Deep Learning - based Models for Performing Multi-Instance Multi-Label Event Classification in Gameplay Footage

***Abstract*—This paper presents two Deep Learning - based models conceived to deal with Multiple-Instances Multi-Labels (MIML) event classification in gameplay footage. The architecture of these models is based on a data generator script, a feature extractor Convolutional Neural Network (CNN) and a deep classifier neural network. More specifically, as both models use the fine-tuned MobileNetV2 as a feature extractor and are trained from the same datasets, they differ from each other essentially by the classifier network, since one of the models uses the *Long Short-term Memory* (LSTM), while the other uses the DeepMIML for classification purposes. The Super Mario bros game is used as a case study. The *Chunks* (or frame groupings) are used as a data representation structure. The main contribution of this work with respect to the state of the art 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 Deep Learning models to perform MIML event classification in gameplay footage.**

***Index Terms*—MIML event classification, gameplay footage, frame-based and chunk-based data representations, deep extractor Neural Networks, deep classifier Neural Networks.**

I. Introduction

In machine learning (ML), the ability of autonomous agents to extract relevant information about the environment they operate in, and use it in order to improve their decision making is one of the main requisites for a good performance (REFERENCIA). When considering dynamic environments, such as the ones represented in videos, one of the key pieces of information are the events, since, in a broad manner, they represent the dynamic changes and interactions that happen in the environment. Then, in situations in which the scenario an agent operates in vary drastically, the occurring events can also be very variable.

In this context, a lot of recent research has been produced with the goal of using autonomous agents to identify events in relevant contexts of modern life (referencia). In areas like personal assistance robots, camera monitoring networks and autonomous vehicles, "instantly" being able to react to events happening in the environment is crucial. This is why artificial intelligence (AI) researches focused on online event detection systems can be very valuable [16] [5].

In addition to that, video games have been used as case studies for ML research for a long time, since their high degree of complexity and variability makes them great benchmarks for state of the art AI algorithms (VER ARTIGO HENRIQUE). Also, a lot of games can be used to emulate practical real world situations, but in a controlled, easily scalable and adjustable way [10]. At last, video games have also been used in the fields of medicine and psychology as, among other things, valuable tools to study behavior, improve motor skills (especially hand-to-eye coordination), and assist with the treatment of developmental disorders (Pegar referencias artigo passado).

Despite all of these application, there is still great lack of works that focus on the retrieval of relevant information from video games. This is a problem, considering that the ability to access relevant information from a study scenario is a fundamental requisite for any research.

Considering this, the present paper proposes a new approach to classify game events in gameplay footage. To test this approach, the game Super Mario Bros was selected, since it is a classic and popular game, which, in addition to being used in numerous researches involving Machine Learning, is pointed out in the field of psychology and medicine as one of the most suitable games to be used as an instrument capable of improving the cognitive abilities of people with psychological or neurological weaknesses [REFERENCIA DO MARIO E DO ARTIGO DE PSICOLOGIA].

It is relevant to point out that this paper is a continuation of the preliminary work [3], which investigated and compared the performance of various combinations of Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) in performing *multi-instance single-label* game event classification in footage of Super Mario Bros levels. Such work also investigated what would be the best type of representation for game scenes: a representation based on frames or chunks (grouping of frames). The winner among the investigated models combines the *standard CNN MobileNetV2* [13] as a feature extractor network, the *RNN LSTM* [7] as a classifier network, and the *chunks* as a data representation structure.

Then, the objectives pursued here go beyond those achieved in such a state-of-art work for aiming both to perform *multi-instance multi-label* game event classification in footage of Super Mario Bros levels and to improve the best model obtained in such work. For this, the main contributions of the present paper consist on extending the current state of the art related to performing event classification from game footage in the following aspects:

- The classification problem was extended from a multi-

instance single-label framework to a multi-instance multi-label one (MIML) [18], in order to include game scenarios where multiple events happen simultaneously;
- In order to produce adequate multi labeled instances to train the proposed models, the following activities were performed: implementation of an automatic data generator script to produce the frames from the game footage; creation of a pre-processed, balanced and multi-labeled dataset from the data produced by the script; from such dataset, generation of the chunks and construction of both datasets to be used to train the proposed models: a frame-based dataset and a chunk-based dataset;
- To boost the feature-extraction process, this work proposed a new fine tuned version of the standard MobileNetV2 ([13]) in which this latter is retrained from a specialized dataset composed of Super Mario Bros frames;
- In order to investigate an alternative classifier deep NN to deal with MIML problem in game play footage of Super Mario Bros, in addition to testing the RNN LSTM successfully used in [3] to deal with multi-instance single-label problem, the present work also tested the Deep MIML NN proposed in [4] to deal with MIML problem (in images and texts).

