Attaques par reconstruction de données biométriques comportementales à l'aide d'alignements d'embeddings

Thomas Thebaud

DOCTORAT BRETAGNE LOIRE / MATHSTIC

LIUM
Laboratoire Informatique
Université du Mans

THÈSE DE DOCTORAT DU

LABORATOIRE D'INFORMATIQUE
DE L'UNIVERSITÉ DU MANS

ÉCOLE DOCTORALE N° 601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : *Informatique*

Par
Thomas THEBAUD

« Attaques par reconstruction de données biométriques comportementales à l'aide d'alignements d'embeddings »

Une thèse CIFRE en collaboration avec le LIUM et Orange Labs

Thèse présentée et soutenue à Le Mans Université, le 21 Octobre 2022
Unité de recherche : Laboratoire d'Informatique de l'Université du Mans - équipe LST
Thèse N° : 2022LEMA1026

Rapporteurs avant soutenance :

Najim DEHAK	Associate Professor, Johns Hopkins Whiting School of Engineering
Sébastien MARCEL	Professeur, IDIAP

Composition du Jury :

Président :	Jean-François BONASTRE	Professeur, LIA
Examinateurs :	Ruben VERA-RODRIGUEZ	Associate Professor, Universidad de Madrid
	Estelle CHERRIER	Maître de Conférence, ENSICAEN
	Grigory ANTIPOV	Ingénieur en machine learning, Ubisoft
Dir. de thèse :	Anthony LARCHER	Professeur, LIUM - Université du Mans
Co-encadrant :	Gaël LE LAN	Chercheur, Meta

Invité(s) :

Encadrant en entreprise : Simon BECOT Orange Business Services

Remerciements

Cette thèse n'est pas le travail d'un seul homme, et pas conséquent, elle ne serait pas complète sans remerciements adéquats pour ceux qui m'ont soutenu et accompagné au cours de ces trois années. Pour commencer, je remercie Najim Dehak et Sébastien Marcel, relecteurs de cette thèse, pour le temps qu'ils ont accepté d'y consacrer et pour les remarques pertinentes et précises qui ont été formulées, qui m'ont permis d'affiner davantage ce document. Je remercie aussi Jean-François Bonastre, Estelle Cherrier, Ruben Tolosana et Grigory Antipov pour avoir accepté de faire partie de mon jury et pour les questions et commentaires appréciés au cours de la soutenance.

Je souhaiterais remercier Gaël Le Lan, mon superviseur, pour avoir été à mes côtés du début à la toute fin de cette thèse, pour ses conseils et remarques, pour son honnêteté constante, et pour la motivation et les idées qu'il m'a apportées, sans lesquelles je n'aurais pas obtenu les résultats suivants. Je remercie Anthony Larcher, mon directeur de thèse, pour les échanges stimulants que nous avons pu avoir, pour les opportunités qu'il m'a offertes avec cette thèse et les évènements auxquels j'ai été amené à participer, mais également pour le temps qu'il a su me donner régulièrement. Je tiens à témoigner ma reconnaissance à Simon Bécot, pour l'environnement de travail sain qu'il a su créer et maintenir à Orange et pour ses précieuses remarques au cours de ma rédaction de thèse.

Pour l'accueil chaleureux qui m'a toujours été fait malgré une présence très irrégulière de ma part, je souhaiterais remercier tout le personnel du Laboratoire d'Informatique de l'Université du Mans, notamment les doctorants, ainsi que ceux et celles toujours prêts pour aller prendre un verre, deux ensembles pas totalement distincts. Pour les discussions intéressantes et variées que nous avons eues, qui m'ont permis de me sortir de mon sujet, je souhaiterais remercier mes collègues de Rennes.

Merci à Anne, Lucas, Mathis, Adina et particulièrement Chloé pour avoir rendu ces années à Rennes si dynamiques et agréables. Enfin, un dernier merci à ma famille : à mes parents qui ont su me soutenir et ont toujours été présents pour moi, à mes grands-parents qui se sont toujours montrés intéressés, et spécialement à ma soeur Élisa, qui a su me tirer vers l'avant grâce à sa motivation et son énergie constante.

