DEEP LEARNING FOR FACE RECOGNITION IN SURVEILLANCE VIDEOS

by

Paul-Darius Sarmadi

A dissertation submitted in partial fulfillment of the requirements for the degree of
Master of Engineering in Computer Science

Examination Committee:	Dr. Matthew Dailey (Chairperson) Dr. Mongkol Ekpanyapong Dr. Manukid Parnichkun
Nationality:	French
Previous Degree:	Bachelor of Science in Math, Physics TELECOM SudParis, France
Scholarship Donor:	Telecom SudParis, France - AIT

Asian Institute of Technology
School of Engineering and Technology
Thailand
May 2016

Table of Contents

List of Figures

Chapter 1

Introduction

1.1 Background

Cameras are everywhere, in front of stores, businesses and on our streets. They are used for several purposes:

- After a crime has been commited, the video can be used as clues or proof. The goal is to get more information on the person or on the event that has occurred.
- In the few minutes after a crime, to intercept the person who committed it. Some police are always scanning surveillance videos, monitoring subways, streets, and so on to be able to catch the perpetrator in the corridor or before he or she could escape.
- Before a crime, typically, to dissuade potential thieves from stealing anything. Sometimes, even cameras that are not working, not recording, or are fakes can be effetive deterrents.

In the context of global terrorism, the problem of recognizing a previously identified terrorist, hoping to follow him or her through video cameras through several feeds sadly appears to be a key issue. More generally, following the path of any criminal using surveillance videos sounds like a main concern.

In this work we can assume that police face several difficulties:

- The number of cameras they may have access to, depending on the locality's policy. This number may be very high, and is growing everywhere very quickly.
- The low quality of the video. It is sometimes very hard to recognize a person in a video with low resolution.
- The crowd. It takes a second to recognize a previously identified thief on a video when he or she is alone. What if there are 30 other people on the video, in a crowded street for example?

Recognizing previously identified criminals in video surveillance feeds adds to the above-mentioned difficulties, requiring a great deal of human resources. These issues lead to a simple conclusion. It is expensive work in terms of time, money, and human resources, but it is nevertheless extremely important to uphold the national security of any country.

Automating the face recognition process in surveillance video seems to be an interesting answer to address some of these problems.

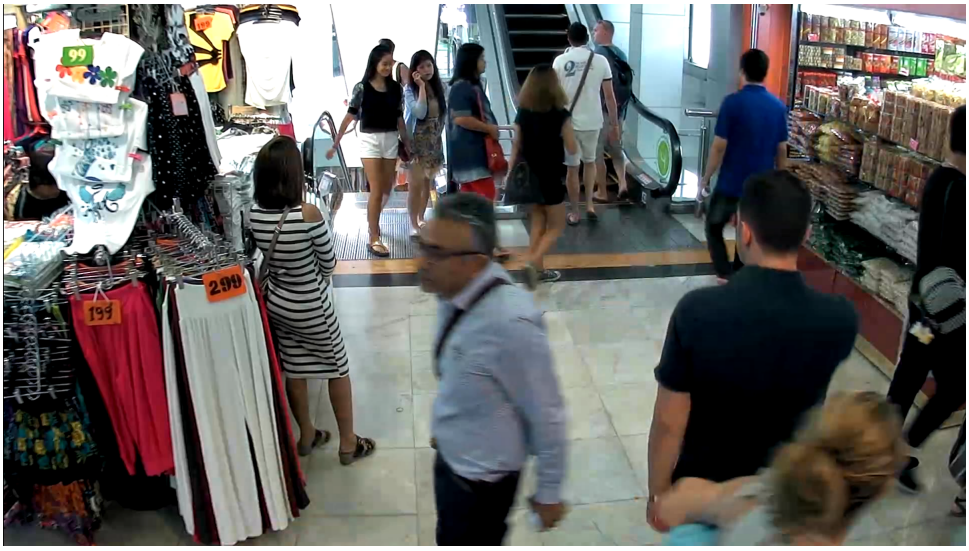


Figure 1.1:: A frame from a video in the MBK dataset.

1.2 Problem Statement

Face identification is a key machine learning issue at present. The best results obtained this far use a "deep" neural network (any artificial neural network with more than one hidden layer). In 2014, DeepFace reached an accuracy of 97.35% on "the Labeled Faces in the Wild (LFW)" dataset for face verification (Taigman, Yang, Ranzato, Wolf, 2014). In 2015, FaceNet reached a 99.63% accuracy on the "LFW" dataset for identification (Schroff, Kalenichenko, Philbin, 2015).

In surveillance video, face recognition faces difficult issues such as blurriness, low resolution, and unexpected poses of faces. Though the problem of face recognition in surveillance video has already been studied, deep learning techniques may improve performance in this scenario.

The goal of this research is to build a deep neural network for face recognition on surveillance videos. This model should be based on the latest algorithms and best practices deep learning offers.

1.3 Objectives

The objectives of this research are to develop and evaluate methods for a database already provided for the study. This database contains recorded videos from the surveillance camera network at the MBK shopping mall in Bangkok. Figure 1.1 shows a frame of one of these videos. Three of our researchers appear over several seconds in some of the videos. They are walking like anyone else in the mall.

Choosing this database serves one goal: to be placed in the theoretical situation of policemen looking

for one or several criminals in the streets or in the corridors of public places, using a few sample pictures of them to try to track and find out where they have gone.

Our researchers are playing the role of the criminals we are looking for. The main objective is to use deep learning techniques to build an automated solution for the task of finding people we have a few photos of over a larger collection of video streams.

