IDIAP COMMUNICATION

Multimodal Identity Verification at IDIAP

Christine Marcel

IDIAP–Com 03-04

August 2003

Institut Dalle Molle d'Intelligence Artificielle Perceptive • CP 592 • Martigny • Valais • Suisse

téléphone +41 – 27 – 721 77 11
télécopieur +41 – 27 – 721 77 12
adr.él. secretariat@idiap.ch
internet http://www.idiap.ch

1 Introduction

In the context of the european project BANCA on Biometric Access Control for Network and e-Commerce Applications, we worked on multimodal identity verification. Identity verification is a general task that has many real-life applications such as access control or transaction authentication. The goal of such system is to recognize if a given person is or not who he claims to be. The system can be based on biometrics characteristics of a person such as, the voice, the face, or the fingerprint depending on the application. In our work, we only took into account two modalities : the voice and the face. The system can take a decision using only the face, or only the voice, but it can also use both face and voice to hope for a better decision. The goal of our work on multimodal identity verification is to merge several modalities (using so-called "fusion" methods), here face and voice, and try to obtain better performance than using only one of them.

Even if fusion already shown great improvements, we also integrated confidence measures during the fusion process, using different methods that will be described later. These methods can be used either to enhance the performance of a multimodal fusion algorithm or to obtain a confidence level on the decisions taken by the system.

Sometimes, authentication systems are too big for given mobile application, thus we need smaller systems. That is why we also worked on reducing the number of parameters of certain systems and we compared the performance of the baseline system (the best system found according to our evaluation) to a smaller system. The goal is to find a more compact system which has similar performance to the baseline system.

This document presents an overview on multimodal identity verification in the context of the BANCA project. In the first part, we detail multimodal identity verification and the methods used to create a system and to compute its performance. In the second part, we briefly describe the databases we worked with (XM2VTS and BANCA). Afterwards, methods and algorithms are described. Finally, we give some results. The aim of this document is to describe all this work in order to be reused and to be carried on by anyone else.

2 Introduction to Multimodal Identity Verification

2.1 General Description

An automatic identity verification system has to verify whether a given person is or not who he claims to be. Our systems are based on biometrics characteristics of a person ; mainly face and voice. Indeed, numerous researchers have worked on the possibility to give to the system more than one characteristic to take the decision. We call that multimodal identity verification and it consists of merging the results of many modalities in order to take better decisions. That is why, in order to do fusion, we need the scores obtained by each modalities and for each access of a person. These scores are the results of many algorithms. For voice, we mainly used the scores obtained by training a Gaussian Mixture Model (GMM) using the EM algorithm in order to maximize the likelihood of the data. For face, numerous algorithms were used (MLP, LDA, ...). The corresponding obtained scores are given to a system which takes all them as input information and gives a new score for each access. This last score is the decision of the fusion system.

We have seen that the goal of an identity verification system is to accept or reject the identity claim made by a given person. If the person is who he claims to be, he is called a client, otherwise he is called an impostor.

Databases are used to train a system and then evaluate it. These databases will be described later in this document. For each database, one part of the dataset is used to train a fusion model, and a different part is used to evaluate it.

In the following, we will describe the procedure used to create a model and then evaluate it.

2.2 Training a Fusion Model

In the training part, the goal is to compute a fusion model using the training part of the dataset. The algorithm takes as input, the two scores of face and voice, creates a fusion model and gives in output a fusion score. We have to select one or more *hyper-parameters* in order to obtain an optimal performance. These hyper-parameters often control the *capacity* [9] of the model, which is related to the size of the function space spanned by the model. These hyper-parameters should be selected according to the number of training examples and the complexity of the training problem. For MLPs, (we will see later that we mainly used this algorithm), this amounts to selecting the number of hidden units and/or the number of training iterations.

Once we have a trained model, we want to evaluate it. In the following, we describe the evaluation part.

2.3 System Evaluation

The system may make two types of errors : false acceptance (FA) when the system accepts an impostor, and false rejection (FR) when the system rejects a client. We then use these two errors to compute the performance of the system. This performance is often measured as follows :

$$FAR = \frac{\text{number of FAs}}{\text{number of impostor accesses}},$$

$$FRR = \frac{\text{number of FRs}}{\text{number of client accesses}}.$$

We generally used a combination of these two measures called Half Total Error Rate (HTER) :

$$HTER = \frac{FAR + FRR}{2}.$$

In order to tune the system, a threshold is used to obtain for example a small FAR, or a small FRR depending on the application. We generally use the threshold such that FAR=FRR, called Equal Error Rate (EER), on a validation set. This threshold is then used on a test set to compute the HTER.

We also use some curves to plot the performance. For example, we use the DET curve (Figure 1), and a graph which represents client and impostor scores as well as the separation obtained by the face, voice and fusion systems.

