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

MATHEUS PRADO PRANDINI FARIA, ETIENNE SILVA JULIA, MARCELO ZANCHETTA DO NASCIMENTO, and RITA MARIA SILVA JULIA, Federal University of Uberlândia, Brazil

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 proposes and analyses new deep learning-based models to identify game events occurring in Super Mario Bros gameplay footage. The architecture of each model is 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 models differ from each other according to the following elements: the type of the CNN; the type of the NN classifier; and the type of the game scene representation at the CNN input, being either single frames, or chunks, which are n-sequential frames (in this paper 6 frames were used per chunk) grouped into a single input. The main contribution of this article is to demonstrate the greater performance reached by the models which combines the chunk representation for the game scenes with the resources of the classifier recurrent neural networks (RNN).

CCS Concepts: • **Artificial Intelligence** → **Deep Learning**; • **Computer Vision** → *Convolutional Neural Networks.*

Additional Key Words and Phrases: gameplay footage, game events, feature extraction, classification, Super Mario Bros, CNN, RNN, Fully Connected Layers, frames, chunks

ACM Reference Format:
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. *Proc. ACM Comput. Graph. Interact. Tech.* 5, 1, Article 5775925 (May 2022), 17 pages. https://doi.org/10.1145/3522624

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 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

Authors' address: Matheus Prado Prandini Faria, matheusprandini.96@gmail.com; Etienne Silva Julia, etienne16sj@gmail.com; Marcelo Zanchetta do Nascimento, marcelo.zanchetta@gmail.com; Rita Maria Silva Julia, ritasilvajulia@gmail.com, Federal University of Uberlândia, Uberlândia, Minas Gerais, Brazil.

2577-6193/2022/5-ART5775925 $15.00
https://doi.org/10.1145/3522624

created to master the games of chess, shogi and go [Silver et al. 2018]. However, the perception that games can also be a useful case study in areas such as education, health and psychology [Annetta 2008; Boyle et al. 2011; Janarthanan 2012; Kozlov and Johansen 2010] has increasingly stimulated the development of works in artificial intelligence (AI) whose objectives are focused on such domains.

In order to be able to operate in parallel with the software-development platform in which a game was implemented (called *game engines*), a player agent must be endowed with the ability of retrieving information that are essential to make it aware of the main game environment characteristics.

Generally, such information is retrieved from game system flags presented in the game engines, 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 et al. 2019]; secondly, even considering game platforms which make available the game logs, it is not possible to count on game information in situations in which it must be retrieved from gameplay footage.

One of the most relevant information to be retrieved from the environment in which agents in general - including player agents - operate refers to the ongoing events that describe what is happening in such environment. Indeed, in areas like personal assistance robots, camera monitoring networks and autonomous cars, "instantly" being able to react to events happening in the environment is crucial. This challenge has motivated recent AI research on developing online event detection systems for real world videos, be it sports, traffic, or every day activities [Gao et al. 2017; Geest et al. 2016a; Xu et al. 2018]. On the other hand, there is a lack of work aimed at retrieving events from video games in real-time.

Even though the definition of events in AI can be a broad subject [Ravanbakhsh et al. 2017; Yu et al. 2020], in the context of this paper, game events are seen as *flags* that depict the most important things happening in the environment, being recorded in the game logs.

For example, in Super Mario, the occurrence of the events *EventJump* and *EventLand* indicate that *Mario has just took off* and *Mario has just landed*, respectively.

Motivated by such facts, this paper proposes various deep learning (DL) based models to identify game events occurring in gameplay footages depicting Super Mario runs (Mario AI Framework [Karakovskiy and Togelius 2012]). The architecture of each model is composed of the following modules: a CNN automatic feature extractor (to produce a concise and adequate feature-based representation for the game scenes) and a NN game event classifier (to identify the game event occurring in the feature-based representation produced by the CNN extractor).

The models differ from each other according to the following elements: the type of CNN extractor (MobileNetV2, ResNet50V2, VGG16 or AlexNet); the type of the classifier NN (long short-term memory (LSTM), gated recurrent unit (GRU) or fully-connected layers (FCL)); and the type of the image representing the game scenes at the CNN extractor input (individual frames or by chunk, that is, a set of consecutive frames).

The main contribution of this paper is to demonstrate the greater performance reached by models that combine chunk representations with a classifier RNN that allows relevant temporal and spatial information to persist throughout the processing of successive instances.