TABLE DES MATIÈRES

INTRODUCTION

Cette thèse, intitulée *Attaques par reconstruction de données biométriques comportementales à l'aide d'alignements d'embeddings*, a été financée par une bourse CIFRE entre octobre 2019 et septembre 2022. Ces bourses sont attribuées pour les thèses réalisées dans le cadre d'une collaboration entre une entreprise et un laboratoire, ici :

1. **Orange Innovation** : Une multinationale des télécommunications, dans laquelle j'ai fait partie de l'équipe en charge de développer les futures solutions d'identité du groupe, incluant une analyse de la démarche, des tracés de chiffres manuscrits et plus tard de la parole.
2. **Le Laboratoire d'Informatique de l'Université du Mans** (LIUM) : Un laboratoire dans lequel j'ai été affilié à l'équipe travaillant sur les technologies du langage et de la parole.

Face aux problématiques de sécurité liées à la protection des données personnelles et plus spécifiquement de nos données biométriques, l'initiative a été lancée de proposer une thèse visant à étudier les vulnérabilités des systèmes utilisés face aux attaques, notamment aux attaques de vol de données et de fraude.

Les chiffres manuscrits

En 2019, Orange étudiait la mise en place d'un système d'authentification par l'analyse des chiffres manuscrits, tracés au doigt sur smart-phone ou tablette tactile. C'est cette biométrie, classée dans la catégorie des biométries comportementales de par son étude du comportement de l'utilisateur lors de son écriture d'un chiffre, qui a servi de support aux travaux réalisés dans la première moitié de cette thèse. Nous nous sommes intéressés aux menaces que pourrait représenter le vol des *embeddings* d'un utilisateur, des vecteurs en grandes dimensions extraits à partir de données biométriques et construits pour permettre de distinguer cet utilisateur par rapport à d'autres.

Sans accès au système qui les avait produits, nous avons d'abord cherché à retrouver des informations sur ces embeddings, comme le chiffre associé à chacun d'eux. Ensuite, nous avons cherché à les reconstruire dans l'objectif de frauder le système qui les avait

produits. Dans les 2 cas, les techniques employées passaient par l'utilisation d'un second système, dont les embeddings et les paramètres seraient parfaitement connus de l'attaquant. Pour permettre d'attaquer les embeddings cibles, plusieurs fonctions linéaires permettant de minimiser leur distance avec les embeddings du deuxième système sont proposées, elles sont appelées *alignements*. Leur entraînement, non supervisé, se base sur la distribution statistique des jeux d'embeddings dans leur espace.

Après l'obtention des premiers résultats, se basant notamment sur certaines propriétés liées au nombre de classes disponibles très restreintes pour les chiffres (10), et sur la faible utilisation de cette biométrie, nous avons décidé d'étendre ces attaques à un nouvelle biométrie. Orange ayant décidé de ne pas poursuivre les travaux d'exploration de cette biométrie, nous avions des raisons industrielles et académiques de ne pas poursuivre. Une biométrie plus utilisée mais également plus précise est l'*analyse de la parole* (aussi appelée *vérification du locuteur*), et c'est aussi un domaine dans lequel le LIUM a une forte activité. En revanche, l'utilisation d'une nouvelle biométrie allait poser d'autres contraintes.

La parole

La réalisation des attaques présentées précédemment sur des embeddings de parole fut plus complexe pour plusieurs raisons :

- La vérification du locuteur est plus précise que la vérification du scripteur, elle est donc plus difficile à attaquer.
- L'information contenue dans la parole est très variée. Si on ne prend en compte que l'information linguistique, c'est déjà un nombre important de contenus différents qui peuvent être prononcés dans une séquence de voix de quelques secondes. Comparée aux 10 classes différentes existantes pour les chiffres, la complexité de l'information à reconstruire sera beaucoup plus élevée. Rien ne nous assure qu'une quantité suffisante d'informations sera conservée dans l'embedding par rapport à l'audio utilisé pour reconstruire celui-ci, il nous faudra donc trouver d'autres méthodes de reconstruction.
- Les fonctions d'alignement proposées précédemment se basaient en partie sur une répartition des embeddings relative à 10 chiffres, nous avons donc dû nous adapter pour passer à la parole.

Quelques ajustements nous ont permis de faire fonctionner nos attaques contre des systèmes de vérification du locuteur, afin de mettre en évidence les dangers que pourraient représenter ces embeddings s'ils n'étaient pas suffisamment protégés.

Enfin, afin de s'ouvrir à d'autres domaines, tout en restant sur la parole et la protection des données personnelles, j'ai collaboré avec un autre doctorant sur l'étude d'un système d'anonymisation de la parole. Nous avons montré qu'il était possible d'inverser ce système dans le domaine de ses embeddings à l'aide des fonctions d'alignement utilisées précédemment.