2.3.1 DET Curve

The curve Receiver Operating Characteristic (ROC) represents the FAR as a function of the FRR. More recently, other researchers proposed the DET curve (Figure 1) a non-linear transformation of the ROC curve in order to make results easier to compare.
To plot such a curve, we used a tool, called plotdet :

`Cf. http://www.idiap.ch/~marietho`

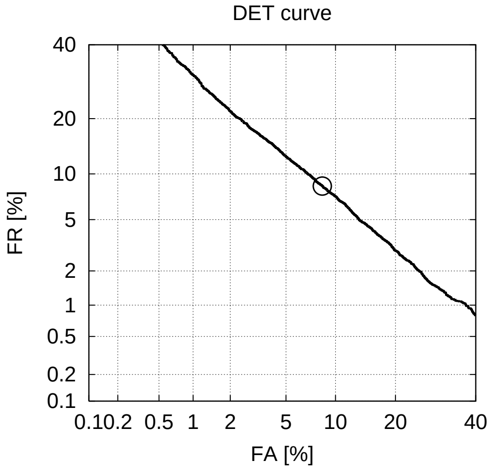

Fig. 1 – DET example.

2.3.2 Separation Scores

Sometimes it can be interesting to see how a system separates the scores. We have a way to create such a graph. This graph represents the voice scores as the x axis and the face score as the y axis. We plot client scores and impostor scores separately. Then, we plot the lines corresponding to the thresholds found with the face system and the voice system. We have implemented an algorithm, see B.4, which can give the separation found by the fusion system. An example of the graph is given in Figure 2.

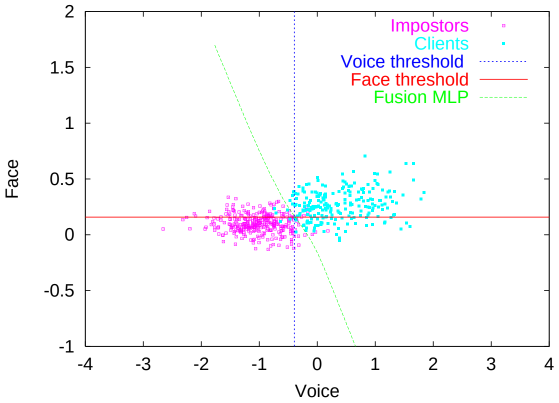

Fig. 2 – Example of separation scores.

2.4 Selection Procedure

As there is only one training set available in general, the same data should be used to select the hyper-parameters of the models, to train them, and finally to select a decision threshold that optimizes a given criterion. In order to do all this with the same dataset without the risk of optimistic bias, a cross-validation method had to be used. The method generally used is the following :

1. For each value of the hyper-parameter of the fusion model,
   (a) perform a K-fold cross-validation in order to obtain unbiased scores for each train examples (in our case, $K = 5$, hence we train a model with the first $\frac{4}{5}$ of the dataset, and save the

scores of the last $\frac{1}{5}$, then we do the same for each of the 4 other partitions $\{\frac{4}{5}, \frac{1}{5}\}$, in order to finally obtain scores for the whole dataset that were taken by a model that was not trained on the corresponding examples);

(b) after a random shuffling of the data, perform a K-fold cross-validation ($K = 5$) on the obtained scores in order to compute for each partition the HTER corresponding to the threshold that optimized the EER on the other partitions;

(c) the performance of the hyper-parameter corresponds to the average of these HTER, which are unbiased.

2. Select the value of the hyper-parameter that has the best average HTER performance.
3. Using the unbiased scores corresponding to the best model, select the threshold that optimizes EER on the whole training set.
4. Train the best model over the whole training set.
5. Apply the best model (found is step 4) on the test set and use the best threshold (found in step 3) to take the decisions.

3 Databases

We worked mainly with two databases : XM2VTS and BANCA.

3.1 XM2VTS

The database contains four recordings sessions of 295 subjects taken at one month intervals. On each sessions, two recordings were made, each consisting of a speech shot and head rotation shot. A protocol has been defined to evaluate the performance of vision and speech based person authentication systems on the XM2VTS database.

The database was divided into three sets : training set, evaluation set, and test set. The training set was used to build client models, while the evaluation set was used to compute the decision. Finally the test set was used only to estimate the performance of different verification algorithms.

The database was randomly divided into 200 clients, 25 evaluation impostors, and 70 test impostors. Two different evaluation configurations were defined. They differ in the distribution of client training and client evaluation data. Both the client training and client evaluation data were drawn from the same recording sessions for Configuration I which might lead to optimistic performance on the evaluation set. For Configuration II on the other hand, the client evaluation and client test sets are drawn from different recording sessions which might lead to more realistic results.

The following numbers of subjects were used :
- Clients :200
- Impostors-Evaluation :25
- Impostors-Test :70

This led to the following statistics :
- client training examples : ConfI :3 per client, ConfII :4 per client
- evaluation - clients : ConfI - 600, ConfII - 400
- evaluation - impostors : 40'000 (25*8*200)
- test client accesses : 400 (200*2)
- test impostor accesses : 112'000 (70*8*200)

For more details, see [3].

3.2 BANCA

The BANCA database contains video and speech data recorded for 5 different languages[1], on 260 subjects (52*5, 26 males and 26 females), during 12 sessions separated into 3 different scenarios : controlled, degraded and adverse. Each language - and gender - specific population was itself subdivided into 2 groups of 13 subjects (denoted $g1$ and $g2$).

For each language, an additional set of 30 other subjects, 15 males and 15 females, recorded one session (audio and video). This set of data is referred to as *world data*.

We then consider 7 distinct training-test configurations, depending on the actual conditions corresponding to the training and to the testing conditions.