II. Theoretical Foundation

This section will present a general view of the main theoretical topics related to this work.

A. Multi-Instance Multi-Label Framework

The MIML framework, proposed in [18], is defined by each data example being comprised of multiple instances, associated with multiple-labels. Traditionally in Machine Learning, each instance of data is associated with a single label, or even, each instance with multiple labels. Figure 1 illustrates such distinct frameworks. However, for problems involving complicated examples with multiple semantic meanings, using more than one instance to represent them can allow for additional inherent patterns to the data to become more clear. So, in these situations, the MIML framework can be a more natural and appropriate way to represent the data.

An important idea in multi-instance learning are sub-concepts. When considering a broad concepts like "*Africa*", it can hardly be described by only one aspect. Usually a group of cultural and environmental identifiers are 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.

1) Multi-Label Evaluation Metrics: In traditional supervised learning, as mentioned in the previous subsection, a single instance is associated to a single label. This allows for an *accuracy* metric (the percentage of examples correctly classified) to often be a good enough indicator of performance [18]. 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. Then, for example, it is better for a model to identify 4 out of 5 labels correctly, than getting 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 that were are 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 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.

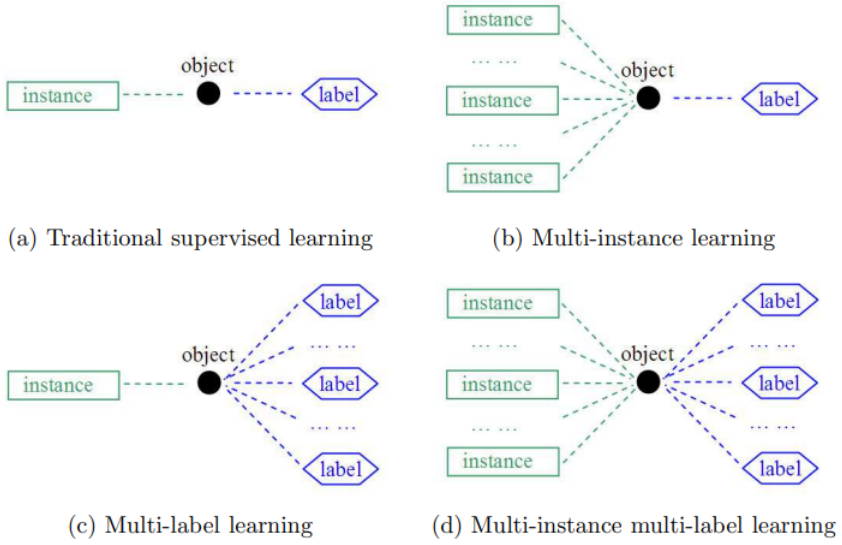
B. Deep Neural Networks

This section will provide a brief overview of the deep neural network architectures that were used in this work.

1) Convolutional Neural Networks: Convolutional neural networks are deep learning models specialized in image classification problems. They are composed of alternated convolutional, and pooling layers that perform an automatic feature selection process in order to transform the network's input into the best possible representation for classification performance [1]. Some of the more used CNNs are: MobileNet, ResNet and VGG-16.

2) Recurrent Neural Networks: RNNs are architectures that incorporate specific connections between neurons in order to allow for relevant information to persist in the network across multiple iterations. They are particularly effective in scenarios where the class of an instance has a correlation with the classes of previously processed examples in the network [2]. Problems like action detection in videos, multiple object detection and text analysis greatly benefit from this feature. Some of the more used RNNs are the Long-Short Therm Memory (LSTM) and the Gated Recurrent Unit.

3) Deep MIML Network: As previously mentioned in section II-A, a lot of real world applications greatly benefit from being represented in a multi-instance multi-label context. The Deep MIML Network [4] is a model that adds an automatic instance generator and a sub-concept learning structure to



(a) Traditional supervised learning (b) Multi-instance learning

(c) Multi-label learning (d) Multi-instance multi-label learning

Fig. 1. Four different learning frameworks, image from [18]

the traditional neural network formation. The structure of the Deep MIML Network (figure 2) consists of a feature-based-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.