2 THEORETICAL FOUNDATION

This section presents an overview of the theoretical topics that support this work.

2.1 Convolutional Neural Networks

A CNN is composed of 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 networks input [Aloysius and Geetha 2017]. This model is made of sequences of pairs composed of *pooling* and *convolution* layers. 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 consists of a function that shrinks the dimensions of the image. The combination of multiple of these layers generates a smaller representation of the input that contains only the features that better represent it. The final step of a CNN is the classification, which is generally done by a set of *fully connected layers* that operate similarly to the traditional multi-layer perceptron, whose weight adjustment is executed by backpropagation.

2.2 Recurrent Neural Networks

RNNs are models that incorporate temporal notions to the learning process. This is done by adding specific connections between neurons, in order to allow for the persistence of relevant information in the network across the processing of different instances. The earlier RNN models [Hopfield 1982; Jordan 1997] were great at dealing with problems where the span of a single event was short, but still struggled with gradient loss problems in longer examples [Hochreiter 1998]. As a solution for this problem, in 1997 *Hochreiter* and *Schmidhuber* proposed the LSTM model [Hochreiter and Schmidhuber 1997], where each intermediary neuron was replaced by a memory cell. Each of these cells had a connection that allowed for the gradient to persist across multiple iterations without loss. Even though this model was presented over 20 years ago, it is still one of the most prominent RNN used in the literature until today. Another, more recent, proposal that also handles the gradient loss problem is the 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 a lower memory consumption and faster training.

2.3 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*, that 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 4.3. 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 and Terlutter 2018], as shown in figure 1. 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 framerate 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.

Fig. 1. In-game footage of the games Super Mario Galaxy on the left (3D) and Super Mario Bros on the right (2D)

3 RELATED WORK

This section summarizes works that stand out in the domain of retrieving important information from videos. Such presentation is divided into two parts: *Information Retrieval in Videos in General* and *Information Retrieval in Video Games*.

3.1 Information Retrieval in Real-World Videos

Among the works conceived to retrieve information from videos, those related to action recognition stand out (due to their practical applicability). Such studies focus on two distinct approaches: *Offline Action Detection*, or *Online Action Detection*. The main difference between them is that offline methods observe the entire video before outputting the action labels, whereas online methods output their predictions as soon as the action happens. Since this paper uses the online approach, this section will focus on online detection strategies-based works.

In [Geest et al. 2016a] the authors proposed a concrete definition for the temporal modeling problem in action detection with LSTMs, and also presented a two stream online-based detection model, where one stream manages the frame features, and the other the temporal dependencies between instances [Geest et al. 2016b]. In the context of RNNs, [Gao et al. 2017] presents an interesting learning model that adapts the training of RNNs to predict actions as early as possible, by adding a module that continually processes previous visual representations, and outputs a sequence of anticipated future predictions. In this same line, the work [Xu et al. 2018] encapsulates a lot of relevant functionalities for online action detection into a single framework, the temporal recurrent network (TRN). The TRN introduces a future event prediction module, through a *temporal decoder* and a *future gate*, to a traditional RNN prediction strategy, in order to enhance prediction results. Still in the context of temporal dependencies in neural networks, the authors of [Karpathy et al. 2014] did an empirical analysis of multiple CNN architectures to identify which temporal connectivity patterns were better at taking advantage of spatio-temporal information in videos and how this information improved CNN performance overall.

3.2 Information Retrieval in Video Games

In [Lee et al. 2017], the authors use the popular game *Grand Theft Auto* as a platform for training DL methods to perform vehicular collision prediction. The work [Summerville et al. 2017] uses the Super Mario Bros game to analyze to which extent data quantity and quality impact on distinct machine learning - based methods which perform content generation. Super Mario Bros is also used in [Guzdial et al. 2017], which proposes an approach to automatically learn level design features and game rules from gameplay footage. The work [Luo et al. 2018] is the closest to the purposes of this paper. It 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 - which comes closest to the objectives of this work - consists on training an AlexNet [Krizhevsky et al. 2012] model from scratch with a manually labeled Gwario game dataset. Their second approach utilizes a *student-teacher* technique [Wong and 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 et al. 2012] related to the game Skyrim. In such approach, the game events come from the UCF 101 dataset.

4 IMPLEMENTATION OF THE DL-BASED MODELS TO RECOGNIZE EVENT GAMES IN SUPER MARIO

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 2.