- Matched controlled (Mc) :
  - client training from 1 controlled session
  - client and impostor testing from the other controlled sessions (within the same group)
- Matched degraded (Md) :
  - client training from 1 degraded session
  - client and impostor testing from the other degraded sessions (within the same group)
- Matched adverse (Ma) :
  - client training from 1 adverse session
  - client and impostor testing from the other adverse sessions (within the same group)
- Unmatched degraded (Ud) :
  - client training from 1 controlled session
  - client and impostor testing from degraded sessions (within the same group)
- Unmatched adverse (Ua) :
  - client training from 1 controlled session
  - client and impostor testing from adverse sessions (within the same group)
- Pooled test (P) :
  - client training from 1 controlled session
  - client and impostor testing from all conditions sessions (within the same group)

  Note that the scores necessary for this experiment can be obtained directly from experiments Mc, Ud and Ua.
- Grand test (G) :
  - client training from 1 controlled, 1 degraded and 1 adverse sessions
  - client and impostor testing from all conditions sessions (within the same group)

Let us note that :

1. We can define protocols P and G as *primary protocols, and the others as secondary protocols.*
2. *In P, the client training has already been performed during protocols M or U.*
3. *In G, for client training, there is a solution which does not need new computation.*

To have a complete description of the BANCA database see [1].

4 Methods and Algorithms Developed

4.1 Algorithms Used for Fusion

We tried many fusion algorithms such as Support Vector Machines and Multi-Layer Perceptrons.

[1] English, French, German, Italian and Spanish

4.1.1 Support Vector Machines

Support Vector Machines (SVMs) [9] have been applied to many classification problems, generally yielding good performance compared to other algorithms. The resulting function is of the form

$$\hat{y} = sign\left(\sum_{i=1}^{l} y_i \alpha_i K(\mathbf{x}, \mathbf{x}_i) + b\right) \tag{1}$$

where $\mathbf{x}$ is the input vector of a example to test, $\hat{y}$ is the decision of the model (accept if $\hat{y}$ is positive, reject otherwise), $\mathbf{x}_i$ is the input vector of the i^{th} training example, l is the number of training examples, the α_i and b are the parameters of the model, and $K(\mathbf{x}, \mathbf{x}_i)$ is a kernel function that can have different forms, such as :

$$K(\mathbf{x}, \mathbf{x}_i) = (\mathbf{x}^t\ \mathbf{x}_i + 1)^d \tag{2}$$

which leads to a Polynomial SVM with parameter d, or

$$K(\mathbf{x}, \mathbf{x}_i) = \exp\left(\frac{-\|\mathbf{x} - \mathbf{x}_i\|^2}{\sigma^2}\right) \tag{3}$$

which leads to a Radial Basis Function (RBF) SVM with parameter σ. Either d or σ must be selected using methods such as cross-validation.

In order to train such SVMs, one needs to solve the following quadratic optimization problem : find the parameter vector

$\alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_l\}$

that maximize the objective function

$$Q(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \tag{4}$$

subject to the constraints

$$\sum_{i=1}^{l} \alpha_i y_i = 0 \tag{5}$$

and

$$0 \le \alpha_i \le C\ \forall i. \tag{6}$$

It is important to note that the training complexity of SVMs is quadratic on the number l of examples, which makes the use of SVMs for large datasets difficult.Note however that in the resulting solution (1), most α_i are equal to 0, and the examples with non-zero α_i are called *support vectors*.

4.1.2 Multi-Layer Perceptrons

A multi-layer perceptron (MLP) is a particular architecture of artificial neural networks [4, 6], composed of layers of non-linear but differentiable parametric functions. The best MLP architecture we found for BANCA was a 1-hidden-layer MLP.

For instance, the output $\hat{y}$ of a 1-hidden-layer MLP can be written mathematically as follows

$$\hat{y} = b + \mathbf{w} \cdot \tanh\left(\mathbf{a} + \mathbf{x} \cdot \mathbf{V}\right) \tag{7}$$

or :

$$\hat{y} = \text{sigmoid}\left(b + \mathbf{w} \cdot \tanh\left(\mathbf{a} + \mathbf{x} \cdot \mathbf{V}\right)\right) \tag{8}$$

where

$$\text{sigmoid}(x) = \frac{1}{1 + \exp(-x)} \tag{9}$$

where the estimated output $\hat{y}$ is a function of the input vector $\mathbf{x}$, and the parameters are $\{b, \mathbf{w}, \mathbf{a}, \mathbf{V}\}$. In this notation, the non-linear function tanh() returns a vector which size is equal to the number of hidden units of the MLP, which controls its capacity and should thus be chosen carefully, by cross-validation for instance.

An MLP can be trained by gradient descent using the backpropagation algorithm [7] to optimize any derivable criterion, such as the *mean squared error* (MSE) :

$$\text{MSE} = \frac{1}{l} \sum_{i=1}^{l} (y_i - \hat{y}_i)^2 \ . \tag{10}$$

We also tried another criterion based on the so-called *decision cost function* (DCF) and was implemented as follows :

$$\begin{aligned} \text{DCF} \quad = \quad & \sum_{i:y_i=1} \frac{(1-\hat{y}_i) \cdot C_{\text{F R}} \cdot P(\text{client})}{\text{number of clients}} + \\ & \sum_{i:y_i=0} \frac{\hat{y}_i \cdot C_{\text{FA}} \cdot P(\text{impostor})}{\text{number of impostors}} \end{aligned} \tag{11}$$

where C_{FR} is the cost of a false rejection, C_{FA} is the cost of a false acceptance, $P(\text{client})$ is the prior probability of a client access, $P(\text{impostor})$ is the prior probability of an impostor access. These variables have been set in the experiments to the following values :

- $P(\text{client}) = 0.5$
- $P(\text{impostor}) = 0.5$
- $C_{\text{FR}} = 1.$
- C_{FA} varied in the set (0.5, 1, 5.) and was chosen by cross-validation.

