Action Recognition in video with a Convolutional+LSTM Model

Luis Navarro Velasco

Master Thesis. Universidad Pontificia de Comillas

Master in Smart Industry

Tutors: Álvaro López López and Lucía Güitta López

Abstract

There are thousands of millions of video surveillance cameras around the world, designed to ensure that the streets or premises in which they are installed are safe. Currently, all of these cameras require constant monitoring by a person, who must ensure that there is no anomalous activity on any of the cameras, and therefore that no crime is occurring and no one is in danger. Developing a model capable of analyzing what is happening in each security camera and drawing conclusions from these videos would save millions of euros worldwide, as well as getting a machine to do the thankless job of constantly monitoring these cameras. This is why there is growing interest in achieving a stable model capable of solving this problem.

This project aims precisely at this application of Computer Vision: detecting actions in videos from video surveillance cameras, which turns the problem into a classification problem, in which the model must be able to indicate whether the action in question has occurred or not in a video.

I. INTRODUCTION

Currently, there is a growing interest in developing Artificial Intelligence models that facilitate the tasks to be performed by humans. The problems whose solution is sought through the application of this type of algorithms are becoming increasingly larger and more varied, ranging from the detection of people and objects in images to the prediction of the stock market or the driving of autonomous cars. Although it may seem that Artificial Intelligence has just landed in the world, this is not so. The first neural network model was designed by Frank Rosenblatt more than sixty years ago; more specifically in 1957 [1]. This model introduced for the first time the Perceptron, which was capable not only of learning, but also of generalizing, and is still used today for pattern recognition.

These neural network models have evolved greatly since Frank Rosenblatt's first design, giving rise to very different architectures that provide solutions to a wide variety of problems. This rapid evolution in recent times has been favored by the rapid increase in processing capacity, which is growing exponentially and is making it possible to develop models of great complexity.

One of the aspects that Artificial Intelligence has adopted is Deep Learning, which has meant a paradigm shift, since before the appearance of this type of models, it was necessary to extract the characteristics of the data before training the Artificial Intelligence model, which made the work more complex and biased the decision making towards the characteristics that the developers considered important. With the advent of Deep Learning, it is no longer necessary to extract these features, but thanks to the depth of the networks (i.e. the number of layers connected to each other), they are able to extract these features by themselves, understanding which ones are relevant when making predictions.

Deep Learning models are currently used for multiple solutions: predicting future values, classifying input data, understanding the content of a text, detecting objects in images.... There are many applications, but the one that will be discussed in depth in this paper is called Computer Vision. This consists in the creation of models capable of analyzing images and extracting information from them. There is a lot of different information that can be extracted from an image, and that is why there are different families of Computer Vision algorithms. The main applications or families of models are the following:

- Image Classification: the goal of this type of models is to classify an image according to its content.
- Object detection: it consists of detecting certain objects and locating them in the image. For this purpose, boxes called "bounding boxes" are placed over the objects to specify their position.
- Image segmentation: this use case is similar to the previous one, but in this case, it is intended to specify to which object each pixel of the image corresponds.
- Key point detection: consists of detecting characteristic points in an image. This technique is used for facial recognition and to detect the posture of people, since it is possible to locate the location of the joints in the image.

Although the most developed application of Computer Vision so far is image analysis, it is not limited only to this task, but also aims to analyze videos.

The analysis of videos by means of Computer Vision techniques presents a complex but interesting solution to many problems that man must currently solve without the help of machines. This type of techniques are currently booming, with a great interest from the developer community due to their potential applications. The most popular application at present is the action recognition in video surveillance cameras, which is the focus of this project.

Video action detection is undergoing a great development in recent years, in which numerous models are emerging that seek to address this problem in different ways. Some of these models are limited to study each frame of the video separately, while other more ambitious models aim not only to extract information from the frames, but also to study their evolution over time.

The first approach, that of only extracting information from the frames, is the simplest and currently the most developed. This consists of using Computer Vision techniques for image analysis and applying them to each of the video frames. This could lead to misunderstanding what action is taking place, since the model will not be able to extract information from the movement of the person, but only from his situation in each frame separately. This makes the model rely heavily on the environment in which the action is taking place to determine the action. However, this technique has great application in other video analysis problems, such as object detection or person identification.

On the other hand, the second approach mentioned for video analysis using Computer Vision consists not only in extracting information from each separate frame, but also how the scene evolves. This approach to the problem of action detection is more complete than the previous one, and allows to really focus on the human action and not so much on the situation in which the person is in the image.