The models differ from each other by the following factors: types of the NNs 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 for 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.

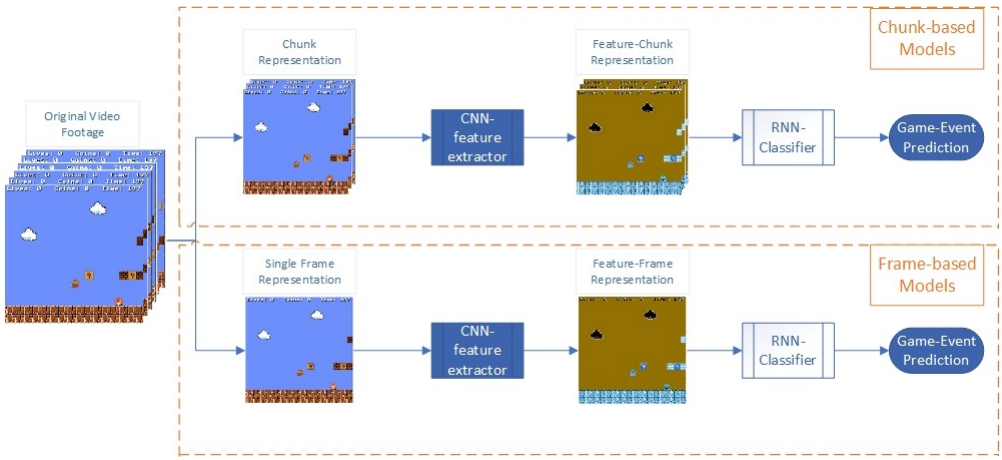

Fig. 2. General architecture of the chunk-based and frame-based models

4.1 Capturing Frames and Chunks from the Gameplay Footage

The required training data samples for this paper (frames and chunks) are extracted from 135 Super Mario gameplay videos with the Mario AI Framework [Karakovskiy and Togelius 2012] (as shown in subsections 4.1.1 and 4.1.2). These data are then pre-processed (subsection 4.2) in order to generate the datasets necessary to the execution of the experiments.

4.1.1 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).

4.1.2 Chunk capture. Inspired by [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.

4.2 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 on 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]).

4.3 Game Events

The Super Mario game events investigated in this work are defined by the Mario AI Framework itself as being fundamental to model 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).

4.4 The Models

In order to pursue its objectives, this work firstly implements 25 distinct models to perform game event detection in Super Mario game footages.

Among these models, 24 correspond to novel approaches proposed in this paper 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).

Table 1. Summary of the models proposed here

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

- *New proposed models* (corresponding to the first 24 models of Table 1): in these unpublished models, the feature extraction and the classification are performed by distinct NNs. More specifically, the feature extraction is carried out by a pre-trained CNN and the classification is executed by a RNN (LSTM or GRU) or a FCL, which must be trained from scratch. These models will be referred to here as *FE+C+IR*, 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). 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 inspiring by [Xu et al. 2018] and by empirical tests.
- *State of Art-based Model* (corresponding to the last model of Table 1): 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. Such 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 trained from scratch AlexNet copes with both feature extraction and classification).

4.5 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) involving the set of 12 game events presented at subsection 4.3. In order to optimize the NNs' training, the frames and chunks that make up such datasets undergo the pre-processing described in subsection 4.2.

4.5.1 Frame Dataset: 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*.

4.5.2 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 4.1.2). This dataset is used to generate the *Chunk-Feature Dataset* to be used to train the classifier C of all models *FE+C+chunk*.

4.5.3 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 (subsection 4.5.1) and the Chunk Dataset (subsecion 4.1.2) 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 2.3). In fact, the simplicity of the images of Super Mario allows for obtaining very satisfactory runtimes 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).

4.6 Feature Extractors

As aforementioned, this work investigates the use of five distinct CNN Feature Extractor: 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 feature 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 number of layers and dimension of the weight vector. This allows for a 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 features extractor in modern researches. Its primary characteristic is the use of very small convolutional kernels in a deep structure with lot of layers [Simonyan and 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 and Geetha 2017], having improved on 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 4.5.

On the other hand, the CNN EC-AlexNet used in the last model of Table 2 is trained from scratch from Frame Dataset (section 4.5). As both the feature extraction and the classification modules of such CNN are trained together, they will be more detailed in section 4.7. 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).