Some results on the XM2VTS database can be found in [3]. Results for the BANCA database can be found in [5].

4.2 Confidence Measures : Methods

We tried to improve the performance of fusion by integrating confidence measures during the fusion process. We implemented different methods, for example using Gaussian hypothesis or non parametric estimation.

A paper [2] describes completely three of the methods we tried : Gaussian hypothesis, non-parametric, and model adequacy. In the following, we will briefly describe these methods, and we will also describe another method, called *distance method*.

4.2.1 Gaussian Hypothesis

We suppose that all the scores s from clients have been generated by the same Gaussian distribution $\mathcal{N}(s; \mu_c, \sigma_c)$, and all the scores from impostors have been generated by another Gaussian distribution $\mathcal{N}(s; \mu_i, \sigma_i)$. Then, a good measure for a given score could be related to the distance between the probability that the score is from a client and the probability that the score is from an impostor. For instance, we can compute the following measure :

$$m_{gauss}(s) = |\mathcal{N}(s;\mu_c,\sigma_c) - \mathcal{N}(s;\mu_i,\sigma_i)| \ . \tag{12}$$

4.2.2 Non-parametric estimation

We use a training set of scores. We partition the space of scores into K distinct subspaces. The partition process could be uniform over the score space (each partition has the same size) or over the score distribution (each partition contains the same number of training scores). In both cases, we simply compute for each subspace SS_i the statistic of interest. In our case, we computed the number of errors that were made in the subspace, divided by the total number of scores in the subspace.

$$m_{np}(SS_i) = \frac{\text{number of FAs } + \text{number of FRs in } SS_i}{\text{number of accesses in } SS_i} \ . \tag{13}$$

Turning to the test set, when we want to compute the confidence of a given score s, we find the subspace SS_i corresponding to the given score and return the associated value $m_{np}(SS_i(s))$.

4.2.3 Model Adequacy

Taking into account that most unimodal verification systems (at least those we are indeed using) are based on some kind of gradient method optimizing a given criterion, we propose to compute the gradient of a simple measure of confidence of the decision of the model given an access with respect to every parameter in the model. The average amplitude of such gradient gives an idea of the *adequacy* of the parameters to explain the confidence of the model on the access. Hence, a global measure m_a could be

$$m_{ma}(\mathbf{X}) = \frac{1}{M}\sum_{i=1}^{M}\left|\frac{\partial f(\mathbf{X})}{\partial \theta_i}\right| \tag{14}$$

where θ_i is one the M parameters of the model and $f(\mathbf{X})$ is a simple measure of confidence of the model given access $\mathbf{X}$.

For our speaker verification system based on GMMs, we chose

$$m_{ma}(\mathbf{X}) = \frac{1}{M_c}\sum_{i=1}^{M_c}\left|\frac{\partial \log p(\mathbf{X}|S_c)}{\partial \theta_i}\right| + \frac{1}{M_w}\sum_{i=1}^{M_w}\left|\frac{\partial \log p(\mathbf{X}|\Omega)}{\partial \theta_i}\right| \tag{15}$$

where M_c is the number of parameters in the claimed client model S_c and M_w is the number of parameters in the world model Ω.

Note that in this case, the measure is strongly related to the average value of the *Fisher score*, which is a sufficient statistics of the generative model.

For our face verification system based on MLPs, the best measure we found was

$$m_{ma}(\mathbf{X}) = \frac{1}{M}\sum_{i=1}^{M}\left|\frac{\partial MLP(\mathbf{X})^2}{\partial \theta_i}\right| \tag{16}$$

where M is the number of parameters of the model and $MLP(\mathbf{X})$ is the output obtained on access $\mathbf{X}$. Considering the fact that the targets of the MLP where -1 and 1, the higher the absolute value of the score was, the more confident the system was, and we thus measured the overall influence of the parameters to obtain high scores (and confidence).

4.2.4 Distance

Let us take an access X (client or impostor) claiming to be the client j. The score is then denoted $S_j(X)$. Suppose that X claims also to be any of the $N-1$ another clients of the database, and let us note $S_i(X)$ ($i = 1,..,N$, $i \neq j$) the corresponding scores. The aim is then to compute the average distance between $S_j(x)$ and all $S_i(x)$.

$$m_d(\mathbf{X}) = \frac{1}{N-1} \sum_{i=1, i\neq j}^{N} |S_j(X) - S_i(X)| \tag{17}$$

The hope is that this distance should be smaller, when X is the client he claims to be than when he is an impostor.

In [2], we give some results comparing these confidence methods on the XM2VTS database. A description of the fusion algorithms is also given.

Confidence measures were also tested for the BANCA database (for more details see [8]).

