ACTION RECOGNITION IN VIDEOS WITH A 3D CONVOLUTIONAL NEURAL NETWORK (3DCNN)

Alberto CASTILLO RODRIGUEZ[1]

[1]Universidad Pontificia Comillas ICAI, 28015 Madrid, Spain

Corresponding author: Alberto CASTILLO RODRÍGUEZ (e-mail: alberto_castillo@alu.icai.comillas.edu).

This work was supported in part by the Chair for Smart Industry ICAI.

ABSTRACT This project aims to develop algorithms based on 3D convolutional network models that are able to detect actions in security camera videos. For this purpose, a database has been generated from public sources such as UCF101 and YouTube and the bias and ethics of the samples have been studied. As for post-processing, the samples were trimmed and classified, the database was split, data augmentation techniques were applied and a motion filter was developed. Using PyTorch as a framework, the pipeline was developed by optimizing the available processing resources. After extensive study, the Adam optimizer and the Cross-Entropy loss function were chosen. Hyperparameter optimization was performed manually with the expectation of automating the process in the future. Saving and visualization of the results were done with the TensorBoard library, which allows comparing different runs and configurations. The algorithm must overcome the intrinsic problems of the task, such as the appearance of several people in the same frame, the fact that they are in different categories and, unlike other action recognition problems, the environment is not representative of the action. The developed model consists of convolutional layers with regularization methods. All objectives were met with the exception of obtaining a model with high accuracy. This is due to the lack of sufficient videos and the reduced quality of the videos. In addition, computational capacity constraints limited the complexity of the models, which prevented overcoming these obstacles.

INDEX TERMS 3DCNN, Action Recognition, AI ethics, Deep learning, PyTorch, TensorBoard.

I. INTRODUCTION

The project arises as a result of the great progress that artificial intelligence has been making in recent decades and the great impact it can have on the business world and on society. These advances in the early days of machine learning were in the field of structured numerical information, but with the development and cheapening of computing power, the field of artificial vision has been growing.

The possibilities of being able to identify actions in video are enormous, as it would revolutionise sectors such as security, where the detection of robberies or violent actions could be improved and where a single person is required to monitor a large number of cameras at the same time. This work aims to respond to this need for innovation in the field of artificial vision and where different companies can find competitive advantages if they achieve this goal.

This progress not only has to be reflected in the accuracy results, but also has to be done taking into account the resource limitations, both in terms of the possibility to get videos and the processing capacity to train and run the network. If the model is resource intensive the images have to be sent to an external server, known as Cloud Computing, as opposed to processing it locally, known as Edge Computing.

This is why the 3 Dimensional Convolutional Neural Network (3D CNN) structure has been chosen as a basis for testing different architectures. They have proven to be the most efficient architecture to date to process images. One advantage of using convolutional networks is that they allow to increase the processing speed with the use of the GPU, as they are optimised to work with matrices, which is precisely the basis of convolutional operations, and in this way to lighten the load when training and applying them. Moreover, compared to recurrent networks, they are more efficient due to the lower number of weights required.

II. OBJECTIVES

The objective is to develop an algorithm capable of detecting shoplifting actions in security camera videos. To do so, a database will be built and an algorithm will be developed.

A. OBTAINING THE DATABASE

The database will be created on which the different algorithms and preprocessing methods will be tested. It must meet the following criteria:

- It must be balanced
- It should preferably be publicly accessible
- Must be well classified
- Must be trimmed
- They should be of the highest possible quality
- There should be a considerable amount of data

B. ALGORITHM DEVELOPMENT

Different algorithms using the 3D CNN structure partially or in its entirety will be developed according to the following criteria:

- Must be lightweight
- Programming language: it must use the Python programming language [1] and one of the two most important Deep Learning frameworks such as Tensorflow [2] or PyTorch [3] to facilitate its integration in a final application.
- That it has an accuracy of at least 70 % in validation.

III. SOLUTION

A. AI ETHICS

When designing a system that implements artificial intelligence algorithms, it is very important to mention ethics. These types of systems eliminate human intervention in decision making and therefore run the risk of creating systems that intentionally or accidentally make decisions based on race, gender or other categories. All this was taken into consideration when generating the database to make it as balanced as possible to prevent biases.

B. DATA

1) DATABASE GENERATION

Public databases such as UCF 101 [4] or Kinetics [5] were explored, looking for categories of security camera videos. In the absence of a reasonable amount to get a robust database, a Python script was made to download YouTube videos with the aforementioned characteristics. The total number of videos extracted was 167 with very diverse characteristics of resolution, frames per second (FPS) and scenarios or environments.