More precisely, considering this database, there are three main objectives in this research study:

- Create a database of images containing faces extracted from the surveillance videos.
- Build a deep neural network for face recognition. The general idea is that the network should be provided with a few photos of our researchers from a training set, and try to find these individuals in a testing set.
- Testing the resulting model. Compare the model's accuracy with other experiments using different techniques.

1.4 Limitations and Scope

There are two main limitations to this research:

- Time. This project is three months long. There are many deep learning techniques existing, but due to the time limit, not all can be explored.

- Material limitations. The laboratory has provided a single NVIDIA GeForce 780 GTX GPU card for the computation. This places some limitations on the mini-batch size for stochastic gradient descent. Preliminary results show that the size will be limited to 20 samples, while the usual size is 128.

  Deep learning gives astonishing results in the task of face recognition. That is why despite those limitations, we should be able to get an interesting model for the context of surveillance videos.

1.5 Research Outline

I organize the rest of this dissertation as follows.

In Chapter 2, I provide a review of the relevant literature.

In Chapter 3, I propose my methodology.

Chapter 2

Literature Review

As said in the previous section, the goal of this study is to exploit *deep learning* algorithms for face recognition in surveillance videos.

2.1 About Deep Learning

According to "Deep Learning Methods and Applications" (Deng, Yu, 2014):

> Deep learning is a set of algorithms in machine learning that attempt to learn in multiple levels, corresponding to different levels of abstraction. It typically uses artificial neural networks. The levels in these learned statistical models correspond to distinct levels of concepts, where higher-level concepts are defined from lower-level ones, and the same lowerlevel concepts can help to define many higher-level concepts.

One of the most well-known deep learning algorithms is the *Convolutional Neural Network* (LeCun, Bottou, Bengio and Haffner, 1998), a variant of the multilayer perceptron (Rosenblatt, 1961) which involves the use of convolutional layers (see Figure 2.1). Biologically inspired by mammalian visual mechanisms, convolutional neural networks are historically known for an application called *LeNet*, able to recognize hand-written digits (LeCun et al., 1989). In the last 10 years, the interest in convolutional networks has grown quickly. In fact, with the evolution of technology — basically massive usage of always more powerful GPU cards — it has been possible to use these networks more widely and for various tasks. They have been used widely for human face classification tasks and have given impressive results.

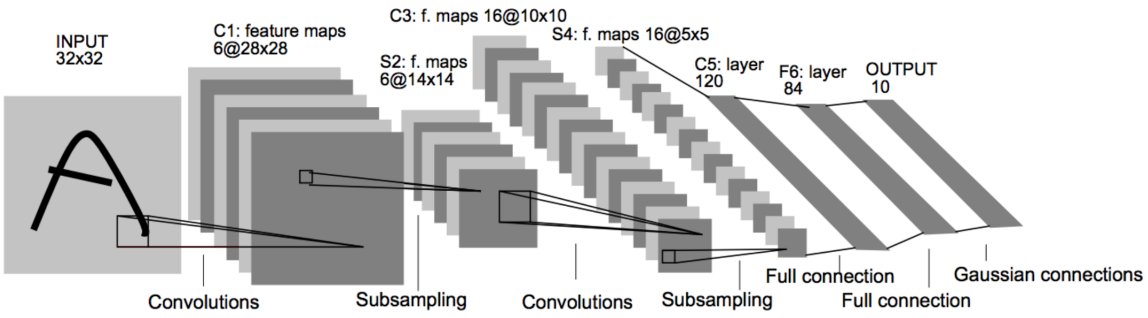


Figure 2.1:: Architecture of LeNet-5 for digit recognition. Extracted from LeCun, Bottou, Bengio and Haffner (1998).

2.2 About Convolutional Neural Networks

A convolutional neural network is generally made of one or more convolutional layers followed by a standard neural network. Convolutional layers are similar to standard layers. The

difference is that in this specific case, we connect only adjacent neurons of a layer to the next one, whereas, standard layers connect all the neurons of a layer to the next one. Then, most frequently a subsampling of the convolutional layer is applied and the output of this step is used as an input to the next layer.

The next figures show a typical convolutional neural network represented in 3 dimensions. Its role is to identify a hand-written digit in an image as an example of a particular written number. This 3D representation has been made by Harley, in 2015.

Figure 2.2 shows the full network. The bottom layer is the input, a hand-written “2” digit.

