Hand gesture recognition in industry

Dumoulin William, Thiry Nicolas, Slama Rim
Haute Ecole d'ingénieurie industrielle de l'Henallux Pierrard , Virton, Belgium

***Abstract*— With the 4th industrial revolution and the increased use of cobots in the industries comes many opportunities for new generation control panels. The work presented here aims to develop a deep learning model to recognise 10 different gestures. The data set fed to the model was completely created. The videos were taken from a computer webcam and then processed to remove the noise created by the background by isolating the movement of the gray scale images. The spatio-temporal features are extracted by the combination of 3D convolution and LSTM layers. On 8 out of the 10 gestures, the recognition rate is more than 90%. Furthermore, ~~the model is usable in a real time performance.~~**

I. INTRODUCTION

While working with image recognition by a computer, the main challenge that we can face is in the case of a real time application. The computational time has to be low enough to keep the program running without lags.

There are many applications for a gesture recognition program in different areas: to open the doors or call for help in a supermarket or to remove the need of a remote control for the television at home. ~~As industrial engineer, we want a program that can be use in our field of expertise. Therefore,~~ our program is developed for an application in industry. For instance, to control a cobot without a control panel.

~~The data fed to our model are homemade. 1500 videos were created by 12 different person, performing 10 gestures at least 3 times each. We created multiples models to find the one that suits the best to our application.~~

~~The interface mainly shows a feed-back of the camera with the predictions from the model. There is also the possibility to create new videos and add new gestures.~~

II. STATE-OF-THE-ART

Many different techniques were developed in the past years. One of the first was an ~~algorithm~~ that separates the fingers from the palm. The position of the hand on the static image was then deduced depending on the number of fingers found by the algorithm and their relative position. A much more developed method is to apply a squeletonization method [4]. The idea is to reduce an object to lines that are only 1 pixel wide. The images are therefore easier to analyse by the model. This method is applied to the videos before the training and the tests. An algorithm to create the skeleton of a shape need a Voronoi tessellation [1] but it's a heavy algorithm that slows down the entire process.

N. Luthfil Hakim, T. K. Shih et al [3] created a model of gesture recognition for a television application. They used an RGB and a depth camera to get more valuable information. Their model is composed by multiple blocs of 3D convolution layers combined with dense and max pooling layers. The last part of their model is the Long Short-Term Memory layers [6] that finds features that are time related which are crucial when dealing with videos.

III. METHODS

A. Data set

~~To get any gestures possible~~ we created our own data set with our gestures. To simplify the process, we made a python script to save the video in a folder and the label in a CSV file. The program is made to create a lot of data in the smallest amount of time. The user can switch between different labels by pressing a key and can start a new video with an other key. See details on Figure 1. There is a delay of one second before the video is taken to give the time to the user to place himself. The computers camera have a frame rate of 25 FPS so each video last 2.4 seconds and contains 61 frames. With a good rhythm, a new video can be made every 5 seconds.

A data set need recordings of different people otherwise the model would only recognise the gesture of the person that did all of the data set videos. The smallest data set need at least 12 different people and each one does each gesture twice. This first project has 10 different gestures which make a minimum of 240 videos for the training set. It is of course not enough so multiple people filmed more videos. The result is a training set of almost 1 500 videos.

Fig. 1. New video program

B. Data processing

The videos from the data set have 61 frames with a dimension of 640 pixels wide by 480 pixels height and 3 layers of depths for the red, green and blue channels. The first step of image ~~processing is to convert the image from RGB to gray scale. The OpenCv library in python has function~~

~~to do the conversion.~~ Each pixel of the new gray image is calculated with the colors channels of source image with this formula: Gray $\leftarrow 0.299 * Red + 0.587 * Green + 0.114 * Blue$

The second step is to isolate the movement in the images. To do that, each image is compared with the next one. The movement of the last frame can't be isolated because there are no more frame to do the comparison. That make 60 frames of movement. The pros of keeping only the movement is to remove the background. Of course, to make the training data, the background need to be still, otherwise it will be considered as part of the gesture. This technique still reduces the noise that can be produced by the background.

The model doesn't get 60 frames for each video because it would make an over complicated model. The third step is to isolate a sample of the frames. The choice was made to take only 10 fps, that make a sample of 24 frames.