La protection des données

Avec l'augmentation du nombre d'appareils et de services numériques utilisés quotidiennement, nous partageons de plus en plus de données personnelles avec un nombre croissant d'acteurs. Ces données pouvant être sensibles (comme des données médicales, judiciaires, financières...), le cadre de leur utilisation est très régulé. En Europe, c'est le Règlement Général de la Protection des Données (*RGPD* en abrégé) [1] qui définit les limites de leur récolte, de leur stockage ainsi que de leur utilisation. Cependant, il n'est pas simple de définir précisément le cadre d'application de ce règlement quand on touche aux technologies les plus récentes. Les réseaux de neurones en sont un bon exemple : *Un réseau entraîné avec des données personnelles constitue-t-il une donnée personnelle ? Les sorties d'un réseau de neurones sont-elles des données personnelles, si elles ont été calculées à partir de données personnelles ?* Cette thèse propose des attaques contre des systèmes d'authentification utilisant diverses biométries pour mettre en évidence les informations sensibles de ces sorties de réseaux, plus spécifiquement des *embeddings*, afin de de faciliter la mise en application et la généralisation des mécanismes de protection de nos données.

Chapitre I

L'authentification biométrique et ses limites

Notre environnement est composé d'appareils électroniques connectés par lesquels, via des interactions de plus en plus fréquentes, nous avons accès à une gamme de services en croissance constante. Pour l'utilisateur comme pour le fournisseur de ces services, l'accès contrôlé à ces services est d'une importance critique : il en va du bon fonctionnement de ces derniers, de la confidentialité et de la sécurité de l'utilisateur. Nous n'imaginerions pas laisser notre vélo sans cadenas ou notre clé sous le paillasson, c'est la même chose pour nos applications bancaires, de communications, ou de stockage par exemple.

La question centrale est donc :

> *Comment prouver son identité, sans ambiguïté, en s'assurant qu'aucune fraude ne soit possible ?*

C'est la problématique de tout système d'authentification : reconnaître les utilisateurs habilités pour les laisser entrer sans problème tout en détectant les imposteurs.

Dans ce chapitre, nous présentons les notions clés d'un système d'authentification biométrique et ses limites face à plusieurs types d'attaques. Les métriques et les données utilisées au cours de cette thèse seront ensuite détaillées. Enfin, le fonctionnement des systèmes de l'état de l'art permettant le traitement des données biométriques sera expliqué.

I.1 Le système d'authentification

Pour répondre aux problématiques d'authentification, les systèmes actuels sont construits majoritairement autour de trois grandes familles de solutions : trois facteurs d'authentifications (mémoriel, matériel, biométrique), qui peuvent être combinés pour augmenter la sécurité. On parle alors d'authentification multi-facteur.

I.1.0.0 - A Que sais-tu ? Le premier facteur s'articule autour des connaissances de l'utilisateur, incluant le fameux **"Mot de passe"**. L'utilisateur retient une information, sous forme de réponse à une question, de schéma à tracer, une succession d'images ou plus simplement une chaîne de caractères ou de chiffres, et la répète lorsqu'on lui demande de s'authentifier. Le mot de passe est simple d'implémentation et d'utilisation, théoriquement : aucun utilisateur ne peut se voir refuser l'accès à un service s'il connaît son mot de passe, et aucun attaquant ne peut s'authentifier sans avoir le bon mot de passe. Seulement, la multiplication des services entraîne une multiplication des mots de passe, et la mémoire d'un utilisateur étant limitée, ce qui entraîne donc la création de nombreuses failles de sécurité : la réutilisation du même mot de passe pour plusieurs comptes, écriture sur papier pour ne pas l'oublier (exemple du post-it avec les codes d'accès laissé sur l'ordinateur), l'utilisation de mots de passe simples pour faciliter leur mémorisation (dates de naissance, suites de chiffres, "1234"...), oubli systématique et ré-initialisation pour chaque tentative de connexion, utilisation d'ingénierie sociale [2] ou de réseaux générateurs [3] pour deviner les mots de passe...