Analysing the bias of the database, taking into account the context of the use of security camera videos, we observed a large imbalance in the representation of different skin colour tones and cultural styles. Therefore, although the database can still be used to test different types of algorithms, the algorithms trained from the database cannot be used in production environments.

2) DATABASE POST-PROCESSING

Once we have obtained the videos, it is necessary to trim them and classify them correctly as normal videos and videos in which the action is committed. To do this, a list was generated in a .txt file with the respective names of each file and the seconds we want to tag next to the name of the file. After this classification, a Python script automatically trimmed the videos and classified them according to the tag. The end result was the creation of 204 videos with normal situations and 208 videos with the action. To improve the quality of the database, the following considerations were taken into account.

Video clipping

The trimmed videos must have exclusively the action to be detected, resulting in clips of very variable length in the range of 2 to 10 seconds. To avoid a bias for the duration of these trimmed videos, for each one that contains an action as far as possible we have tried to obtain another one without any action with the same duration.

Bias of background and people

One of the ways in which the algorithm could falsely classify the videos could be by detecting characteristics of the people who commit the action versus those who do not. That is why we tried whenever possible to obtain snippets of each video showing the same person committing the action and in a normal situation.

FIGURE 1. Example of how the action can be determined without seeing the person committing it. From left to right: jumping, golf, canoeing and ice skating [6].

This is also true of the recording environment, as reflected in the paper "Why Can't I Dance in the Mall? Learning to Mitigate Scene Bias in Action Recognition" [6]. This paper demonstrates that both algorithms and humans make use of context to determine, without seeing the person, the action being committed (figure 1). By removing this bias, it is possible to improve the generalisation of the algorithm but it also makes the task more difficult by reducing the information available. The bias has been eliminated by generating videos of normal situations and with the action in the same environment avoiding the memorisation of features of the environment for classification.

Movement filter

Another way to solve this problem would be to capture only the motion that is detected in each video. The viewing angle and position of the images are constant and therefore it is only the people that modify the pixels over time as they move. By calculating the absolute difference between consecutive black and white frames, we get black pixels where there has been no movement and white pixels where there has. To be more resilient to noise, it is preferable to calculate the difference with a weighted moving average of the previous frames although this produces the appearance of the ghosting side effect as can be seen in the figure 2. This operation can be performed with the OpenCV cv2 library [7]

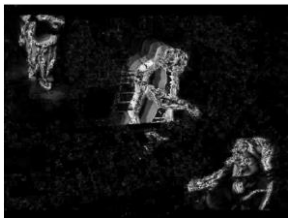

FIGURE 2. **Frame of one of the videos when applying the motion filter.**

Division of the database

The last step to prepare the database for the learning process is its division into three groups where we have tried to balance the categories as much as possible:

- *Train*: this subgroup will be used to train the model automatically. It is therefore the largest, with 70% of videos, 144 normal and 141 with action.
- *Validation*: this subgroup will be used to manually or automatically tune the hyperparameters of the model. It will contain 15% o of the videos, 31 normal and 35 with action.
- *Test*: this last subgroup is used to validate the results of the model. The hyperparameters should not be modified to improve the result in this subgroup. It will have 15% of the videos, 31 normal and 30 with the action.

Data Augmentation

The aim of this techniques is to increase the amount of data available by applying modifications to the existing data to allow the algorithm to better generalise to data that is slightly different from the existing ones. Modifications can be:

- *Spatial*: videos are mirrored horizontally to keep the vertical axis in the same orientation.
- *Colour*: modifying the colour spectrum, contrast and brightness among other parameters allows the algorithm to be more resilient to these variations between videos.
- *Noise*: introducing noise in the images allows the algorithm to work with different qualities of videos and be more robust to these variations.

C. PIPELINE

The job of the pipeline is to manage the training process as optimally as possible. This ranges from performing the relevant transformations on the data, loading it into the RAM, CPU or GPU, performing the training, validation and testing operations and finally saving the results and the model.

1) HARDWARE LIMITATIONS

The first aspect to consider when designing the pipeline is the available resources during the whole process and the requirements of the algorithms during training. The goal is to maximise the training speed by optimising the use of the system hardware. In this project, there are two limiting resources when it comes to minimising training time: RAM and GPU.

2) MACHINE LEARNING FRAMEWORKS

To facilitate and optimise the definition of the algorithms, the training process and the reproducibility of the whole process, several frameworks have been developed. PyTorch [3] allows us to easily access the training loops in order to modify them according to our needs, both to test new architectures and to visualise the data.