4.3 Scalability

Sometimes, the devices to use are very small and our verification systems may be too big. That is why we focus on the problem of scalability. A paper accepted to the AVBPA conference [5] describes these systems, and gives the results. A lot of score files and results can be found in :
ftp ://ftp.idiap.ch/pub/bengio/banca/banca_scores/html/banca_experiments/banca_experiments.html.

5 Experimental Results

5.1 Results on XM2VTS

In Table 1, we summarize the results obtained in voice, face, fusion and confidence on XM2VTS in 2002. The results presented in [2], had be obtained in 2001. In 2002, the method Gaussian hypothesis was not kept because of the less interesting results.

Modalities	Configuration I			Configuration II		
	FAR	FRR	HTER	FAR	FRR	HTER
Face	1.75	2.00	1.88	1.46	2.25	1.86
Voice	1.47	1.00	1.23	1.30	1.25	1.28
Fusion (MLPs)	0.32	0.75	0.53	0.13	0.25	0.19
Non-parametric (MLPs)	0.20	0.75	0.48	0.05	0.25	0.15
Model Adequacy (MLPs)	0.36	0.75	0.56	0.13	0.25	0.19
Distance (MLPs)	0.18	0.50	0.34	0.04	0.25	0.15

TAB. 1 – Performance on the test set of different verification systems

5.2 Results on BANCA

We give all the results for the baseline systems obtained with the BANCA database in Table 2.

In Table 3, we present the results obtained in the scalability study for voice on protocol P.

In Table 4, we present the results obtained in the scalability study for voice on protocol G.

In Table 5, we present the results obtained in the scalability study for face on both protocol P and G.

Protocols	Voice	Face auto	Face man	Fusion auto	Fusion man
G	1.15	10.66	3.50	0.37	0.27
P	4.70	23.05	17.20	3.90	2.94
Ma	4.01	8.73	9.86	1.44	1.12
Mc	0.56	15.76	5.29	0.96	1.20
Md	3.93	15.46	8.09	2.08	2.24
Ua	9.70	29.09	26.36	8.01	6.01
Ud	1.92	25.00	15.38	1.12	2.72

Tab. 2 – Baseline system : HTER for all the protocols for voice, face and fusion

Number of parameters	HTER	LFCC	Number of voice Gaussians
510	15.49	12	10
670	13.14	16	10
1675	9.05	16	25
3350	6.62	16	50
5025	5.90	16	75
6700	5.82	16	100
13400	**4.70**	**16**	**200**
15300	4.99	12	300
20100	5.82	16	300

Tab. 3 – Voice, protocol P : the best methods wrt the number of parameters

Number of parameters	HTER	LFCC	Number of voice Gaussians
510	7.05	12	10
670	5.85	16	10
1675	2.96	16	25
3350	1.95	16	50
5025	1.71	16	75
6700	1.31	16	100
10200	**1.10**	**12**	**200**
15300	1.26	12	300

Tab. 4 – Voice, protocol G : the best methods wrt the number of parameters

Number of parameters	HTER (P)	HTER (G)
25	23.69	13.76
50	20.51	11.14
100	19.02	8.68
200	17.49	7.72

Tab. 5 – Face, protocol P and G : HTER wrt the number of parameters

Number of parameters	HTER	LFCC	Number of voice Gaussians	Number of face parameters	Number of fusion hidden units
756	10.44	16	10	25	15
761	10.07	16	10	50	10
851	9.05	16	10	100	20
1386	7.26	12	25	50	15
1766	6.22	16	25	50	10
3436	5.72	16	50	25	20
3511	5.48	16	50	100	15
3601	5.21	16	50	200	12.50
5181	4.49	12	100	50	7.50
6831	4.27	16	100	100	7.50
10286	3.93	12	200	25	15
10371	3.87	12	200	100	17.50
10441	**3.79**	**12**	**200**	**200**	**10**
13521	3.53	16	200	50	17.50
13661	**3.12**	**16**	**200**	**200**	**15**

Tab. 6 – Fusion, protocol P : the best methods wrt the number of parameters

In Table 6, we present the results obtained in the scalability study for fusion on protocol P. In Table 7, we present the results obtained in the scalability study for fusion on protocol G

Number of parameters	HTER	LFCC	Number of voice Gaussians	Number of face parameters	Number of fusion hidden units
616	2.70	12	10	25	20
631	2.56	12	10	50	17.50
681	2.03	12	10	100	17.50
751	1.58	12	10	200	10
1371	1.47	12	25	25	17.50
1416	1.20	12	25	100	10
1826	1.01	16	25	100	12.50
3501	0.72	16	50	100	12.50
3591	0.53	16	50	200	10
5266	0.24	16	75	200	10
6941	0.24	16	100	200	10
10441	0.53	12	200	200	10
13641	**0.24**	**16**	**200**	**200**	**10**
15541	**0.43**	**12**	**300**	**200**	**10**
20341	0.37	16	300	200	10

Tab. 7 – Fusion, protocol G : the best methods wrt the number of parameters

6 Conclusion

In this document, we have presented an overview of the work did at IDIAP in multimodal identity verification in the context of the BANCA european project. The algorithms used, and the methods have been presented, as well as some results on the XM2VTS and BANCA databases.

Références

[1] S. Bengio, F. Bimbot, J. Mariéthoz, and V. Experimental protocol on the BANCA database. Technical Report IDIAP-RR 02-05, IDIAP, 2002.