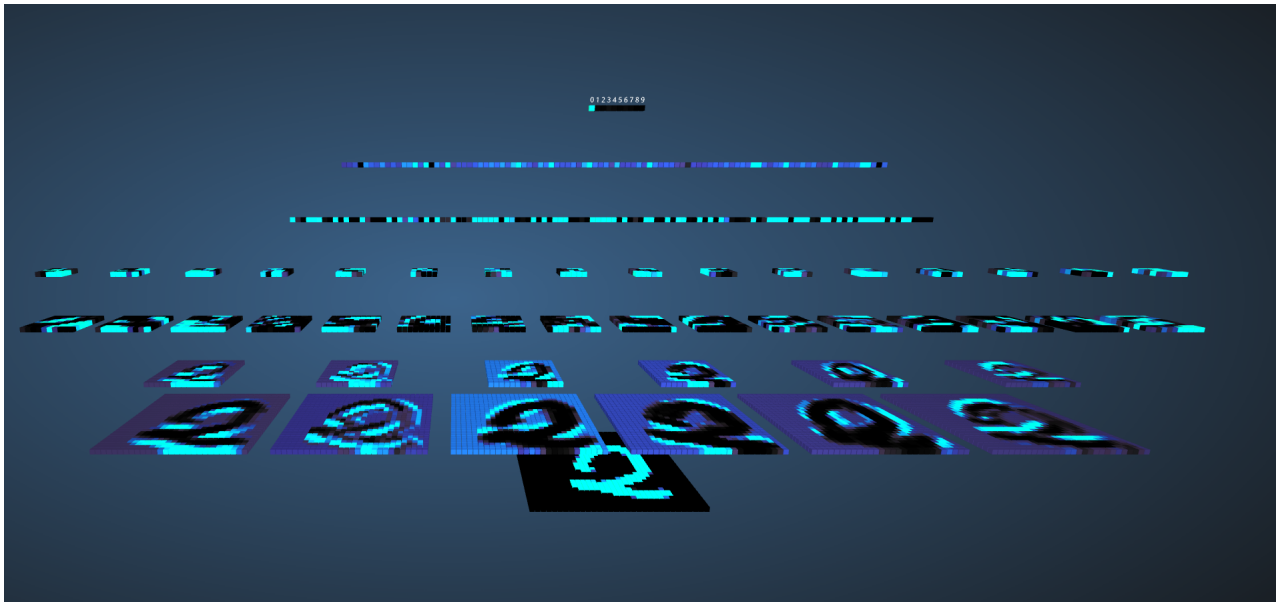


Figure 2.2:: 3-dimensional representation of a typical convolutional neural network. Generated by Harley, 2015.

The next layer is the first convolutional one. See Figure 2.3. Some restricted sub-regions of the input are received by the neurons of the next layer. Convolutional units have tied weights, so that the same processing is applied at every local subregion of the input layer. This idea is biologically inspired by cells from the visual cortex of animals, which are sensitive to fixed small regions of the visual field and perform similar computation over the entire visual field.

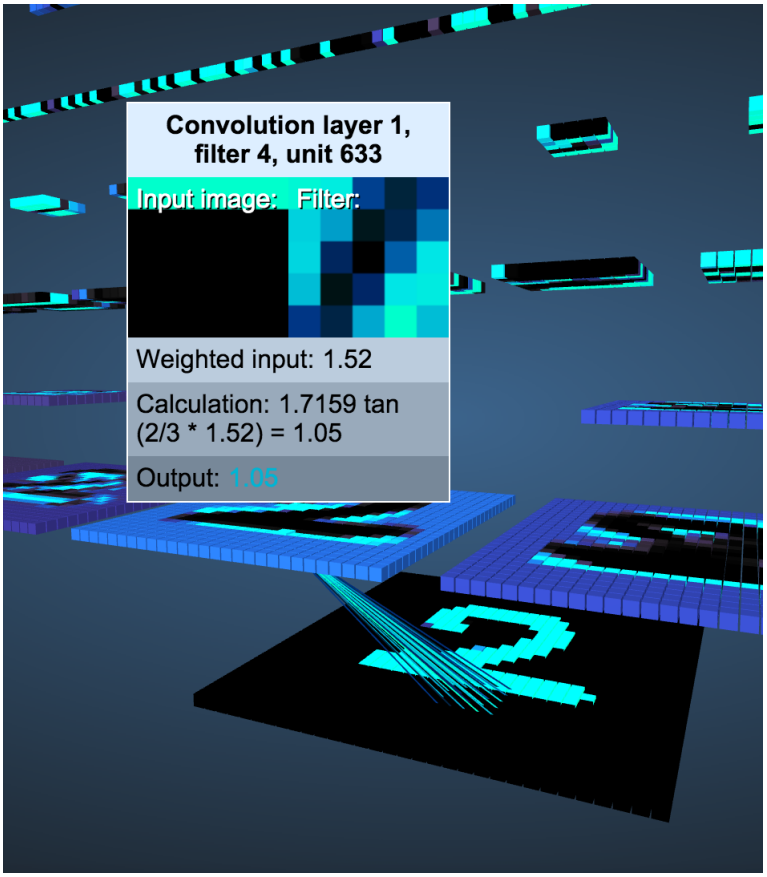


Figure 2.3:: The first convolutional layer. Generated by Harley, 2015.

In the next layer of the convolutional network, subsampling occurs. Subsampling reduces sensitivity to small translations of similar features. Different types of subsampling exist. The most used is "max-sempling" which extracts the maximum value among the pixels of a region. See Figure 2.4.

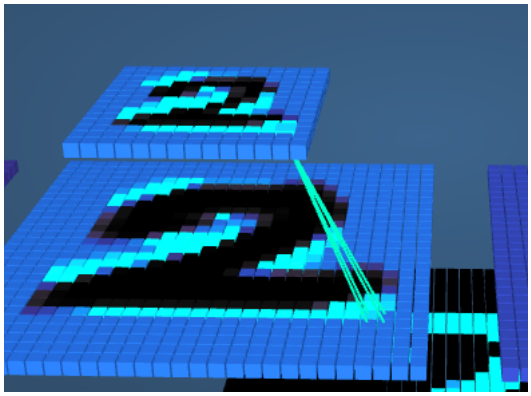


Figure 2.4:: The first subsampling layer. Generated by Harley, 2015.

Figure 2.5 represents the second convolutional layer. Its input is the output of the first subsampling layer which represents the result of the computations of the first convolutional layer.