The fourth step is to resize and normalize the videos. The normalization take all the pixels and map them from 0-255 to 0-1. After all the processing, the videos have 24 frames with a dimension of 120 pixels wide by 90 pixels height. All the pixels have a value between 0 and 1.

IV. MODEL

A. Inspiration

Our first model is based on the work of N. Luthfil Hakim, T. K. Shih et al[3]. They have an RGB and a depth camera. The images from the two cameras are processed separately and combined after the LSTM units [2]. The image processing is made with multiple 3D convolutional layers. The team had a data set with thousands of videos and got a result of 91%.

B. Model improvement

The results of our first architecture were not conclusive. The problem was probably the small data set and the large amount of parameters. We simplified their model to create our own. We trained 15 different models, each one had a small difference compare to the last one in order to understand how each parameter affects the model. Its accuracy raised from 26% to 83%. A clear improvement in the quality of the models can be seen by the increase of their accuracy : the 4th model showed an accuracy of 46% while the 12th showed an accuracy of 79%. ~~We finally arrived to a model with 3,446,046 parameters, the details of the model can be seen on Figure 2.~~

C. Training

The videos are stored raw, they need to be processed when they are loaded to train the model. The data set of 1 500 videos is too heavy to be loaded in the RAM so the videos

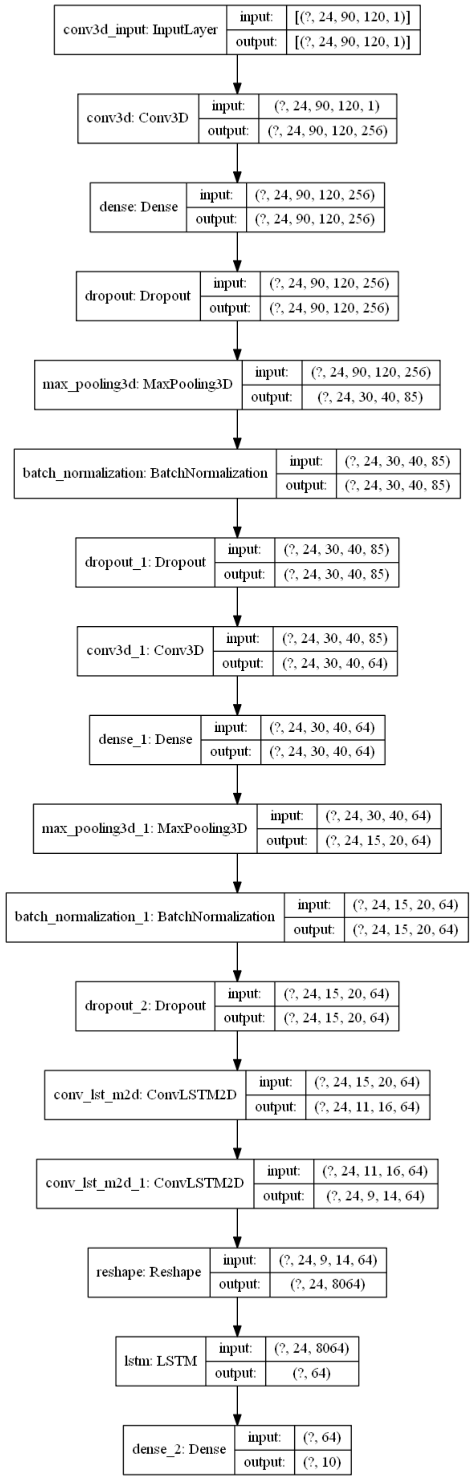

Fig. 2. Architecture of our model

have to be load by package. The label of the videos are saved in a list which is randomly mixed. The videos can then be loaded in the order of the list. It's important that the model is trained on different gestures every time.

To train the model faster, the data processing and training are on two different threads. The process thread loads two packs of 40 videos and waits for the training part to be done with the previous pack. When the model is done training with a pack, the next pack is transferred to the training section and the thread loads the next one.

The 2 tasks are handled by the computer in parallel, that's what is called thread. It allows to reduce the overall time of computing. The videos are loaded by packs to avoid overloading the computer's memory.