[2] S. Bengio, C. Marcel, S. Marcel, and J. Mariethoz. Confidence measures for multimodal identity verification. *Information Fusion*, pages 267–276, 2002.

[3] S. Bengio, J. Marithoz, and S. Marcel. Evaluation of biometric technology on XM2VTS. Technical Report IDIAP-RR 01-21, IDIAP, 2001.

[4] C. Bishop. *Neural Networks for Pattern Recognition.* Clarendon Press, Oxford, 1995.

[5] J. Czyz, S. Bengio, C. Marcel, and L. Vanderdope. Scalability analysis of audio-visual person identity verification. *4th International Conference, AVBPA, LNCS 2688*, pages 752–760, 2003.

[6] S. Haykin. *Neural Networks, a Comprehensive Foundation, second edition.* Prentice Hall, 1999.

[7] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In D. E. Rumelhart and James L. McClelland, editors, *Parallel Distributed Processing*, volume 1. MIT Press, Cambridge, MA., 1986.

[8] S. Sarangi. Enhanced performance of multimodal biometric systems by confidence estimation. Diploma thesis, EPFL, 2003.

[9] V. Vapnik. *The Nature of Statistical Learning Theory.* Springer-Verlag, New York, 1995.

7 Acknowledgment

The author wants to thank the Swiss National Science Foundation for supporting this work through the National Center of Competence in Research (NCCR) on "Interactive Multimodal Information Management (IM2)". This work was also funded by the European project "BANCA", through the Swiss Federal Office for Education and Science (OFES), project number 99-0563-1. The author thanks also S. Bengio, J. Mariéthoz, S. Marcel and R. Collobert for fruitful discussions and collaboration.

A Databases, Experiments and Results

A.1 XM2VTS

The experiments done with XM2VTS cannot be found anymore in the home directory. They have been archived and some CDs were created. These CDs contain the data used, the experiments done, the Torch modules used to perform the experiments, some scripts and the results.

A.2 BANCA

A.2.1 Directories and Files

All the data, results, scripts for BANCA are in /home/learning/alves/banca. In order to create the data in fusion using the face and voice data, a script was implemented in perl, see B.5.

The hierarchy of the directory is as follows :

- data : This directory contains the data used in the context of BANCA for all the modalities.
  - voice
    - [scale_]institutName_algorithm[_version]
  - face
    - [scale_]institutName_algorithm[_version]
  - fusion
    - [scale_]institutName_face_algorithm_institutName_voice_algorithm[_version]
- results : This directory contains all the results obtained in terms of performance of the systems. These results are generally summarized and discussed in many papers we referred to in the document. The names of the directories are associated to the name of the directories in data.
  - voice
    - [scale_]institutName_results_algorithm[_version]
  - face
    - [scale_]institutName_algorithm
  - fusion
    - [scale_]institutName_face_algorithm_institutName_voice_algorithm[_version]
- fusion_prototype : In order to be used in real life, we have created a fusion model to do field tests. This directory contains all the data, results, and scripts to create the models for the prototype. A model can be created using the program mlp_train (TorchII) or mlp in mode train (TorchIII). The script script_example_mlp_torch2 gives an example of script for mlp in TorchII and script_example_mlp_torch3 in TorchIII. The same kind of script can be used for svm or another machine. These scripts can also be used to do all the process of fusion we described in the selection procedure part, see B.5.
  - data
    - institutName_algorithm[_version]_institutName_algorithm[_version]
  - results
    - model_fusion_institutName_institutName
  - scripts
    - script_example_mlp_torch(n)

where :

- scale : option. The name of the directory begins or not by scale.
- version : option. Sometimes there is no version, and sometimes there is one. In this latter case, version can take the values : v1, v2, ...
- institutName : The values are : idiap, ucl, uc3m, surrey.

– algorithm : mlp, gmm, nc, svm, ...

A.3 Scalability

A.3.1 Speech Scalability

All the experiments in Speech for scalability are in :
/home/learning/alves/scale_asv_banca/BANCA/Experiments. In expe_8k_y*2 are all the experiments where the number of features are (y*2 + 1).
The data are in :
/home/learning/alves/banca/data/speech/scale_institut_model[_version]. The results are in :
/home/learning/alves/banca/results/speech/scale_institut_model[_version].

A.3.2 Face Scalability

Face scalability was not done at IDIAP (Cf Surrey, UCL, etc...).
The data are in :
/home/learning/alves/banca/data/face/scale_institut_model[_version].
The results are in :
/home/learning/alves/banca/results/face/scale_institut_model[_version].

A.3.3 Fusion

The data are in :
/home/learning/alves/banca/data/fusion/scale_institut_model_institut_model[_version].
The results are in :
/home/learning/alves/banca/results/fusion/scale_institut_model_institut_model[_version].