3) DATA LOADING

Once we have the entire database generated, the interface must be configured to read the data, process it and load it into the training device. As there is not enough RAM or space in the GPU memory to hold the entire database, the strategy to follow will be to load the data little by little as it is needed during the training process. Two classes will be used for this purpose: Dataset and DataLoader.

4) OPTIMIZER

The process of training an artificial intelligence algorithm consists of defining a loss function that represents how well it performs its task and improving this result over training epochs. The problem with minimising this function is that in general they are not convex and therefore do not have a single minimum but have local minima. The strategy with which one tries to find the global minimum is very important to avoid getting stuck in local minima.

Optimisers are tools to solve all these problems. Most optimisers calculate the learning rate automatically and apply the gradient to the model making it learn. A good optimiser is one that trains fast and at the same time prevents the model from getting stuck at a local minimum. Adam's optimiser Adam [12] is the one that performs best in most [9] scenarios and that is why it is the one that has finally been applied.

5) LOSS FUNCTION

When evaluating how well an algorithm has performed a task, a metric must be defined to quantify it mathematically.

One must choose one that represents the loss function. This function will be the one that is minimised during the training process and therefore should not only represent how well it has performed the classification, but also its confidence in both hit and miss. The Cross-Entropy Loss function is the one selected because:

- Using the last Softmax layer allows us to easily check the probabilities that the modelgives to each category. This way you can intuitively visualise what the model does.
- It can be applied to both binary and multi-class classification problems. This allows usnot to have to modify the structure of the model in case of adding more categories.
- It is a function extensively tested in the literature with proven results.

6) HYPERPARAMETER TUNING

Hyperparameter optimisation is one of the most important phases in the training process of a model. While parameters are normally trained automatically, hyperparameters are those that describe or shape the training process and are therefore fixed during the training process. The impact of each is different and there are different optimisation processes. During this work, novel methods were considered but, due to the computation limits, a manual approach was finally used.

D. ALGORITHMS

Once we have designed a pipeline that is able to provide the data and manage the training process, we must face the design of an algorithm able to solve the problem of action detection in security camera videos.

1) INTRINSIC PROBLEMS OF THE TASK

When solving problems with artificial intelligence models, great care must be taken when analysing the difficulties that the data may present. By considering all sources of bias, it is possible to target the model design to avoid these problems at their root. The following challenges have been identified:

- Multiple categories in the same video.
- Overlapping subjects.
- Non-identifying environment.

2) MODEL

Once we have identified the difficulties, several models or combinations of them have been proposed to overcome them. In addition to these considerations, the limitations of computing power and the generated database have been taken into account.

Intrinsic problem elimination strategies

The main intrinsic problem to be solved that could present the greatest difficulties is that of the appearance of multiple persons with different categories. To solve it, two different methods are proposed: use of the motion filter and the set of YOLO v4 models with Deep Sort.

Main classification algorithm

The aim of this work is to develop an algorithm based on the 3DCNN layer, which consists of convolutional operations of two spatial dimensions and one temporal dimension. There are many models using this structure with varying complexity but due to the large number of parameters involved and the limited computational capacity, it has been decided to explore the simplest architecture applying it as a proof of concept.

This architecture is composed of 5 convolutional layers in series followed by 2 fully connected layers with the softmax operation at the end to calculate the assigned probabilities for each class (figure 3). This architecture is described in the article "Multi-cue based 3D residual network for action recognition" [10] as the base model of this layer.

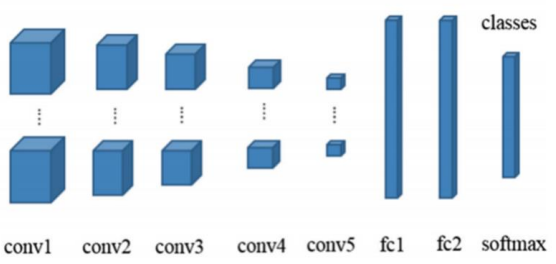

FIGURE 3. **Layers of the 3DCNNN model [10].**

This architecture has been modified to solve several of the classic problems that Deep Learning models have:

- Overfitting
- Underfitting
- Computational load
- Internal covariate shift

The layers used to combat these problems are:

- *Max pooling:* Max pooling is a discretisation process that aims to reduce the size of your input by transforming the window of the operation to the maximum value contained in it. This reduces the number of parameters required while preserving the characteristics of the region and reduces underfitting by providing an abstract form of the sub-regions to which the filter has been applied.
- *Dropout*: This regularisation technique aims to prevent overfitting. It is achieved by ignoring a percentage of units during the training process chosen randomly at each epoch. Ignoring them allows not modifying their value during the process and eliminating co-dependencies between parameters.
- *Batch normalization*: This layer was designed in the article "Batch Normalization: Accelerating Deep

Network Training by Reducing Internal Covariate Shift" [11] to address the problem of internal covariate shift. It allows to speed up training by reducing the number of epochs and smoothing the loss function.