I.1.0.0 - B Que possèdes-tu ? Le deuxième facteur s'articule autour des possessions de l'utilisateur [4]. On va vérifier que l'utilisateur est toujours en possession d'un objet physique ou numérique auquel il est normalement le seul à avoir accès, comme sa carte bancaire, son téléphone, une clé physique, un certificat ou une clé privée. Ce système permet d'offrir une meilleure sécurité pour certains cas d'usages, et peut-être utilisé en parallèle du mot de passe pour aller encore plus loin. Mais cette méthode n'est pas non plus parfaite : l'utilisateur n'est pas exempt du vol ou de la perte de cette clé physique, elle présente une difficulté et un coût d'implémentation accrus pour le fournisseur du service, et peut également prendre plus de temps à l'utilisateur pour chaque authentification.

I.1.0.0 - C Qui es-tu ? L'augmentation du nombre d'appareils portables connectés, à la puissance croissante et dotés de capteurs variés, a permis la démocratisation d'un troisième facteur d'authentification : **la biométrie** [5], ou la mesure du vivant. Nous allons étudier plus en détail la biométrie et les systèmes d'authentification biométrique [6] dans la section suivante.

I.1.1 La biométrie

De par leurs différences, les multiples caractéristiques qui permettent d'identifier un humain sont séparées en deux grandes catégories [7] :

- La mesure d'empreintes de parties du corps de l'utilisateur, de sa physiologie, parle de **biométrie physiologique**. Quelques exemples sont :
  - Empreinte digitale [8], [9]
  - Systèmes vasculaires (ex : main, doigt) [10]-[12]
  - Empreinte de Visage [13]-[16]
- La mesure du comportement de l'utilisateur, on parle alors de **biométrie comportementale**. Quelques exemples sont :
  - Parole [17]-[20]
  - Démarche [21]
  - Écriture manuscrite [22]-[24]

Le cadre de cette thèse est limité à l'étude des biometries comportementales, il sera en particulier question de la parole et de l'écriture dynamique de chiffres manuscrits. On parle dans ces deux cas de *vérification du locuteur* et de *vérification du scripteur* respectivement.

I.1.2 Comment traiter une donnée biométrique ?

La donnée biométrique est l'information numérique tirée d'une mesure physique, soit de l'empreinte du corps de l'utilisateur, soit de son comportement. Elle est mesurée par un *capteur analogique numérique*, capturant une information physique - analogique par essence - et la convertissant en données numériques. Elle a donc une dimension et un format définis, qui dépendrons du sujet de la mesure (extraits audios sur un canal ou plus pour la Parole, images en noir et blanc, en couleurs ou avec une mesure de la profondeur/de la température pour les analyses du visage, position et pression des points composant le tracé pour l'écriture manuscrite). Son traitement, schématiquement représenté dans la figure I.1 (inspirée de [6]), est beaucoup plus complexe que celui d'un mot de passe ou d'une clé physique pour plusieurs raisons :

- Les problématiques de protection des données personnelles : parce qu'on mesure directement l'utilisateur, les données traitées sont des données personnelles, ce qui

signifie qu'elles doivent être particulièrement protégées [1]. Cette contrainte est de plus en plus présente avec la démocratisation de la biométrie pour l'authentification et l'évolution des réglementations, et un des objectifs de cette thèse est de trouver des moyens de défense contre les menaces pesant sur les données biométriques lors de leur utilisation dans les systèmes d'authentification.

- Le format des données, en grandes dimensions (potentiellement variables), qui rend leur traitement logiciel plus complexe. Pour l'analyse de ces données une solution largement adoptée est l'utilisation de réseaux de neurones, présentés section I.4.
- La variabilité entre les différents échantillons analysés par le système, qui peut être causée par des bruits liés aux capteurs ou à l'utilisateur. Pour limiter l'impact des bruits tout en gardant l'information discriminante à partir d'une donnée qui peut être de taille variable, une solution technologique bien installée est le *système d'authentification biométrique à base d'embeddings* [6], [25], présenté ci-dessous. Le principe étant d'extraire l'information discriminante présente dans une donnée, et d'encoder cette information dans un vecteur, appelé *embedding*. On compare ensuite les embeddings produits lors d'une tentative d'authentification à des embeddings calculés précédemment pour savoir s'ils correspondent bien au même utilisateur.

I.1.2.1 Le système d'authentification à base d'embeddings

La figure I.1 schématise le fonctionnement d'un système d'authentification biométrique à base d'embeddings. Son utilisation se fait en 2 phases, l'*enrôlement* (premier contact de l'utilisateur avec le système, son "enregistrement") puis l'*authentification* (pour chacune des tentatives de connexion suivante).