The packs have enough information to allow multiples training. However, if a model is trained multiple times on the same pack, the risk of overfitting increases.

To reduce this risk, the model should be trained on the entire data set once before being trained on videos that it already saw.

The ideal process would be to train the model on the entire data set in a single pack. Since it is not possible due to memory size, the videos should be loaded by smaller packs. Each one of them should then train the model once and leave its place to the next pack. If multiple epochs are mandatory, the whole process needs to start again.

Multiple epochs can be necessary when the model has a bad accuracy.

Our algorithm doesn't use that process because each videos would be processed multiple times and the training would be unnecessarily long. Our model is trained on each pack three times in a row before moving to the next pack.

D. Architecture

The model is visible on the Figure 2, this section will explain the meaning of each layer.

TABLE I
EDGE DETECTION KERNEL

-1	-1	-1
-1	8	-1
-1	-1	-1

1) 3D Convolution [5]: In the convolution layers, new channels are created with different kernels. A kernel is a squared matrix, whose size is usually 3x3 or 5x5. To analyse videos, it can be better to use 3D convolution layers, then the size are 3x3x3 or 5x5x5. The goal of this layer is to find the features of the images or videos. For example the Table I shows an edge detection kernel.

2) Dense: The dense layer is also called fully-connected layer. Each node of a layer is connected to all the nodes of the next layer. In our case a node is an entire image. Each channel of the last dimension is also an image. For example, the dimensions of the first dense layer are (?x24x90x120x256) which stands for 24 moments in time, 90 pixels height by 120 pixels wide and 256 images. The different images for one moment in time are created by the convolution layer. ~~(The "?" stands for the number of videos in a batch, this is not a parameter of the model)~~

The connection is defined by a weight with a value between minus one and one. In this architecture, the dense layer is placed after a convolution one. We can compare the dense layer to a filter. The meaningful channels created in the 3D convolution layer have a weight with a high value, the useless ones have a weight close to zero.

Of course this is a deduction of the system, we cannot prove it because the thousands of kernels and weights in the model does not make sens for a human.

3) Dropout: When this layer is placed after a dense layer, it removes a certain percentage of the weights by giving them new random values. The goal is to reduce the risk of overfitting.

4) Max pooling 3D: This layer reduces the size of the data set by keeping the max value of a section of an image (or multiple images when it's 3D). A 3D Max pooling layer with a matrix (3x3x3): The dimension go from (24,90,120,256) to (24,30,40,85) because the three last dimensions are divided by 3. This layer simplifies the data by keeping only the meaningful pixels.

5) Batch Normalization: The layer transforms inputs so that they are standardized. This means that they have a mean value of zero and a standard deviation of one.

6) LSTM [6]: Long Short-Term Memory layers are a type of recurrent neural network capable of finding time dependent features. This layer (and the ConvLSTM) is very important in this application because there are needed to differentiate "Swipe to the left" and "Swipe to the right" for example.

7) ConvLSTM2D: Like the classic LSTM, the convLSTM is capable of learning order dependence in sequence but this layer is specialized for the videos analyses because it's the combination of LSTM and convolution layer.

In this architecture, two ConvLSTM 2D are stacked. The first ConvLSTM layer provides an output for each input while the second layer gives only one output for a sequence of input.

8) Reshape: The layer doesn't learn anything in the training process. It only changes the dimension of the data to pass them to the LSTM layer which needs specific input.

V. ~~RESULTS~~

A. Accuracy

The Figure 3 shows the confusion matrix of our best model. The numbers on the columns and lines of the confusion matrix correspond to this list:
0 = Closing one hand.
1 = Opening one hand.
2 = Clap with both hands.
3 = The right hand swipes to the left.
4 = The left hand swipes to the right.
5 = Crossed fingers.
6 = The hands start closed in the middle of the image and they go in opposite directions while they open.
7 = The right arm starts horizontally and rotates around the elbow, it ends when the hand is pointing to the sky.
8 = The left arm starts horizontally and rotates around the elbow, it ends when the hand is pointing to the sky.
9 = Say "hello" with one hand.