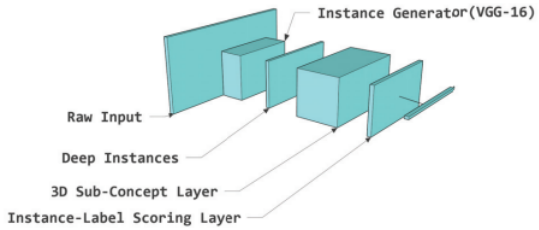


Fig. 2. Representation of a Deep MIML Network, image from [4]

C. Super Mario Bros

The game Super Mario Bros was first released in 1985 by the company Nintendo. In it, the player plays as a plumber called Mario and has to traverse a series of stages in order to reach and save a princess. Even though it is a very simple premise, each stage presents a different set of obstacles and enemies that require precise actions from the player in order to advance. These actions are: walking, running, jumping and throwing a fire ball. All of these trigger different *Game Events*, that represent the interactions the player has with the environment through his actions. All the game events related to Super Mario are listed in section IV-A.

From a visual standpoint, the game has very simple graphics in a ”pixelated” style. The game also presents a 2D (two dimensional) point of view, which lacks the object depth that is present in 3D games [12] as can be seen in figure 3.

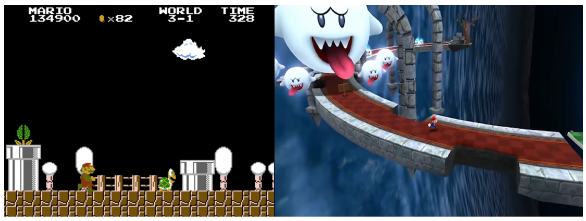


Fig. 3. Super Mario Bros on left (2D), Super Mario Galaxy on right (3D)

III. Related Works

This section will present a brief overview of some works in the literature that relate to the domains of: *information retrieval in Video Games*,and *Multi-Instance Multi-Label classification*.

A. Information Retrieval in Video Games

In the work [10], the authors developed a system to generate a vehicle collision dataset from the game *Grand Theft Auto V*, that was then used to train a CNN model to detect collisions in real-world settings.

In the context of information retrieval in the Super Mario game, [15] uses the game to investigate the impact that data quantity and data quality have on different ML models. [6] proposes a new approach to learn gameplay rules, as well as game level design characteristics from gameplay footage. The paper [11] proposed three DL based approaches for multi-label game event classification in game footage of the games: *Gwario* (a Super Mario clone), *Megamen*, and *The Elder Scrolls V: Skyrim*, their primary contribution was showing how

CNN networks pre-trained on real-world data can have great results when applied to the correct game scenario. Finally, in [3], which is the predecessor to this paper, the authors tested 26 models that combined state of the art DL networks for the task of classifying game events in Super Mario.

B. Multi-Instance Multi-Label Classification

In [18], the authors proposed the Multi-Instance Multi-Label framework, that is widely used to structure classification problems across many research fields. [4] took this framework and proposed a new *Deep MIML Network* as a general DL model that, not only is able to automatically generate multiple instances from a single piece of data thought a CNN, but also can keep track of instance-label relations. Also, in [14] the authors explored the usage of CNNs on multi-label image classification problems. They proposed a deep Multi-Modal CNN, that was designed for MIML learning, and whose main appeal was combining the benefits of the MIML framework, with the image classification capabilities of CNNs.

IV. Proposed Approach

This section details the models investigated in this work 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 [3], the authors investigated various models to perform single event detection in game-play footage of *Super Mario Bros*, and the winner model was made of a feature extractor MobileNetV2 combined with a classifier LSTM; 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; [8];
- 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 standard pre-trained version of MobileNetV2 [13] 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 Neural Networks were investigated: 1) LSTM [7]; 2) the classifier Sub-concept Layer of the DeepMIML NN [4].

Briefly, the multi-label game event classification process carried out in this paper can be resumed as following: firstly, the Mario AI Framework is used to capture game play 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 following: 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 standard 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 firstly 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 in the state-of-art work [3], in which such representation produced much better results than the frame-based one.