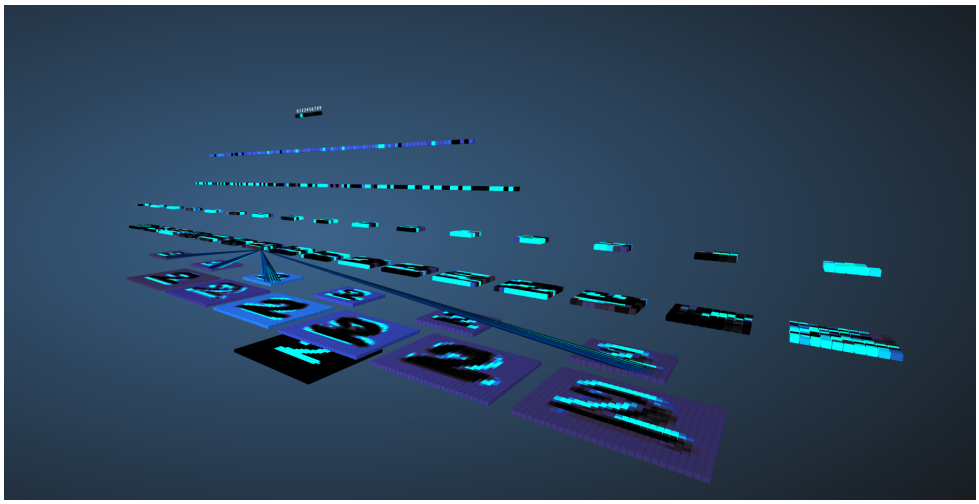


Figure 2.5:: The second convolutional layer. Generated by Harley, 2015.

The two last layers before the final output represent standard fully-connected layers. See Figure 2.6.

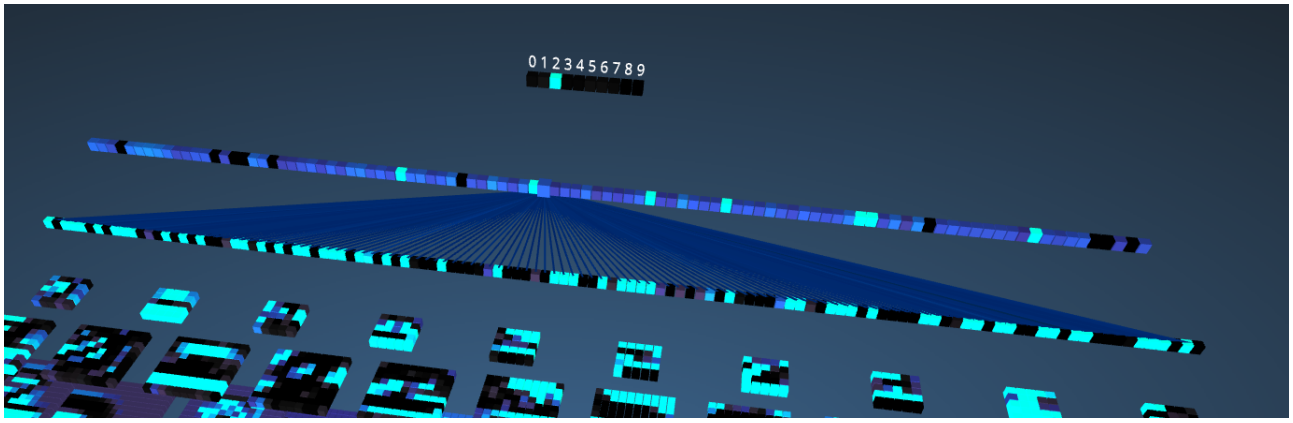


Figure 2.6:: The fully-connected layers. Generated by Harley, 2015.

2.3 Previous Work

2.3.1 Deep learning for face recognition

There are two main schemes for face recognition:

- Recognition by person identification. In this case, a network takes an image as an input and returns a label that identifies one and only one person as an output. The literature on this type of architecture is extremely abundant. For face verification, DeepFace (Taigman, Yang, Ranzato, Wolf, 2014) reached 97.35% accuracy on the Labeled Faces in the Wild (LFW) dataset. The state of the art for face identification is FaceNet (Schroff, Kalenichenko, Philbin, 2015).
- Recognition by comparison. The "Same/Not Same" algorithms. The basic idea is that the training dataset is made of pairs of images linked to the label "1" if they represent the same object — in our case, the face of a person — and "0" otherwise. In deep learning, the standard same/not same network is called a "Siamese Network" (Chopra, Hadsell, LeCun, 2005). A Siamese network is built out of two convolutional networks. The input to the first is the first image of the pair and the input to the second is the second image. The two networks share the same parameters and return two values each. Those values are used to compute an energy. If the energy is high, the two images are considered to be "very different". Otherwise they are considered to be "similar." The energy is a function of a loss function used to update the parameters of the two convolutional networks by contrastive gradient descent (Figure 4.1). Intuitively, the computed energy is similar to gravitational potential energy. If a mass is far from Earth, its GPE is high, and the mass will be considered as not belonging to the planet. If the mass is stuck to the Earth, its GPE will be low, and the mass is considered part of the planet itself.

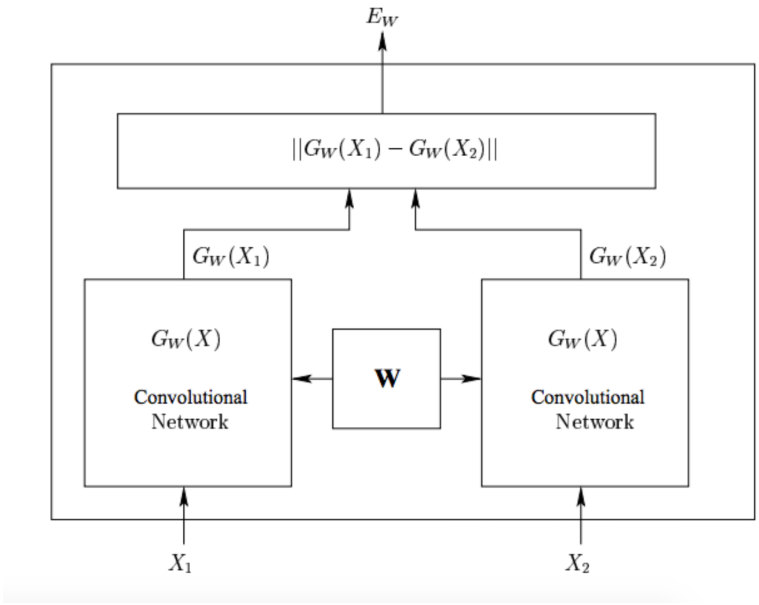


Figure 2.7:: Architecture of a Siamese network. Extracted from Chopra, Hadsell, LeCun, 2005.

