>.

SIMONYAN, K.; ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. **arXiv 1409.1556**, 09 2014.

SOOMRO, K.; ZAMIR, A.; SHAH, M. Ucf101: A dataset of 101 human actions classes from videos in the wild. **CoRR**, 12 2012.

SYNNAEVE, G. et al. Torchcraft: a library for machine learning research on real-time strategy games. 11 2016.

SZELISKI, R. **Computer Vision: Algorithms and Applications**. 1st. ed. Berlin, Heidelberg: Springer-Verlag, 2010. ISBN 1848829345.

______. **Computer Vision: Algorithms and Applications**. 1st. ed. Berlin, Heidelberg: Springer-Verlag, 2010. ISBN 1848829345.

TOGELIUS, J. Ai researchers, video games are your friends! In: . [S.l.: s.n.], 2017. p. 3–18. ISBN 978-3-319-48504-1.

UDO, G. J. Neural networks applications in manufacturing processes. **Computers Industrial Engineering**, v. 23, n. 1, p. 97 – 100, 1992. ISSN 0360-8352. Disponível em: <http://www.sciencedirect.com/science/article/pii/036083529290072R>.

VINYALS, O. et al. Starcraft II: A new challenge for reinforcement learning. **CoRR**, abs/1708.04782, 2017. Disponível em: <http://arxiv.org/abs/1708.04782>.

______. Starcraft ii: A new challenge for reinforcement learning. 08 2017.

WANG, H.; SCHMID, C. Action recognition with improved trajectories. In: **2013 IEEE International Conference on Computer Vision**. [S.l.: s.n.], 2013. p. 3551–3558.

WIDROW, B.; RUMELHART, D.; LEHR, M. A. Neural networks: applications in industry, business and science. **Commun. ACM**, v. 37, p. 93–105, 1994.

WONG, J. H. M.; GALES, M. J. F. Sequence student-teacher training of deep neural networks. In: **INTERSPEECH**. [S.l.: s.n.], 2016.

XU, M. et al. Temporal recurrent networks for online action detection. **CoRR**, abs/1811.07391, 2018. Disponível em: <http://arxiv.org/abs/1811.07391>.

YU, M. et al. Spatiotemporal event detection: a review. **International Journal of Digital Earth**, Taylor Francis, v. 13, n. 12, p. 1339–1365, 2020.

ZEILER, M. D. Adadelta: an adaptive learning rate method. **arXiv preprint arXiv:1212.5701**, 2012.

Zhao, Z. et al. Object detection with deep learning: A review. **IEEE Transactions on Neural Networks and Learning Systems**, v. 30, n. 11, p. 3212–3232, 2019.

ZHOU, Z.-H. et al. Multi-instance multi-label learning. **Artificial Intelligence**, Elsevier, v. 176, n. 1, p. 2291–2320, 2012.

ZYDA, M. From visual simulation to virtual reality to games. **Computer**, v. 38, n. 9, p. 25–32, 2005.