All the steps of these processes will be broken down and explained in detail in the following sub-sections.

A. Data Generation

In order to cope with the objectives of this work, the authors had to implement a script, based on the Mario AI Framework [8], 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 controlling 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.

The footage was captured at 30 frames per second, and the following 12 events (todos que pertencem ao framework) were selected to be 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 (Figure 4).

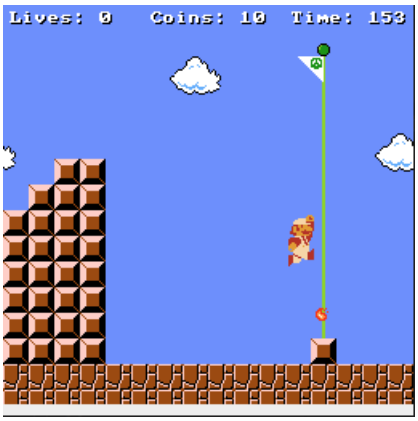


Fig. 4. Example of an *EventWin* frame.

B. 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 paper:

- **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 events to be used to label the training data;
- **Definition of a correct capture window**: Super Mario AI Framework annotates each event occurrence in the frame associated to the exact moment it starts. However, for some kind of events, such frame is 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**: the models proposed in this paper were implemented in the Keras API. 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.

C. Training Datasets

This section presents how both training datasets (*Frame Dataset* and *Chunk Dataset*) were created.

After being generated by the generator script (Sub-section IV-A) all labeled row frames with *at least one* event were pre-processed (Section IV-B and stored in a dataset. Next, as such 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 from, it had to be submitted to an adequate balancing strategy. As, due to intrinsic characteristics of 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.

At 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 on 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 .

1) Frame Dataset: In order to cope with the objective of improving the feature extraction process, here the standard MobileNetV2 pre-trained from ImageNet [13], had to be retrained, in such a way as to produce a fine tuned extractor CNN. To speed up such retraining, it was performed through the *Frame Dataset*, which is composed exclusively of the $9,782$ frames with single label present in the pre-processed and balanced dataset.

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 event);
3) Apply the same pre-processing described in IV-B to all the frames taken in the previous step that present 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 [3].

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 increasing in the number of multi-labeled instances available to train the models proposed in this work (compared to the few quantity of multi-labeled instances present in the pre-processed frame-based dataset (which is equal to 217).

D. Feature Extraction

Concerning the feature extraction process, as aforementioned, the main objective here is to produce a fine-tuned version of the standard MobileNetV2 pre-trained on the ImageNet dataset [13]. 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 number of layers and dimension of the weight vector, which allows for a 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 [3]), which performs single event classification in Super Mario bros game-play footage.

In this way, here this standard pre-trained MobileNetV2 was retrained from the *Frame Dataset* presented in Subsection IV-C. Such a strategy is based on the following fact: the ImageNet dataset, from which the standard 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 IV, still inspired in the best model obtained in [3], here the training of the feature extractor and the classifier deep NNs is performed separately, as following: 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 following: 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.

E. Classifiers

In order to find an adequate NN to perform MIML classification in game-play footage, this work investigate two distinct deep classifier NN: the Sub-Concept layer of the DeepMIML NN [4] and the LSTM [7].

The reasons for such choice are the following: concerning the Sub-Concept layer of the DeepMIML network, as resumed in Subsection II-B3, 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 [18].

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 game-play footage carried out in [3]. 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 [4], 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 RNN used in the literature until today.

V. Experiments and Results

The purpose of this section is to present the experiments carried out to built and to 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.

A. Experiments 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 feature extractor in the models proposed in this paper (Section IV-D), as well as to perform a performance comparison between such fine-tuned extractor and the standard 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 IV-C1, of the complete architecture (feature extractor layers + classifier layers) of the standard MobileNetV2 pre-trained on the ImageNet datset; 2) In order to preform a performance evaluation between the fine-tuned MobileNetV2 obtained in the *step 1* and the standard 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 standard pre-trained MobileNetV2 [13], that is: 1280 features and 2,257,984 parameters, as shown in Table I.

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 [9], the loss parameter was Categorical cross-entropy and the learning rate was equal to 0.0005, following what was used in [3].