2.3.2 Video-Based Face Recognition

On automated face recognition for surveillance video, a number of articles have been published. They are using a wide range of methods. Liu and Chen (2003) used a Hidden Markov Model for video-based face recognition. Le An, Kafai and Bhanu (2012) used a Dynamic Bayesian Network. Goswami, Bhardwaj, Singh and Vatsa (2014) proposed an interesting methodology. First, an algorithm extracts the most "memorable" frames in the video. Then, the chosen frame is applied to a deep learning network performing face recognition. This idea provides the state-of-the-art in low false acceptance rates.

2.4 Conclusion

Video-based face recognition is a very important area of research. Though many articles are published on this subject, some of the latest deep neural network techniques have yet to be applied to the context of images extracted from surveillance videos.

Chapter 3

Methodology

3.1 System Design

Figure 3.1 shows the design of the final product.

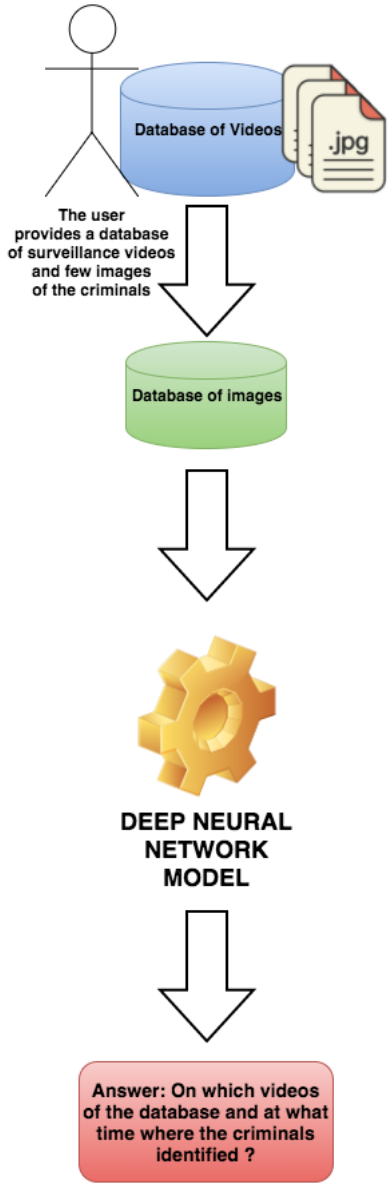


Figure 3.1:: Design of the final product.

3.2 Solution overview

The goal of this research is to build an algorithm which can detect our three researchers in the surveillance videos of a mall, as shown in the previous section. The solution is made of five steps.

- Initially, we have a database of videos.
- Then, these videos are processed to extract the faces of every person appearing in them. We have now a database of faces.
- From this database are generated some files which are necessery for the learning process.
- A model is learnt to recognize our researchers.
- The model is tested.

3.3 Solution Design

Figure 3.2 presents two main ideas. In blue, the steps described in the above section are shown. In green, the solutions used to go from one step to the next one are described.