B Algorithms

The algorithms used were implemented with the Torch library. In order to do fusion, we have implemented many classes, and put them in the directory
/home/learning/alves/Torch/dev in TorchII and
/home/learning/alves/Torch3/dev_christine in TorchIII. These classes are as follow :

```
IOBanca: This class is used to read Banca datasets format:
<id> <claimed_id> <access_file_name> <score1> ... <scoreN>

BancaDataSet: Format used in fusion. Cf IOBanca.

BancaOutputMeasurer: This class can be used to save the
outputs of a Trainer in a file in such a format that it
can be read again in Torch using the BancaDataSet class.

ThresholdTrainer: This class gives a trainer for the thresholds.
In train, this trainer computes the best thresholds
(optimizing either eer, hter, zfa, zfr, dcf=frr+x*far, or
wer=(frr+G*far)/(1+G) for a given x).
In test, it computes hter, eer, zfa, zfr , dcf, wer,
with every best thresholds found in train.
It supposes that the output class 0 = impostor and
the output class 1 = client.
```

```
ThresholdMachine: machine which can be trained with
a ThresholdTrainer. Threshold machines take in
inputs sequences which have always the same frame size,
given by n_inputs, and outputs sequences which have always
the same frame size too, given by n_outputs.

ThresholdMeasurer: prints all the performance in a table.

ClassKFold: Kfold for classification problems that keeps the class prior
distribution constant accross all folds.

BancaKFold: Kfold for banca classification problems that ensures not to
have the same client in two different partitions.

NPConfidenceMeasurer: This class gives a measurer to compute the
non-parametric confidence measure. This measurer compute a ratio
for all the intervals on data test. Then it compares the results
with the ratio computed by the trainer on train and computes an
average of the difference. It gives us an average confidence.

NPConfidenceTrainer: This class gives a trainer to compute a confidence.
This trainer compute a ratio on data train.

NPConfidenceMachine: This class gives a machine for the NPConfidenceTrainer.

DCFCriterion: Compute the weighted classification error, taking into account
false positives and false negatives separately.
```

We worked with MLPs and SVMs, but mainly with MLPs.

B.1 MLP code

The binaries to train and test a MLP can be found in /home/learning/alves/Torch/fusion for TorchII.

1. mlp_kfold

```
This program will train a MLP with tanh outputs for classification
using k-fold cross-validation and saving out-of-samples outputs in a file.

usage: Linux_OPT_FLOAT/mlp_kfold [options] data_file output_file

Arguments:
   - data_file: the training data, string.
   - output_file: the file to save the out-of-samples outputs, string.

Model Options:
   - nhu: number of hidden units [25], integer.
   - raw: use raw data (do not normalize)
   - kfold: the number of partitions in K-fold [5], integer.
   - pt: positive target [1], real.
   - nt: negative target [-1], real.
```

```
Learning Options:
   - iter: max number of iterations [25], integer.
   - lr: learning rate [0.01], real.
   - e: end accuracy [1e-05], real.
   - lrd: learning rate decay [0], real.

Misc Options:
   - seed: the random seed [-1], integer.
   - load: max number of examples to load[-1],integer.
```

2. mlp_train

```
This program will train a MLP with tanh outputs
and save the resulting model to a file.

usage: Linux_OPT_FLOAT/mlp_train [options] data_file model_file

Arguments:
   - data_file: the training data, string.
   - model_file: the file to save the model, string.

Model Options:
   - nhu: number of hidden units [25], integer.
   - raw: use raw data (do not normalize)
   - pt: positive target [1], real.
   - nt: negative target [-1], real.

Learning Options:
   - iter: max number of iterations [25], integer.
   - lr: learning rate [0.01], real.
   - e: end accuracy [1e-05], real.
   - lrd: learning rate decay [0], real.

Misc Options:
   - seed: the random seed [-1], integer.
   - load: max number of examples to load[-1], integer.
```

3. mlp_test

```
This program will test an MLP with tanh outputs
and save the outputs to a file.

usage: Linux_OPT_FLOAT/mlp_test [options] data_file model_file output_file

Arguments:
   - data_file: the testing data, string.
   - model_file: the file to load the model, string.
   - output_file: the file to save the outputs, string.

Model Options:
   - nhu: number of hidden units [25], integer.
   - raw: use raw data (do not normalize)
```

```
      - pt: positive target [1], real.
      - nt: negative target [-1], real.

   Misc Options:
      - seed: the random seed [-1], integer.
      - load: max number of examples to load [-1], integer.
```

The option nhu is the number of hidden units. This value can be tuned in order to find the best model. The option raw precises if the data has to be normalized or not. The options nt and pt are the targets. They can be set to -1 and 1 respectively, or 0 and 1, depending on the outputs of the MLP.

In TorchIII, these three programs are merged into only one : mlp. It can be found in : /home/learning/alves/Torch3/fusion.

```
This program will train a MLP with log-softmax outputs for
classification and linear outputs for regression.

usage: Linux_OPT_FLOAT/mlp [options] file

The argument file is the train file, string.

Model Options:
   - nhu: number of hidden units [25].
   - nt: negative target  [-1].
   - pt: positive target [1].

Learning Options:
   - iter: max number of iterations [25], integer.
   - lr: learning rate [0.01], real.
   - e: end accuracy [1e-05], real.
   - lrd: learning rate decay [0], real.
   - kfold: number of folds, if you want to do cross-validation [-1], integer.
   - wd: weight decay [0], real.
   - out: the output file [out], string.

Misc Options:
   - seed: the random seed [-1], integer.
   - load: max number of examples to load for train [-1], integer.
   - dir: directory to save measures [.], string.
   - save: the model file [], string.

or: Linux_OPT_FLOAT/mlp --test [options] model file output_file

Arguments:
   - model: the model file, string.
   - file: the test file, string.
   - output file: the output file, string.

Misc Options:
   - load: max number of examples to load for train [-1], integer.
   - dir: directory to save measures [.], string.
```

B.2 SVM code

The same programs, as above, exist for svm.