Two different methods are currently being used when analyzing videos in this way: a first approach that consists of pre-processing the images to extract features from them to later introduce the processed data into a model, and a second approach that consists of using only Deep Learning with the raw data, and having the model itself extract the features.

The first approach is to perform pre-processing in the images to extract some relevant characteristics, and use the new data to train a neural network. An example of this is the publication "Real-Time Action Detection in Video Surveillance using Sub-Action Descriptor with Multi-CNN" [2]. In this case it is proposed to process each frame to extract from each of them the position and movement of the person, and then feed a Convolutional Neural Network with these new processed frames.

The second method mentioned above makes it possible to use the raw videos, i.e. without the need to perform any prior transformation on the input data, thanks to the use of Deep Learning. To solve the classification problem in this way, Neural Network structures such as Conv3D are being used, which allows a spatio-temporal analysis of the videos. This type of models have been emerging for only 3 years and are still in the research and development phase. Some of the models worth mentioning are the Conv + LSTM (which has been developed in this project), Conv3D, Two-Stream Networks and Two-Stream Inflated 3D ConvNets. [3]

The use of Deep Learning has made it possible to combine several types of networks to extract desired features from both the frames and the temporal component. An example of this is the possibility of detecting key points of people (their joints) and studying their evolution over time or even carrying out a segmentation of the image and extracting the pixels that correspond to people to later analyze them temporally.

In addition to these Deep Learning algorithms mentioned above, it is worth mentioning the so-called Optical Flow. This model aims to know the movement of objects in the image by studying the "movement" of the pixels that make up the image, i.e., it is to find in the next frame the pixel values found in the current frame, and thus compare the initial position and the next by simply observing their position in both frames. This can allow not only to know the movement of objects in a video, but also to predict how these objects will move in subsequent frames.

In this project, it has been decided to analyze both the frames and the time sequence because, although it is presented as a more complex solution, it presents a higher accuracy in videos with static environments, that is, in those videos that are recorded from the same camera positioned in the same place, thus offering a better solution for video surveillance applications, which is the scope of this project. In addition, after exploring the two mentioned methods, it has been decided to make use of Deep Learning, since the previous extraction of features can bias the decisions of the final model.

Something to have in mind is that the fact that the action to be detected occurs in video surveillance camera recordings increases the complexity of the model. This is due to the fact that it is not possible to use the information provided by the environment, as will be discussed later. In addition, recordings from security cameras are usually restricted, so there are not enough videos available on the Internet for the action to be detected in this case. It is also to be mentioned that the quality of these videos is usually very low.

II. METHODOLOGY

The development of this project can be divided into four successive parts, which make up a Computer Vision pipeline:

- Creation of the dataset.
- Processing of the videos that make up the dataset.
- Design and implementation of a functional model.
- Training analysis.
- Model optimization.

The first task of the project is to create a balanced (with a similar amount of videos of each action to be detected) and meaningful dataset, in order to provide the model a sufficient and unbiased amount of data so that it can generalize, and thus be able to determine the action shown in videos that are not part of the dataset with which it has been trained.

Before starting the creation of a dataset, it is necessary to understand those characteristics of the dataset that could lead to poor results in model training. In addition to the need of a balanced and meaningful dataset so that all classes are adequately represented to obtain good prediction results in all of them, the influence of the environment must be considered in order to develop a model that is agnostic to it, as in the case of this project. This need for an environment-agnostic model is due to the fact that what is intended to be detected are actions in video surveillance cameras, in which there is no change of environment, since they are pointing to a fixed point.

This is opposite to the vast majority of current video action detection solutions, which rely heavily on the environment to detect actions, just as the human brain does in many cases. This problem is presented in the paper "Why Can't I Dance in the Mall?" [4], which explains that, in a similar way to people, models that have been trained on actions in different environments are able to predict the action without actually looking at the person, but by analyzing the environment. This effect can be understood in the following images, where the person performing the action is hidden by a green block, but nevertheless we know which action is being performed in each case thanks to the environment:

Figure 1. Images hiding the person

Not being able to support the decision in the environment increases the complexity of this project considerably, since the model must be able to fully understand human movements, as if it were playing a mime game. This means that the model needs to extract more and more complex characteristics, and so the algorithm must be more complex and the amount of data required is greater.

Having this in mind, the first task to be performed to generate the dataset is to search the Internet for those databases already created that may be useful for the problem to be solved (detection of a specific action in video surveillance cameras). These must be filtered to obtain only those videos of interest, i.e., those containing scenes of the actions to be detected. In this case, there are no databases available on the Internet that contain the desired action, so the search was continued in public repositories that could contain such videos.