IV. RESULTS

The model development consisted of three parts: database construction, pipeline development and model design and training.

A. DATABASE

A total of 412 videos were obtained, 204 of them with the normal category and 208 of the action to be detected. These were divided into three partitions: train, validation and test with 70%, 15% and 15% respectively. They were ensured to be balanced in categories in each subdivision. Many of them were of poor quality, so it was decided to create a second database with only those that met minimum resolution requirements and met a clearer definition of the action. This second database contains 210 videos, 105 from each category, with similar percentages and balanced categories.

In terms of the ethical quality of the database, there is no great ethnic and socio-cultural diversity due to the greater restrictions on this content in Western countries and their greater capacity to apply them. This is why it is not recommended for use in developing production models.

B. PIPELINE

After applying all the techniques mentioned above, an efficient pipeline has been achieved with the available resources. The correct loading and transmission of data to the algorithm has been checked, as well as the application of the relevant transformations before training. It has not been possible to apply the YOLO algorithm together with Deep SORT due to unreliability during the clipping of individuals. This is due to the low quality of the videos and the excessive overlapping of multiple persons. The Adam optimiser was finally used due to its easy implementation and good results in most deep learning applications. As for the loss function, the function applied was Cross-Entropy Loss due to its versatility in terms of the number of classes with which it can be applied, the good results in the literature and allowing us to directly observe the probabilities that the model assigns. Hyperparameter optimisation was performed manually, making decisions based on information recorded in TensorBoard from past training.

C. MODEL

After many iterations on the model structure and hyperparameters, overfitting has been achieved on the training data, implying that the model has sufficient complexity to learn the problem. This can be seen in figure 4, where the loss function decreases over epochs in the training data but increases in the validation data.

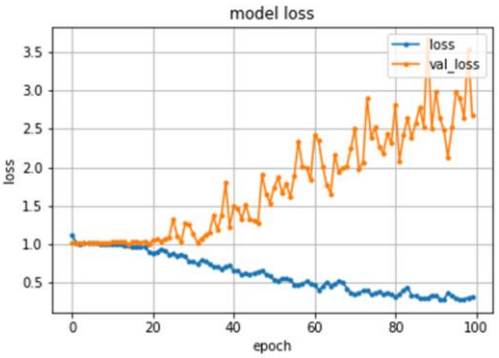

FIGURE 4. **Evolution of the loss function.**

Despite this, the model has not been found to perform better in the classification process than making random predictions, resulting in an accuracy of around 50% in validation as is the case when we have a balanced two-category database. These observations are reflected in figure 5, where we obtain an accuracy in the training data close to 100 percent due to overfitting but in the validation data it oscillates around 50 percent.

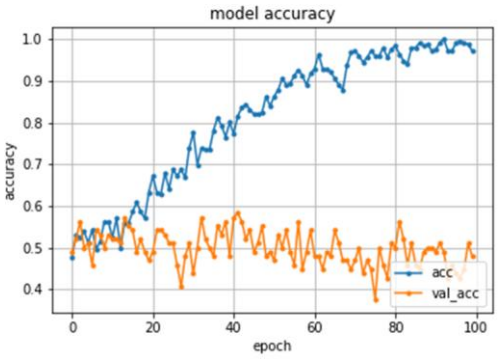

FIGURE 5. **Accuracy evolution.**

Plotting the confusion matrix (figure 6), it is observed that the algorithm prefers to classify as the action the normal cases leading to false positives. In this case, it is preferable to false negatives which are also found in lower proportion.

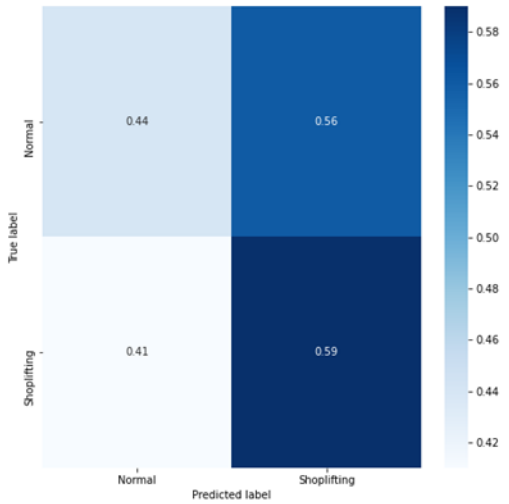

FIGURE 6. **Confusion matrix.**

V. CONCLUSIONS

From the results obtained, different causes can be extracted as to why a model capable of making predictions.