B.3 Evaluation

We have also implemented some code to evaluate the system :

In TorchII :
- threshold_kfold to compute a threshold by cross_validation on train data.
- threshold_train to compute a threshold on train data.
- threshold_test to compute the performance on test data with the threshold computed in train.

In TorchIII : the program threshold :

```
This program will train a ThresholdTrainer to select the optimal
thresholds, save the resulting model to a file.
In mode test, this program computes a HTER optimizing the EER with
the threshold found in train.

usage: Linux_OPT_FLOAT/threshold [options] file

The argument file is the train file, string.

Model Options:
   - nt: negative target  [-1], real.
   - pt: positive target [1], real.

Learning Options:
   - kfold: number of folds, if you want to do cross-validation [-1], integer.
   - col: the column containing the score [0], integer.
   - far_factor: the factor x in DCF = FRR + x FAR [1], real.

Misc Options:
   - seed: the random seed [-1], integer.
   - load: max number of examples to load for train [-1], integer.
   - dir: directory to save measures [.], string.
   - save: the model file [], string.
   - out: the output file [], string.

or: Linux_OPT_FLOAT/threshold --test [options] model file output file

Arguments:
   - model: the model file, string.
   - file: the test file, string.
   - output file: the output file, string.

Misc Options:
   - load: max number of examples to load for train [-1], integer.
   - dir: directory to save measures [.], string.
   - G: the factor G in WER [1], real.
   - threshold_is_client: the threshold is client.
   - threshold: the threshold [1], real.
```

B.4 Code to Separate the Scores with an MLP in Fusion

The program separe_scores_mlp_mse.cc can be found in /home/learning/alves/Torch/fusion/sources/ and allows to compute the separation did by the mlp in fusion.

```
usage: Linux_OPT_FLOAT/separe_scores_mlp_mse [options] data_file model_file output_file

Arguments:
   - data_file: the testing data, string.
   - model_file: the file to load the model, string.
   - output_file: the output file, string.

Model Options:
   - nhu: number of hidden units [25], integer.
   - raw: use raw data (do not normalize).
   - pt: positive target [1], real.
   - nt: negative target [-1], real.

Misc Options:
   - seed: the random seed [-1], integer.
   - load: max number of examples to load [-1], integer.
   - threshold: threshold [1], real.
   - precision: precision [1], real.
   - rate: rate [1], real.
```

This algorithm covers the pairs (x,y), using the option rate. These pairs are printed in the ouptut file only if the distance between the output of the fusion algorithm and the threshold is less than a given number, called precision.

B.5 Scripts and Various softwares

A script to create fusion data was implemented in perl : merge_voiceData_faceData.pl. It can be found in /home/learning/alves/bin. This script takes the voice file and the face file as argument and gives the fusion file as output. The input files have the BANCA dataset format :

```
<id> <claim_id> <access> <score 1>.....<score n>
```

In our case, we have only one score for each modality. So the format is :

```
<id> <claim_id> <access> <score>
```

Another script, called merge_voiceConfidenceData_faceConfidenceData.pl does the same but the input files have the following format :

```
<id> <claim_id> <access> <score> <confidence>
```

To create the files with the confidence distance method, we have also a script :confidence_distance.pl. It can also be found in /home/learning/alves/bin. It takes as input the file (face or voice) with the score and gives in outputs the same file with a new column corresponding to the confidence. We have to do that for each modality and then merge the files with the script described above.

To do confidence with Gaussian or non-parametric methods, we use the program add_confidence in /home/learning/alves/Torch3/fusion.

```
usage: Linux_OPT_FLOAT/add_confidence [options] file test_file

Arguments:
   - file : the file, string.
   - test file : the test file, string.

Options:
   - N : number of bins [10], integer.
   - t : the threshold [0], real.
   - linear : linear
   - dir : directory [.], string.
   - out : the name of the output confidence file [confidence], string.

or: Linux_OPT_FLOAT/add_confidence --gaussian [options] file

Arguments:
   file : the file, string.

Options:
   - mean1 : mean of the scores when client [0], real.
   - sigma1 : standard deviation of the scores when client [0], real.
   - mean2 : mean of the scores when impostor [0], real.
   - sigma2 : standard deviation of the scores when impostor [0], real.
   - out : the name of the output confidence file [confidence], string.
   - dir : directory [.], string.

or: Linux_OPT_FLOAT/add_confidence --distance [options] <file>

Arguments:
   file : the file, string.

Options:
   - out : the name of the output confidence file [confidence], string.
   - dir : directory [.], string.
```

For the model adequacy method, the process have to be done at the same time than the computation of the scores for each modalities.

Some scripts were written to perform all the process of fusion. The description of the process is given in the selection procedure part in this document. The scripts allow to :

- Tune the parameters (nhu for example) doing cross_validation.
- Choose the best parameter to train and test.
- Train and save a model.

– Test the model.
– Compute the performance.

They can be found in /home/learning/alves/banca/fusion_prototype/scripts. There is one for TorchII(script_example_mlp_torch2) and one for TorchIII (script_example_mlp_torch3).