The first two gestures are often confused with one another. This is probably due to the proximity of theses gesture. We can see that closing a hand and opening a hand is done at the same places on the videos. Furthermore, the difference between closing and opening a hand is much more subtle than the difference between swipe left and swipe right.

For all the others gestures, the model is very efficient. On the live test we often got a precision of +90%.

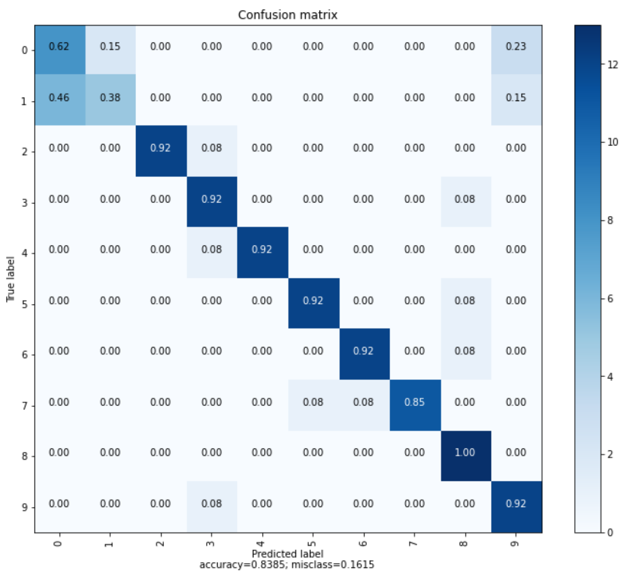

Fig. 3. Confusion matrix

VI. ~~INTERFACE~~

The interface was coded using the python libraries tkinter and openCV.

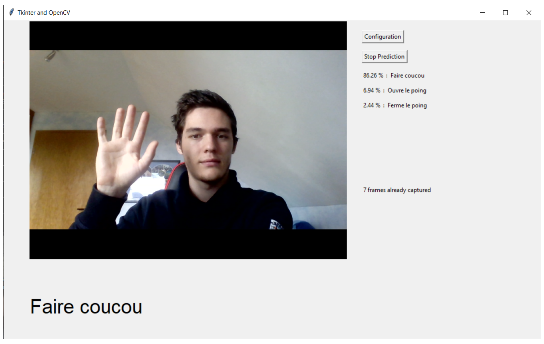

Fig. 4. First Window

The main window, that can be seen at the Figure 4, shows the live feed from the camera. It also shows the prediction that have the highest score at the bottom of the screen. Furthermore the window shows the 3 first predictions for the same video with the percentage of recognition. If the first prediction given by the model is lower than 60%, the predictions are not displayed and a message is written on the screen explaining that the model did not recognise the gesture.
The last information that is shown is the number of frames that are already captured in the video. This allows the person in front of the camera to perform the full gesture on one video. As soon as the video is fully captured, another video is being captured.
The main window also shows two buttons. The first one freezes the prediction. In other words, the videos are no longer captured and the prediction stays on the window. The second button opens a second window. While the second window is active, the first is completely freezed to avoid wasting the resources of the computer.

Fig. 5. Second window

The second window displays an example of each gesture that the model is able to recognise with its name. It also allows the user to add a new gesture. The first step is to create a new label or title for the gesture and add it to the list through the text field on the window. If the label already exists on the list, a message is displayed on the screen.
On this window, there are two buttons. The first one opens program to record a video and add it to the training set. This program is explained at III METHODS : A. Data set. The second button retrains the model. This operation takes a lot of time. Therefore, a safety needs to be installed to avoid miss-clicks.

~~VII. LIMITS OF THE MODEL~~

~~While doing the tests of the complete project, we stumbled across ways to perform the gestures that our model had trouble recognising.~~

~~*A. Distance from the camera and speed*~~

~~The first issue that we can address is the distance between the camera and the hands. This issue comes from the way in which the data set was created. The persons performing the gesture were placed at the same distance of the camera so the model was not trained to recognize the depth of the videos.~~