Lors de son *enrôlement*, un utilisateur u interagit avec un *capteur analogique-numérique* (micro, caméra, capteur d'empreinte digitale, écran tactile...) qui transforme la donnée personnelle qui lui est soumise (analogique par nature) en information numérique, la donnée mesurée. La donnée mesurée par le système (x_u), est alors traitée par un *extracteur de caractéristiques* pour produire une représentation en haute dimension, nommée embedding ($e_{x,u}$), représentative de l'utilisateur u. Cet embedding est stocké dans une base de données d'enrôlement $\mathcal{E}^{enroll}$. Un seul embedding n'est pas suffisant pour être représentatif d'un utilisateur. Comme chaque donnée est légèrement différente, chaque embedding sera aussi différent, causant donc une variance entre les embeddings d'un même utilisateur. Pour évaluer cette variance, on a donc besoin d'un nombre d'échantillons suffisant. C'est donc après avoir fourni assez de données pour produire un nombre d'embeddings suffi-

sant qu'on peut considérer l'utilisateur enrôlé. Il peut ne s'agir que d'un embedding pour certains systèmes.

Par la suite, l'utilisateur enrôlé sera en mesure de se faire authentifier ou identifier, en fonction du système. L'*identification* est la reconnaissance d'un utilisateur parmi tous ceux présents dans la base d'enrôlement, quand l'*authentification* est la vérification qu'un utilisateur est bien celui qu'il prétend être. C'est le deuxième cas qui est considéré ici : le système a donc connaissance de qui l'utilisateur prétend être.

Pour la phase d'*authentification*, un utilisateur, qu'on appellera v, désire s'authentifier en tant qu'utilisateur u. Il fournit alors une nouvelle donnée physique qui, une fois mesurée par le capteur, sera appelée x_v. x_v est transformée en un embedding $e_{x,v}$ par le même extracteur de caractéristique que celui utilisé lors de l'enrôlement. Pour comparer $e_{x,v}$ aux embeddings calculés lors de l'enrôlement, on calcule ensuite un score de similarité entre $e_{x,v}$ et les embeddings d'enrôlement de u stockés, pour savoir si l'utilisateur est bien celui qu'il prétend être, c'est-à-dire si $u = v$. Si le score de similarité dépasse un *seuil*, propre au système, alors on considère que $u = v$, il est authentifié en tant qu'utilisateur u. Dans le cas contraire, on considère que la comparaison ne permet pas de l'authentifier de manière suffisante, ce qui a pour conséquence de se voir refuser l'accès au service demandé.

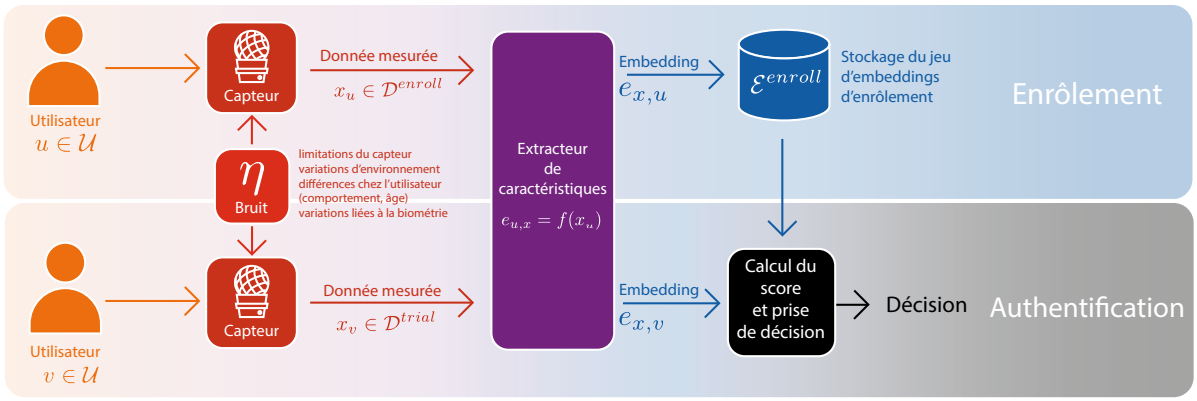

FIGURE I.1 – Architecture d'un système biométrique. Schéma tiré de [6].