Finally, after a search of several public repositories, videos were found on YouTube that contained the desired class, although it is to mention the poor quality of these videos. To generate the dataset, a code was created to download both videos and lists of videos from YouTube, which made it possible to obtain the content quickly and automatically. After this procedure, a total of 210 videos were obtained.

Once the videos that will compose the dataset are available, it is necessary to process the data. This consists in choosing the video section in which the specific action takes place, discarding the rest of the video to avoid confusing the model. In addition, the videos must be formatted so that they all have the same dimensions, since the model requires that all input data has the same format. For this, the videos must be scaled, modifying their frame height and width dimensions and the number of frames in the video. It is also a good practice to keep the number of frames per second constant, so that the time shift between each frame is constant in all videos. After doing some research, 10fps was found to be a good value to be kept.

Once the videos were formatted, different techniques were carried out to increase the amount of data available in the dataset. This is a technique called Data Augmentation. For this purpose, codes were implemented to generate a mirror mode of the video (symmetry on vertical axis) and videos with Gaussian noise. This last form of Data Augmentation was developed based on the idea of Adversarial Training. With the implementation of these methods, the dataset size was increased to a total of 630 videos. These videos were divided between the training, validation and test sets, with a proportion of 60% of the videos for training, 20% for validation and the remaining 20% for testing.

Once the creation of the dataset was completed, the model design and implementation proceeded. The model designed to try to predict video actions follows a Conv2D + LSTM structure. Following this structure, three different models were created. The first one was designed using the Keras library. After studying the flexibility provided by the PyTorch library, it was decided to migrate the model to this framework. The initial model developed in PyTorch follows the following structure:

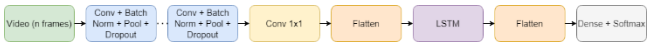

Figure 2. Conv2D + LSTM model architecture

The purpose of this model is to extract features both from the video frames, which is handled by the convolutional block (in blue in the image), and from the evolution of these frames over time, which is handled by the LSTM. It is worth to mention the use of a 1x1 convolutional layer, which makes it possible to reduce the number of input channels to the LSTM, thus drastically reducing the number of parameters of this layer. This data flow and the different dimensions of the data as it passes through the model are shown below:

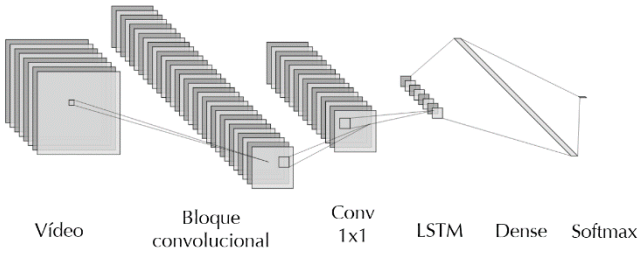

Figure 3. Conv2D + LSTM characteristic dimensions

Numerous modifications were made to this initial structure in order to achieve better results. Some of the most outstanding modifications are the elimination of most of the Pool layers, the elimination of the Dropout layers due to their low effect on the network and the implementation of a convolutional block that follows a structure based on the residual networks (ResNet).

Despite these modifications to the architecture and a search for optimal hyperparameters, it was not possible to avoid overfitting. This led to the implementation of a third network, for which Transfer Learning was used. In this case, Transfer Learning consisted of introducing a pre-trained ResNet18 network as an estimator, so that it extracts the necessary features from the images and passes them as input to an LSTM, resulting in a ResNet18 + LSTM model:

Figure 4. ResNet18 + LSTM model architecture

After the creation of the model, and during the iteration on the hyperparameters of the model, different training optimization methods were developed, which allowed to reduce the training times. These developed techniques are hardware acceleration using GPU and the implementation of Mixed Precission.

The latter method allows training the network with both float32 and float16 weights while ensuring model stability. This is achieved by evaluating the size of the weight, and if its significant figures can be correctly represented in float16, that data format is used, and otherwise float32 is used. This allows not only to increase the training speed by reducing the time required to perform operations with the weights of the network, but also to reduce the computation requirements, which makes it a very interesting technique to apply in Machine Learning models.

Regarding the evaluation of the model, several techniques were used to know the training status during training and its subsequent analysis. During training, the accuracy and loss values after each epoch are shown, as well as the time taken to complete each epoch in order to understand the impact of the different optimization methods on the model. As for the a posteriori training analysis, the precision and loss plots of the model are generated, which allow to analyze the behavior of the model and thus to be able to decide which will be the next modifications in the architecture and in the hyperparameters' values. In addition, the confusion matrix is analyzed, and activation maps are generated for each of the layers, which allows understanding what the functional block is focusing on when extracting features from the frames.