4.7 Classifiers

This paper investigates the performance of RNNs, FCL and AlexNet in the classification process.

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

The RNNs used herein are LSTM [Hochreiter and 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 et al. 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 4.5).

On the other hand, in the unpublished models *FE+C+IR* proposed here (subsection 4.4), 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 4.5).

The optimizer and loss parameters used in all training sessions performed in this paper were Adam [Kingma and 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.

5 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 and 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 paper is available in Github[3].

5.1 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 2.3); 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 a RNN to interrelate information received over time makes it very suitable to identify the main game event occurring in the multiple 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 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 runtime, Table 3 corroborates the following theoretical expectation (subsection 4.7): 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 (section 2.2), 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 (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 set of features larger than those produced by the other ones (as seen in subsection 4.6), 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).

[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)
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
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
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
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

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 paper, 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*.

5.2 Test Scenario 2

As provided in the beginning of section 5, 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* presented in subsection 4.5.1. 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 paper:

- **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)
5	25	84.2%	1.0%	0.5405	0.540	390s
	26	80.4%	0.2%	1.0454	0.108	169s

5.3 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	93.1%	0.5%	6s	240s	2s
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 paper (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 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 paper is more suitable considering an online setting.

Still 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. It 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 paper, 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.

6 CONCLUSION AND FUTURE WORKS

This 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; a FCL or a 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.

As future works, the authors intend: to extend the present proposal in such a way as to include chunks devoid from events or containing more than one event; to investigate the performance of the approaches studied here for 3D games; and, finally, to use such approaches in the construction of player agents apt to improve the cognitive ability of people with syndromes such as Down or autism. More specifically, in this last extension associated to human health, the idea is to use the approaches studied here to retrieve events occurring in recorded scenes of games played by people with a given syndrome. From these game events, the agent will be able to abstract the actions executed by these people in various game situations and to map the specific situations in which their decision-making was fragile. From this point on, the agent's engine must try to direct the game in order to provoke the occurrence of such situations and present some clues to help the player to better deal with them.

ACKNOWLEDGMENTS

The authors thank CAPES for financial support.

REFERENCES

N. Aloysius and M. Geetha. 2017. A review on deep convolutional neural networks. In *2017 International Conference on Communication and Signal Processing (ICCSP)*. 0588–0592. https://doi.org/10.1109/ICCSP.2017.8286426

Leonard A. Annetta. 2008. Video Games in Education: Why They Should Be Used and How They Are Being Used. *Theory Into Practice* (2008). https://doi.org/10.1080/00405840802153940

Elizabeth Boyle, Thomas M. Connolly, and Thomas Hainey. 2011. The role of psychology in understanding the impact of computer games. *Entertainment Computing* 2, 2 (2011), 69–74. https://doi.org/10.1016/j.entcom.2010.12.002 Serious Games Development and Applications.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. arXiv:1406.1078 [cs.CL]

Jiyang Gao, Zhenheng Yang, and Ram Nevatia. 2017. RED: Reinforced Encoder-Decoder Networks for Action Anticipation. arXiv:1707.04818 http://arxiv.org/abs/1707.04818

Roeland De Geest, Efstratios Gavves, Amir Ghodrati, Zhenyang Li, Cees Snoek, and Tinne Tuytelaars. 2016a. Online Action Detection. arXiv:1604.06506 [cs.CV]

Roeland De Geest, Efstratios Gavves, Amir Ghodrati, Zhenyang Li, Cees Snoek, and Tinne Tuytelaars. 2016b. Online Action Detection. *CoRR* abs/1604.06506. arXiv:1604.06506 http://arxiv.org/abs/1604.06506

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

Matthew Guzdial, Boyang Li, and Mark O. Riedl. 2017. Game Engine Learning from Video. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence* (Melbourne, Australia) *(IJCAI'17)*. AAAI Press, 3707–3713.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Sepp Hochreiter. 1998. The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 6 (04 1998), 107–116. https://doi.org/10.1142/S0218488598000094

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation* 9, 8 (1997), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735 arXiv:https://doi.org/10.1162/neco.1997.9.8.1735

J J Hopfield. 1982. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences* 79 (1982), 2554–2558. https://doi.org/10.1073/pnas.79.8.2554

V. Janarthanan. 2012. Serious Video Games: Games for Education and Health. In *2012 Ninth International Conference on Information Technology - New Generations*. https://doi.org/10.1109/ITNG.2012.79