~~One of the solutions is data augmentation. It is a way to create new videos that are based on videos from the data set. This method can then be used to increase the distance from the camera by resizing the images and adding black pixels on the sides to keep the same data size. The black pixels can be added on both sides or just one : for the height, placing the pixels on the top and the bottom would keep the video on the center of the resized video while adding them only on the bottom would place the video on the top of the new video. Furthermore, the number of pixels added on each sides doesn't have to be the same. For example, 10% of the pixels can be added on the left of the video and 90% on the right. This technique allows to create a large amount of new video based on the original data set and should solve the issue of the distance between the hands and the camera.~~

~~Another solution to this issue is to use a second camera like the article [3] that we based our work on. This second camera would be a camera that doesn't record a rgb image but the depth information. In addition to this camera, a modification of the data set creation protocol is needed. The gesture would need to be performed at multiple distances from the camera.~~

~~The second issue that we can address is the speed at which the gesture is performed. This issue is the same as the previous one except that it as to do with time and not space. The solution remains data augmentation. A number of images are removed from the videos. Those images should not follow one another in order to keep the movement recognisable. The more images are removed, the faster the gesture in the future video is going to be. Therefore, multiple gesture speed can be create based on the data set. When a certain quantity of frames is removed, the same number of images needs to be added in black images to keep the same data size. The same comment as the distance problem can be made. The black images can be placed at the beginning or at the end of the video. Another possibility is to place 20% of them at the beginning and the 80% at the end for example. This methods allows to create a large amount of videos with the gesture done at different speeds and at different moments of the video.~~

~~These two issues can be resolved by creating more data the same way the initial data set was created. However this would take more time than using data augmentation.~~

~~*B. Orientation*~~

~~The orientation can be defined as the angle between the camera and the plane created by the shoulders and the feet of the person. When the orientation is different from 0°, the model has difficulties to recognise the gestures. The project is set in the context of the industry and it allows us to impose the orientation. Furthermore, this issue can not be resolved with data augmentation. A solution would be to create more videos in the original data set and to impose multiple orientation angles.~~

VIII. CONCLUSIONS AND FUTURE WORKS

A. Conclusions

The first method we considered was the skeletonization. The pros is a simpler model with less parameters because the skeleton is only one pixel wide with no colors information. The cons is that the process of skeletonize an image is very heavy and the gain of the simpler model is lose by the computational time of the skeletonization. Furthermore, the skeletonization is heavy in the training and the testing part while a big model takes time to train but is not a problem for the testing part.

To get rid of the noise made by the background, we isolate the movements in the videos. Then, we reduced the frame rate and the size of the images. The resulting data is fed to the model. The 1500 videos used to train the models were enough to get an overall accuracy of 84% and 90% for 8 gestures out of 10.

B. Future Works

~~If we have the possibility to train our model on a good GPU (or even TPU), we can create other models to check our deductions.~~ We can train models with different LSTM characteristics. ~~We can replace the 3D Convolution by a classic 2D convolution to make sure that there are useful.~~ After we have found the best solution, we can modify the characteristics of the data set fed to the model. Theses tests could also help us find if it is better to have a higher definition or a higher frame rate. Maybe there are both too computationally expensive and therefore not an interesting modification. When the characteristics of the new model are well defined, we can fix the limits of the model by increasing the size of the data set. The process is called data augmentation and is well described in the section ”Limits of the model”

REFERENCES

[1] A. Iraola, ”Skeleton based visual pattern recognition: applications to tabletop interaction”, San Sebastian, 2009.

[2] A. Ullah, J. Ahmad et al, ”Action Recognition in Video Sequences using Deep Bi-directional LSTM with CNN Features”, *IEEE Access*,vol. 6, Seoul Korea, Peshawar Pakistan,2017.

[3] N. Luthfil Hakim, T. K. Shih et al, ”Dynamic Hand Gesture Recognition Using 3DCNN and LSTM with FSM Context-Aware Model”, *MPDI*, 2019.

[4] R. Ogniewicz, M. Ilg, ”Voronoi Skeletons: Theory and Applications”, *IEEE*, Zurich, Switzerland,1992.

[5] S. Ji, W. Xu, M. Yang et al, ”3D Convolutional Neural Networks for Human Action Recognition”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*,vol 35, NO.1, 2013.

[6] S. Hochreiter, J. Schmidhuber, ”Long Short-Term Memory”, *Neural Computation*,vol. 9, Massachusetts, USA,1997.