The analysis of these graphs allows understanding the current state of the model and how it is behaving, which is essential to continue with the iterative process of optimizing its architecture and hyperparameters.

III. RESULTS

The results obtained in this project are not limited to the results of the model, since in order to obtain them, a complete Computer Vision pipeline has been developed. That is why in this section it is also worth mentioning the developments that have been completed in each of the parts of the pipeline, which have led to the implementation of a model capable of extracting features from videos.

First of all, a database has been generated with videos found on the Internet. For this purpose, not only an exhaustive search was performed to collect 210 videos of a type of action with scarce data on the Internet, but also, to speed up the download, a code was implemented to download both individual videos and lists from YouTube automatically.

Once all the data was collected, a code was implemented that allows splitting the videos to extract the frames in which the action of interest takes place, and that tags the video according to whether that action appears or not.

Then, having the videos classified, two Data Augmentation techniques were successfully implemented (mirror mode and Gaussian noise), which allowed to increase the size of the dataset by 300%, up to a total of 630 videos. In addition, functions to format each of the videos were implemented, so each one of them had the same dimensions, so that they could all be analyzed by the model to make predictions.

As far as the model is concerned, three different models were developed. The first model was developed using the Keras library. This model was eventually discarded, and a migration was made to PyTorch, where a similar but more complex model was developed. This Conv2D + LSTM model presents the following results:

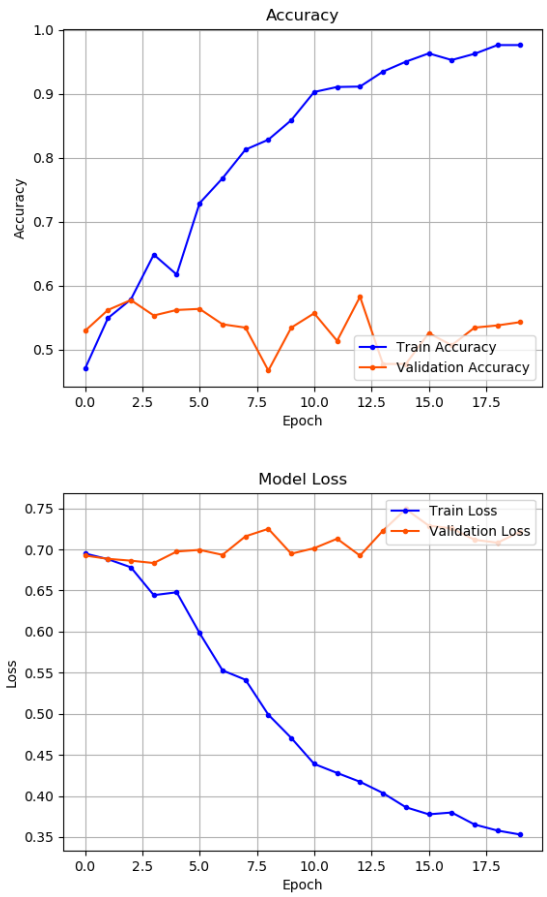

Figure 5. Conv2D + LSTM accuracy and loss plots

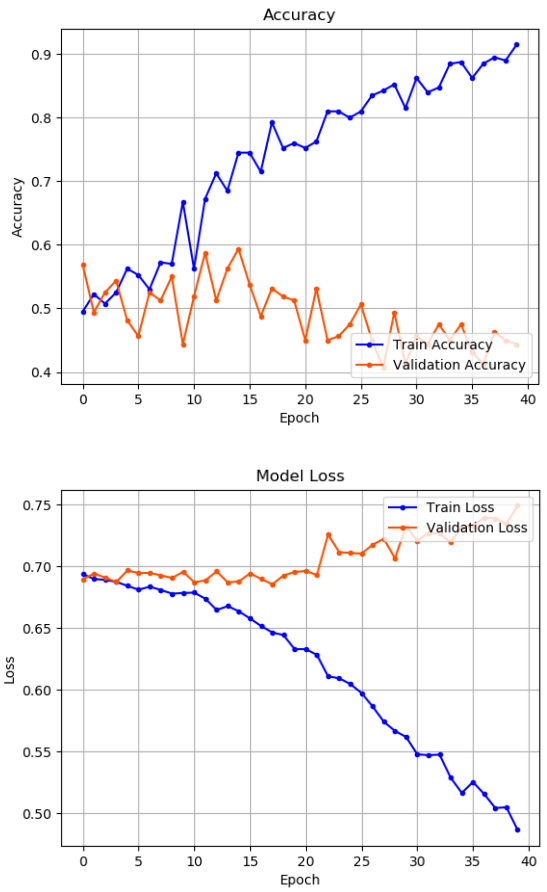

Figure 6. ResNet18 + LSTM accuracy and loss plots

As can be seen, there is a great deal of overfitting in the model despite the various techniques that have been implemented to avoid it. In the quest to reduce this overfitting, Dropout was introduced, L2 Regularization was implemented, the data volume was increased (by using the Data Augmentation techniques mentioned above) and the complexity of the model was tuned and tested. Since the implementation of these techniques did not favor the generalization capability, it was decided to implement Transfer Learning to try to reduce this overfitting.

This decision to make use of Transfer Learning is what led to the development of the third and last model implemented in this project. This model is a ResNet18 + LSTM. In this way, the convolutional block was replaced by a pre-trained ResNet18, which allowed to substantially reduce the number of network parameters to be trained, since the ResNet18 is used as an estimator, so that it extracts the features from the frames, but is not trained with such data. The results are presented next:

As can be seen, despite the fact that the beginning of training presents higher validation accuracy values, it has not been possible to avoid the overfitting that was also present in the previous model. In this model, the same techniques have been applied as in the previous one to try to avoid this overlearning, but despite their implementation, the problem has not been solved.

Even though this problem persists in the results, it is to be noted that the fact that the model does overfitting indicates that it is a functional model (it is capable of extracting features), which is a starting point for further development in the search for better prediction results. This also shows that the previous steps developed in the pipeline work correctly, and allow the model to learn to extract features from the data.

Finally, it is worth mentioning that in addition to all of the above, this project has successfully developed training optimization techniques such as hardware acceleration, which has greatly reduced training times through the use of GPU and Mixed Precission, which has reduced training times and computational requirements.

IV. CONCLUSIONS

Aiming to automate the process of action recognition in video surveillance cameras, this project develops a Computer Vision model. For this purpose, different architectures have been studied and it has been chosen to implement a Conv2D+LSTM model.

In this project, not only the model has been implemented, but also a complete Artificial Vision pipeline has been developed, from data collection for dataset training to model optimization, including video processing, model architecture design, training analysis...

When analyzing the results of the model, it can be observed that it is able to extract features, which indicates that the developed pipeline works properly. Despite this, it has not been possible to avoid overfitting in the model, even though numerous techniques were applied to try to solve this common problem in Machine Learning models. On the other hand, these results lead us to think that there may be an error in the data (either in quality, quantity or in the way of classifying them) or that the model designed is not suitable for this type of problem, not being able to extract the adequate features despite the multiple modifications that have been made to its hyperparameters. This second idea leads to the fact that in future developments of this project, different network structures will be proposed in order to face the problem with a different approach than the current one.

One of the major problems and challenges of this project has been the creation of the dataset. This is due to the scarcity of data on the Internet and the impossibility of generating such data. On the other hand, the quality of the videos available on the Internet is very low, in many cases not exceeding a quality of 240p. This may have greatly limited the learning capacity of the model, since, if the data with which it is trained are not of high quality, neither can its predictions be of high quality.

The development of Machine Learning algorithms is a complex and time-consuming task, which is not always successful, but we must always learn from the previous developments to know what steps to take next to achieve a successful implementation.

V. REFERENCES

[1] «Historia de las Redes Neuronales,» Itnuevolaredo, [En línea]. Available: http://www.itnuevolaredo.edu.mx/takeyas/apuntes/Inteligencia%20Artificial/Apuntes/tareas_alumnos/RNA/Redes%20Neuronales2.pdf.

[2] C.-B. Jin, «Real-Time Action Detection in Video Surveillance using Sub-Action,» Arxiv, [En línea]. Available: https://arxiv.org/ftp/arxiv/papers/1710/1710.03383.pdf.

[3] J. C. y. A. Zisserman, «Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset,» Arxiv, 12 Febrero 2018. [En línea]. Available: https://arxiv.org/pdf/1705.07750.pdf.

[4] e. a. Jinwoo Choi, «Why Can't I Dance in the Mall? Learning to Mitigate Scene Bias in Action Recognition,» Arxiv, 11 Diciembre 2019. [En línea]. Available: https://arxiv.org/pdf/1912.05534.pdf