Michael I. Jordan. 1997. Chapter 25 - Serial Order: A Parallel Distributed Processing Approach. 121 (1997), 471–495. https://doi.org/10.1016/S0166-4115(97)80111-2

S. Karakovskiy and J. Togelius. 2012. The Mario AI Benchmark and Competitions. *IEEE Transactions on Computational Intelligence and AI in Games* 4, 1 (2012), 55–67. https://doi.org/10.1109/TCIAIG.2012.2188528

Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. 2014. Large-Scale Video Classification with Convolutional Neural Networks. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*. 1725–1732. https://doi.org/10.1109/CVPR.2014.223

Diederik P. Kingma and Jimmy Ba. 2015. 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*.

Michail D. Kozlov and Mark K. Johansen. 2010. Real Behavior in Virtual Environments: Psychology Experiments in a Simple Virtual-Reality Paradigm Using Video Games. *Cyberpsychology, Behavior, and Social Networking* (2010). https://doi.org/10.1089/cyber.2009.0310

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger (Eds.). Curran Associates, Inc., 1097–1105.

Kangwook Lee, Hoon Kim, and Changho Suh. 2017. Crash To Not Crash: Playing Video Games To Predict Vehicle Collisions. In *ICML 2017*.

Zijin Luo, Matthew Guzdial, Nicholas Liao, and Mark Riedl. 2018. Player Experience Extraction from Gameplay Video. *CoRR* abs/1809.06201 (2018). arXiv:1809.06201

Zijin Luo, Matthew Guzdial, and Mark Riedl. 2019. Making CNNs for Video Parsing Accessible. *CoRR* abs/1906.11877 (2019). arXiv:1906.11877

M. Ravanbakhsh, M. Nabi, E. Sangineto, L. Marcenaro, C. Regazzoni, and N. Sebe. 2017. Abnormal event detection in videos using generative adversarial nets. In *2017 IEEE International Conference on Image Processing (ICIP)*. 1577–1581. https://doi.org/10.1109/ICIP.2017.8296547

Johanna Roettl and Ralf Terlutter. 2018. The same video game in 2D, 3D or virtual reality – How does technology impact game evaluation and brand placements? *PLOS ONE* 13, 7 (07 2018), 1–24. https://doi.org/10.1371/journal.pone.0200724

Mark Sandler, Andrew G. Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. 2018. Inverted Residuals and Linear Bottlenecks: Mobile Networks for Classification, Detection and Segmentation. *CoRR* abs/1801.04381 (2018). arXiv:1801.04381

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. 2018. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science* 362, 6419 (2018), 1140–1144. https://doi.org/10.1126/science.aar6404

Karen Simonyan and Andrew Zisserman. 2014. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv 1409.1556* (09 2014).

Khurram Soomro, Amir Zamir, and Mubarak Shah. 2012. UCF101: A Dataset of 101 Human Actions Classes From Videos in The Wild. *CoRR* (12 2012).

Adam Summerville, Sam Snodgrass, Matthew Guzdial, Christoffer Holmgård, Amy K. Hoover, Aaron Isaksen, Andy Nealen, and Julian Togelius. 2017. Procedural Content Generation via Machine Learning (PCGML). *CoRR* abs/1702.00539 (2017). arXiv:1702.00539 http://arxiv.org/abs/1702.00539

Jeremy Heng Meng Wong and Mark John Francis Gales. 2016. Sequence Student-Teacher Training of Deep Neural Networks. In *INTERSPEECH*. https://doi.org/10.21437/Interspeech.2016-911

Mingze Xu, Mingfei Gao, Yi-Ting Chen, Larry S. Davis, and David J. Crandall. 2018. Temporal Recurrent Networks for Online Action Detection. arXiv:1811.07391 http://arxiv.org/abs/1811.07391

Manzhu Yu, Myra Bambacus, Guido Cervone, Keith Clarke, Daniel Duffy, Qunying Huang, Jing Li, Wenwen Li, Zhenlong Li, Qian Liu, Bernd Resch, Jingchao Yang, and Chaowei Yang. 2020. Spatiotemporal event detection: a review. *International Journal of Digital Earth* 13, 12 (2020), 1339–1365. https://doi.org/10.1080/17538947.2020.1738569 arXiv:https://doi.org/10.1080/17538947.2020.1738569