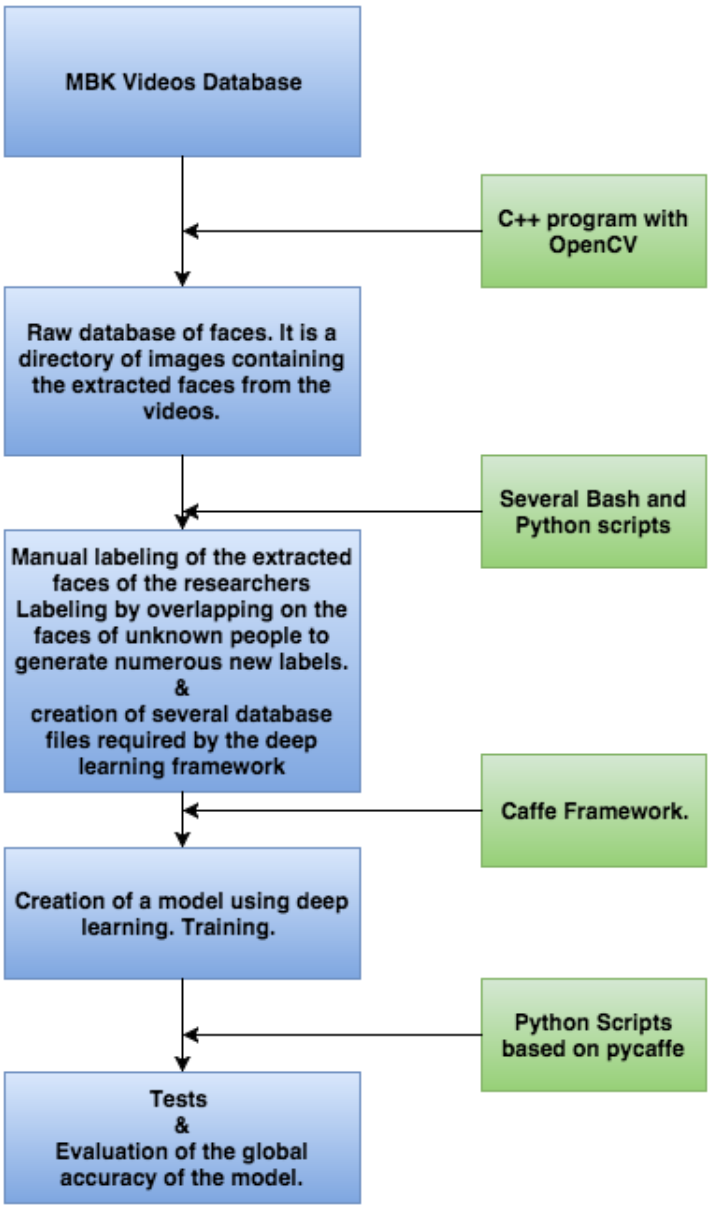


Figure 3.2:: An overview of the global design of the study.

3.4 Database

As said in the introduction, deep learning algorithms will be applied to perform face recognition in a video surveillance system. A database of surveillance videos is required to generate a training set and a testing set for our model. The database that will be used for the learning process is a set of 14 videos recorded in the MBK Shopping Center of Bangkok. The duration of the videos is variable, from a minute to around 3 minutes and 30 seconds. Three of the researchers of our laboratory appear in the videos, walking in the mall like any other person.

3.5 Raw Database of Faces

A C++ algorithm using OpenCV (Bradski, 2000) provided in Algorithm 1 will be used for face detection.

for *each video in the database* **do**
    for *each frame N of the current video* **do**
        Detect all the faces of the frame N;
        Save the P-th detection in "Database/video/FrameNFaceP.jpg";
    end
end

Algorithm 1: Face detection Algorithm

The algorithm used for face detection uses a machine learning process called Haar feature-based cascade classifiers, described by Viola and Jones (2001).

Once the process is over, a "Database" directory is created with one directory for each video. In each directory, all the faces are saved as .jpg files.

Figure 3.3:: An example of two faces extracted from the surveillance system.

3.6 Creation of database files

The framework that will be presented in the next section requires two files to work: a train.txt file and a test.txt file. Their role is trivially linked to their name in a supervised learning process.

3.6.1 Case of direct face identification

In the case of direct face identification, each of the train.txt and test.txt files share the same structure:

/adress/of/the/training/image1.jpg label1
/adress/of/the/training/image2.jpg label2
...
/adress/of/the/training/imageN.jpg labelN

The label being 1 for the first of our researchers appearing in the surveillance system, 2 for the second one, 3 for the third one, and 0 for any other person.

The labeling has to be done manually. The chosen method consists in modifying the name of the files where a researcher appears in this way:

"filename.jpg becomes Kfilename.jpg", where K is the label of the researcher.

3.6.2 Case of a "Same/Not same" model

In the case of a "Same/Not same" model, the problem is a bit different. A slightly strange structure is best for this model. A solution is to generate two files for training and two files for testing. Let's name them train1.txt, train2.txt, and test1.txt and test2.txt. Their individual structure will be as explained before.

/adress/of/the/training/image1.jpg label1
/adress/of/the/training/image2.jpg label2
...
/adress/of/the/training/imageN.jpg labelN

In the line K, train1.txt and train2.txt will contain respectively "address/to/file1K.jpg labelK" and address/to/file2K.jpg labelK. The labels will be identical. However, the images will be different. If both the images represent the same person, the label will be 1, and 0 otherwise. test1.txt and test2.txt will work the same way. Only one of the two labels will be used for the supervised learning, the other one, being identical, will not be used. The syntax of the training and testing files being fixed with Caffe, this solution seems to be the most adapted to the problem encountered with the "Same/Not Same" models.

3.6.3 Conclusion

It is a feasible task to create a serie of python scripts that will run one after the other to create the required train.txt and test.txt files. These python scripts will be indirectly launched through bash scripts. A README file will be provided to explain which commands to type and which options to select to generate the required files from the database.

The purpose of the designed architecture is to make those scripts reusable for any new database, and generalizable for an arbitrary number of labels. If a user provides another database of videos, and follows the process described in the README file, the required train.txt and test.txt files will be generated, and the models presented in the next section can be used for learning or testing directly on these data.

3.7 Model

3.7.1 Framework

The chosen deep learning framework for this study is Caffe (Jia et al., 2014). It is possible that Torch (Collobert, Kavukcuoglu, Farabet, 2011) will be used also, if useful.

3.7.2 Strategy

Different strategies have been considered for this modelisation. As explained in the previous chapter, there are two usual schemes for face recognition.

- The first one is the "same/not same" scheme. The idea is to readjust the Siamese Network described by Lecun et al. (2005) to our particular dataset.

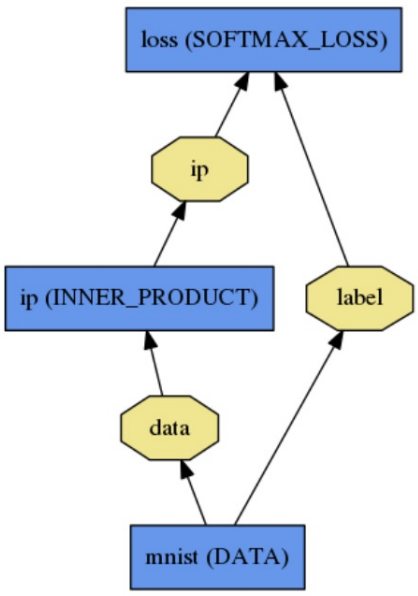
- The second scheme is direct face identification. For this purpose, the idea is to modify the last layer of a state-of-the-art deep neural network architecture for face identification. The output of this last modified layer should be binary. Either the input image represents one of our researchers —in practice, a criminal—-, or it does not. The previous layers should already be trained, and the training should be done in the last layer only.