A. DATABASE

Regarding the database, the following objectives have been met:

- Build a balanced database.
- Use publicly accessible data.
- Correct classification of videos.
- Cropped to only represent the section where the action appears.

Individual people have not been finally cropped with YOLO together with Deep SORT as it is not very reliable when performing the operation well in situations with many people in close proximity. Regarding the quality of the videos and the number of videos found, the severe restrictions prevent finding quality videos in quantity, limiting the variety available and with major shortcomings in terms of ethics. It is very important, especially when dealing with security camera videos, to take into account the elimination of bias by race, gender and socio-cultural aspects. Because of this, algorithms developed with this database cannot be applied in production.

B. MODEL

In terms of the model, the following objectives have been met:

- Use the 3DCNN layer.
- To be light enough to be trained with the available resources.
- The use of Python as programming language and the PyTorch framework.

The accuracy of 70% has not been achieved due to several reasons. The first is the lack of sufficient data to train a model to solve such a complex task as action detection. Second, there were not enough resources available to train more complex models that are able to compensate for the problem of data sparsity. Finally, the low quality of the videos limited the algorithm's ability to appreciate the detail of the action, especially when applying transformations.

C. RECOMMENDATIONS FOR FUTURE STUDIES

To solve the problems encountered, it is recommended that the following points be explored.

1) INCREASE THE SIZE OF THE DATABASE

It is necessary to increase the quality and quantity of videos in the database. This will allow to avoid bias, to increase the image quality of the clippings of each person and to take into account the ethical aspect. In order to fulfil all this, it is proposed to generate the videos by using actors to make sure that the ethical obligations are met and that the current legislation is respected.

2) TRANSFER LEARNING

In case of not being able to increase the database, it is proposed to train a model on other databases of human actions with more populated categories and with better quality in order to transfer what has been learned to the database.

3) HUMAN DETECTION

Continue to explore human detection algorithms such as YOLO + Deep SORT to isolate people. If their reliability could be increased, the model could be trained with only one person per video. This would minimise environment bias or the appearance of people from different categories in the same image.

4) COMPUTATIONAL POWER

Access to more computing power is necessary in order to train models with videos. Increased computational capabilities would allow significantly reduced training times for testing new architectures and more rigorous and automated hyperparameter optimisation.

REFERENCES

[1] «Python», Python Software Foundation. Available: https://www.python.org/. [Last access: 06/06/2021].

[2] «Tensorflow: Open Source Machine Learning Framework for Everyone», Tensorflow. Available: https://www.tensorflow.org/. [Last access: 06/06/2021].

[3] «PyTorch: An open source machine learning framework that accelerates the path from research prototyping to production deployment», PyTorch. Available: https://pytorch.org/. [Last access: 06/06/2021].

[4] «UCF101 - Action Recognition Data Set», Center for Research in Computer Vision (CRCV). Available: https://www.crcv.ucf.edu/data/UCF101.php. [Last access: 05/06/2021].

[5] «Kinetics,» DeepMind. Available: https://deepmind.com/research/open-source/kinetics. [Last access: 05/06/2021].

[6] J. G. C. M. J. C. &. H. J. B. Choi, «Why Can't I Dance in the Mall? Learning to Mitigate Scene Bias in Action Recognition.», arXiv, 2019 - https://arxiv.org/abs/1912.05534.

[7] «OpenCV, Open Source Computer Vision», OpenCV. Available: https://docs.opencv.org/4.5.2/. [Last access: 15/06/2021].

[8] «Introduction to loss functions» Algorithmia. Available: https://algorithmia.com/blog/introduction-to-loss-functions. [Last access: 05/07/2021].

[9] «Various Optimization Algorithms For Training Neural Network», Sanket Doshi. Available: https://towardsdatascience.com/optimizers-for-training-neural-network-59450d71caf6. [Last access: 10/07/2021].

[10] M. W. R. C. Z. W. M. W. X. &. P. J. Zong, «Multi-cue based 3D residual network for action recognition», Neural Computing and Applications, vol. 33, nº 10, pp. 5167-5181, 2020.

[11] S. &. S. C. Ioffe, «Batch normalization: Accelerating deep network training by reducing internal covariate shift,» International conference on machine learning, pp. 448-456, 2015.

[12] D. a. B. J. Kingma, «Adam: A method for stochastic optimization.», arXiv, 2014 - https://arxiv.org/abs/1412.6980.