TABLE I
Feature-Extractor Models Overview

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

TABLE II
Mean Accuracy and Loss Results

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

Table II 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 standard MobileNetV2 (accuracy 74.69% and Mean Loss $0,7021\%$). This makes sense, considering that ImageNet, from which the standard 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.

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

B. Experiments 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 II-B3.

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 IV-D.

Table III resumes the information related to both classifiers. Obviously, the 12 labels mentioned in such table refer to the 12 events described in Section IV-A.

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* [17].

As this works copes with MIML setting, in accordance with what is stated in Section II-A, 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 V-B.

TABLE III
CLASSIFIERS MODELS OVERVIEW

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

TABLE IV
CLASSIFIER RESULTS

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

The results show that LSTM 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 [4], where the authors investigated both LSTM and MIML-based models dealing with image object detection in the Common Objects in Context (COCO) dataset.

VI. CONCLUSIONS AND FUTURE WORKS

In order to extend the current state of art in the domain of performing MIML event detection in gameplay footage, this paper proposed two distinct Deep Learning 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 uses 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 has performed a little better than the *Fine-tunned-MobileNetV2 + DeepMIML-SubConcept-Layer* model.

Finally, as future works, the authors intend to use the models produced in this paper in the implementation of a player agent endowed with the ability of stimulating the cognitive and/or motor skilness of patients with some kind of syndrome (such as Down syndrom), in the following way: the events identified in the gameplay footage will represent the basis to perceive the actions performed by the game users and, through them, to map their possible cognitive weaknesses. From this information, the agent must adapt its decision-making engine in order to lead the game to situations that stimulate the cognitive development of its users.

REFERENCES

[1] Neena Aloysius and Geetha Ganesh. A review on deep convolutional neural networks. pages 0588–0592, 04 2017.

[2] Wei Fang, Yupeng Chen, and Qiongying Xue. Survey on research of rnn-based spatio-temporal sequence prediction algorithms. *Journal on Big Data*, 3(3):97, 2021.

[3] Matheus Prado Prandini Faria, Etienne Silva Julia, Marcelo Zanchetta do Nascimento, and Rita Maria Silva Julia. 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), may 2022.

[4] Ji Feng and Zhi-Hua Zhou. Deep miml network. In *AAAI*, 2017.

[5] Roeland De Geest, Efstratios Gavves, Amir Ghodrati, Zhenyang Li, Cees Snoek, and Tinne Tuytelaars. Online action detection. volume abs/1604.06506, 2016.

[6] Matthew Guzdial, Boyang Li, and Mark Riedl. Game engine learning from video. pages 3707–3713, 08 2017.

[7] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[8] Sergey Karakovskiy and Julian Togelius. The mario ai benchmark and competitions. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(1):55–67, 2012.

[9] Diederik P. Kingma and Jimmy Ba. 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*, 2015.

[10] Kangwook Lee, Hoon Kim, and Changho Suh. Crash to not crash: Playing video games to predict vehicle collisions. In *ICML 2017*, 2017.

[11] Zijin Luo, Matthew Guzdial, Nicholas Liao, and Mark Riedl. Player experience extraction from gameplay video. *CoRR*, abs/1809.06201, 2018.

[12] Johanna Roettl and Ralf Terlutter. The same video game in 2d, 3d or virtual reality – how does technology impact game evaluation and brand placements? *PLOS ONE*, 13:1–24, 07 2018.

[13] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[14] Lingyun Song, Jun Liu, Buyue Qian, Mingxuan Sun, Kuan Yang, Meng Sun, and Samar Abbas. A deep multi-modal cnn for multi-instance multi-label image classification. *IEEE Transactions on Image Processing*, 27(12):6025–6038, 2018.

[15] Adam Summerville, Sam Snodgrass, Matthew Guzdial, Christoffer Holmgård, Amy Hoover, Aaron Isaksen, Andy Nealen, and Julian Togelius. Procedural content generation via machine learning (pcgml). *IEEE Transactions on Games*, PP, 02 2017.

[16] Mingze Xu, Mingfei Gao, Yi-Ting Chen, Larry S. Davis, and David J. Crandall. Temporal recurrent networks for online action detection, 2018.

[17] Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

[18] Zhi-Hua Zhou, Min-Ling Zhang, Sheng-Jun Huang, and Yu-Feng Li. Multi-instance multi-label learning. *Artificial Intelligence*, 176(1):2291–2320, 2012.