3.7.3 The learning process with Caffe

The Caffe framework requires several files to learn the model.

- First, train.txt and test.txt files, which give paths to all the images with their corresponding labels.

– Second, a train_test.prototxt file which describes the architecture of the network. This file is written with protobuf. According to the README file available on the project's GitHub, "Protocol Buffers (a.k.a., protobuf) are Google's language-neutral, platform-neutral, extensible mechanism for serializing structured data.". Figure 3.4 shows how a logistic regression classifier is easily defined in a train_test.prototxt file with Caffe.




```
name: "LogReg"
layer {
  name: "mnist"
  type: "Data"
  top: "data"
  top: "label"
  data_param {
    source: "input_leveldb"
    batch_size: 64
  }
}
layer {
  name: "ip"
  type: "InnerProduct"
  bottom: "data"
  top: "ip"
  inner_product_param {
    num_output: 2
  }
}
layer {
  name: "loss"
  type: "SoftmaxWithLoss"
  bottom: "ip"
  bottom: "label"
  top: "loss"
}
```

Figure 3.4:: Logistic regression classifier definition with Caffe. Extracted from the official website of the framework.

– Third, a solver.prototxt file, which contains information on the batch size or on the variables related to the used loss function.

The learning process produces two files: A .caffemodel and a .solverstate. These files are used to store the value of the parameters of the designed model after a number of steps of learning chosen in the .solverstate file.

3.8 Testing

As we just said, the output of a face identification model should be binary. A python script to test the accuracy of the model can be written with no difficulty. On the contrary, the input of a Siamese Network is two images and the output is an energy. This energy is high for two images representing two different people and low otherwise. A threshold on this energy has to be determined to make classification possible with the network. Thus, before any test, a script has to be written, determining a good threshold. This script should be written using pycaffe, the caffe model for python. Then, and only then can the tests be done.

Chapter 4

Results

This chapter describes the results obtained with this research. First, the deep neural network model used in the study is described. Then, the results obtained with this network in the context of face verification are given. Finally, we will describe the final experimental software which is available online and the results one can expect with it.

4.1 Preliminary results

Before the proposal, some parts of the solution have been built.

- Scripts to generate the database of face images from a video surveillance sequence have been written. The database has been generated and is usable for a direct face identification model. However, the overlap calculation for face detection is not written yet. Hence, the number of available classes is limited, decreasing the performances of a "Same/Not Same" network. The direct face identification model is not be affected by this issue, however it can be built with the current version of the database. The database contains 55,203 images. After manual labeling of the faces in the database, 183 images were labeled 1 for the first researcher, 325 were labeled 2, 15 were labeled 3.
- The scripts to generate the database files mentioned in Figure 3.2 are fully written, and work both for a direct face identification and for a Siamese network. They were executed for this second scenario. Four files "train1.txt", "train2.txt", "test1.txt", and "test2.txt", as described in section 3.6.2, were built.
- A Siamese model has been designed with Caffe and was trained on the generated database for 50,000 epochs. This training took a night, with batches of size 20, on the 780 GTX GPU card given by the laboratory. The accuracy of the resulting network could not be tested yet because of the singular output generated by such a network, as mentioned in section 2.3.1. A script is being written to face this issue.

4.2 Network used in this research

Wu, He and Sun published a light convolutional neural network for face verification (November 2015). Its particularity is that it is extremely light but still reaches state-of-the-art results.

4.2.1 Architecture

The following figure describes its architecture.

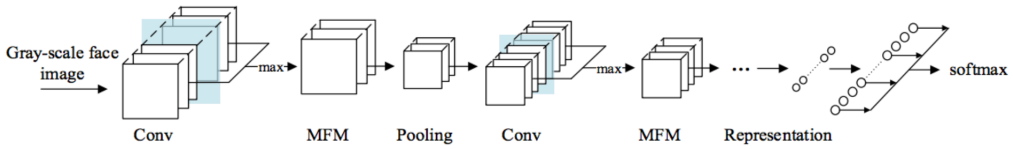


Figure 4.1:: Architecture of the lightened convolution network. Extracted from Wu, He, Sun, 2015.

First, as an input, a Gray-scale face image is provided. The article suggests the use of two possible models. The one used in this research is built with 4 convolution layer with Max-Feature-Map activation functions, 4 max-pooling layers and 2 fully connected layers. The details are given in the following figure.

A				B			
Name	Filter Size /Stride	Output Size	#param	Name	Filter Size /Stride, Pad	Output Size	#param
input	-	$144 \times 144 \times 1$	-	input	-	$144 \times 144 \times 1$	-
crop	-	$128 \times 128 \times 1$	-	crop	-	$128 \times 128 \times 1$	-
conv1_1	$9 \times 9/1$	$120 \times 120 \times 48$	3.8K	conv1_1	$5 \times 5/1, 2$	$128 \times 128 \times 48$	1.2K
conv1_2	$9 \times 9/1$	$120 \times 120 \times 48$	3.8K	conv1_2	$5 \times 5/1, 2$	$128 \times 128 \times 48$	1.2K
mfm1	-	$120 \times 120 \times 48$	-	mfm1	-	$128 \times 128 \times 48$	-
pool1	$2 \times 2/2$	$60 \times 60 \times 48$	-	pool1	$2 \times 2/2$	$64 \times 64 \times 48$	-
				conv2_a	$1 \times 1/1$	$64 \times 64 \times 48$	0.04K
conv2_1	$5 \times 5/1$	$56 \times 56 \times 96$	2.4K	conv2_1	$3 \times 3/1, 1$	$64 \times 64 \times 96$	0.8K
conv2_2	$5 \times 5/1$	$56 \times 56 \times 96$	2.4K	conv2_2	$3 \times 3/1, 1$	$64 \times 64 \times 96$	0.8K
mfm2	-	$56 \times 56 \times 96$	-	mfm2	-	$64 \times 64 \times 96$	-
pool2	$2 \times 2/2$	$28 \times 28 \times 96$	-	pool2	$2 \times 2/2$	$32 \times 32 \times 96$	-
				conv3_a	$1 \times 1/1$	$32 \times 32 \times 96$	0.09K
conv3_1	$5 \times 5/1$	$24 \times 24 \times 128$	3.2K	conv3_1	$3 \times 3/1, 1$	$32 \times 32 \times 192$	1.7K
conv3_2	$5 \times 5/1$	$24 \times 24 \times 128$	3.2K	conv3_2	$3 \times 3/1, 1$	$32 \times 32 \times 192$	1.7K
mfm3	-	$24 \times 24 \times 128$	-	mfm3	-	$32 \times 32 \times 192$	-
pool3	$2 \times 2/2$	$12 \times 12 \times 128$	-	pool3	$2 \times 2/2$	$16 \times 16 \times 192$	-
				conv4_a	$1 \times 1/1$	$16 \times 16 \times 192$	0.19K
conv4_1	$4 \times 4/1$	$9 \times 9 \times 192$	3K	conv4_1	$3 \times 3/1, 1$	$16 \times 16 \times 128$	1.1K
conv4_2	$4 \times 4/1$	$9 \times 9 \times 192$	3K	conv4_2	$3 \times 3/1, 1$	$16 \times 16 \times 128$	1.1K
mfm4	-	$9 \times 9 \times 192$	-	mfm4	-	$16 \times 16 \times 128$	-
pool4	$2 \times 2/2$	$5 \times 5 \times 192$	-	pool4	$2 \times 2/2$	$8 \times 8 \times 128$	-
				conv5_a	$1 \times 1/1$	$8 \times 8 \times 128$	0.12K
				conv5_1	$3 \times 3/1, 1$	$8 \times 8 \times 128$	1.1K
				conv5_2	$3 \times 3/1, 1$	$8 \times 8 \times 128$	1.1K
				mfm5	-	$8 \times 8 \times 128$	-
				pool5	$2 \times 2/2$	$4 \times 4 \times 128$	-
fc1	-	256	1,228K	fc1	-	256	524K
fc2	-	10,575	2,707K	fc2	-	10,575	2,707K
loss	-	10,575	-	loss	-	10,575	-
total			3,961K				3,244K

Figure 4.2:: Details on the architecture of the lightened convolution networks. Extracted from Wu, He, Sun, 2015.

The MFM function works as explained in the following figure.

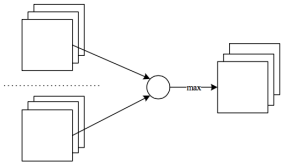


Figure 4.3:: The MFM activation function on a convolution layer. Extracted from Wu, He, Sun, 2015.

Given a convolution layer $C \in R^{hw2n}$, the MFM can be written :

$$f_{ij}^{k} = \max_{1 \leq k \leq n} (C_{ij}^{k}, C_{ij}^{k+n})$$

$2n$ being the channel of the input convolution layer, $1 \leq i \leq h$ and $1 \leq j \leq w$.

4.2.2 Results

As said previously, these networks reach state-of-the-art results. The two networks were trained on the LFW database and tested both on LFW and YTF (YouTube Faces Database). The results are given in the next two tables, compared with other state-of-the-art methods.

Method	#Net	Accuracy	Protocol
DeepFace [24]	1	91.40%	supervised
WebFace [27]	1	88.00%	unsupervised
WebFace+PCA [27]	1	90.60%	unsupervised
VGG [16]	1	92.80%	unsupervised
Our model A	1	90.72%	unsupervised
Our model B	1	91.60%	unsupervised

Figure 4.4:: Comparison with other state-of-the-art methods on YTF. Extracted from Wu, He, Sun, 2015.

Method	#Net	Accuracy	TPR@FAR=0.1%	Protocol	Rank-1	DIR@FAR=1%
DeepFace [24]	1	95.92%	-	unsupervised	-	-
DeepFace [24]	7	97.35%	-	unrestricted	-	-
Web-Scale [25]	1	98.00%	-	unrestricted	82.1%	59.2%
Web-Scale [25]	4	98.37%	-	unrestricted	82.5%	61.9%
DeepID2 [19]	1	95.43%	-	unsupervised	-	-
DeepID2 [19]	4	97.75%	-	unsupervised	-	-
DeepID2 [19]	25	98.97%	-	unsupervised	-	-
WebFace [27]	1	96.13%	-	unsupervised	-	-
WebFace+PCA [27]	1	96.30%	-	unsupervised	-	-
WebFace+Joint Bayes [27]	1	97.30%	-	unsupervised	-	-
WebFace+Joint Bayes [27]	1	97.73%	80.26%	unrestricted	-	-
FaceNet [17]	-	99.63%	-	unrestricted	-	-
VGG [16]	1	97.27%	81.90%	unsupervised	74.10%	52.01%
Our model A	1	97.77%	84.37%	unsupervised	84.79%	63.09%
Our model B	1	**98.13%**	**87.13%**	unsupervised	**89.21%**	**69.46%**

Figure 4.5:: Comparison with other state-of-the-art methods on LFW. Extracted from Wu, He, Sun, 2015.