I.1.2.1 - A La mesure d'une donnée physique Les systèmes biométriques sont basés sur une ou plusieurs mesures de l'utilisateur. Cette mesure numérique de caractéristiques physiques de l'utilisateur passe par un capteur analogique numérique, est représenté par la partie rouge du schéma de la figure I.1.

Il existe des différences fondamentales entre la caractéristique physique de l'utilisateur

et la donnée mesurée. La caractéristique physique est continue, et sujette à des variations et des bruits venant :

- Du comportement variable de l'utilisateur (en cas de maladie, alcoolémie, fatigue...).
- Des changements physiques de l'utilisateur dans le temps (pousse de cheveux, vieillissement, prise de poids...).
- De l'environnement au moment de la mesure.

La donnée mesurée est un signal numérique, dégradé par :

- Les imperfections du capteur
- L'échantillonnage lors du passage de l'analogique au numérique

La donnée mesurée est donc une combinaison des caractéristiques de l'utilisateur et d'une quantité de bruits η.

On pose : $\mathcal{U}$ l'ensemble des utilisateurs. Pour chaque utilisateur $u \in \mathcal{U}$, on note x_u une donnée mesurée de cet utilisateur et $\mathcal{D} := \{x_u \mid u \in \mathcal{U}\}$ l'ensemble de ces données.

Dans les schémas de cette thèse, les ensembles d'utilisateurs et les interactions de l'utilisateur avec le système seront représentés en **orange**. Les données mesurées, ensembles de données et capteurs produisant ces données seront représentés en **rouge**.

I.1.2.1 - B L'extracteur de caractéristiques Présenté en violet sur le schéma de la figure I.1, c'est la partie du système qui encode les données mesurées, x_u, en représentations de dimension $d_E \in \mathbb{N}^*$. Cette fonction est assurée par un système d'apprentissage machine (*machine learning*), le plus souvent à base de réseaux de neurones, pour compresser l'information discriminante caractéristique d'un utilisateur dans un vecteur. Ce vecteur représentatif est appelé *embedding* (ou x-vector [19] dans le cadre du traitement de la parole).

On définit la fonction d'extraction f comme :

$$f(x_u) = e_{u,x_u} \ \forall x_u, u \in \mathcal{D} \times \mathcal{U} \tag{I.1}$$

e_{u,x_u} étant l'embedding extrait d'une donnée x_u et associé à un utilisateur u. L'ensemble des embeddings est noté $\mathcal{E}$.

Dans les schémas de cette thèse, extracteurs de caractéristiques, aussi appelés "*encodeurs*" seront représentés en **violet**. Les embeddings, ensembles d'embeddings et représentations en hautes dimensions produites par un encodeur seront représentés en **bleu**.

I.1.2.1 - C L'enrôlement La phase d'enrôlement, partie supérieure du schéma de la figure I.1, est la première phase d'utilisation du système par un utilisateur. Pour reconnaître un utilisateur lors d'une authentification, il faut déjà avoir une idée de ses caractéristiques. L'enrôlement est la première récolte des données de l'utilisateur, qui permet de définir son profil, qu'on appelle *gabarit biométrique* ou *template* en anglais. De la même manière qu'on demande à un utilisateur de définir un mot de passe à la création d'un compte, et qu'on lui redemande à chaque authentification pour un système d'authentification mémoriel, l'utilisation d'un système d'authentification biométrique nécessite la création de ce *template* pour chaque utilisateur.

Ce *template* est défini comme l'ensemble des embeddings de référence $\mathcal{E}_u^{enroll}$ d'un utilisateur u. Certains systèmes demandent l'obtention de plusieurs embeddings par utilisateur, car chaque donnée prélevée est légèrement différente (à cause de bruits, voir le paragraphe I.1.2.1 - A), donc chaque embedding sera légèrement différent, causant une dispersion des embeddings pour un utilisateur donné. Dans ce cas, on génère donc suffisamment d'embeddings pour pouvoir modéliser la distribution des embeddings de l'utilisateur dans l'espace.

L'ensemble des embeddings de référence $\mathcal{E}^{enroll}$ d'un système, indépendamment de l'utilisateur qui les a généré, est appelé ensemble d'**enrôlement**.

I.1.2.1 - D L'authentification Finalement, lorsqu'un utilisateur v désire s'authentifier, il soumet une nouvelle donnée biométrique, x_v, à partir de laquelle sera calculé un nouvel embedding e_{v,x_v}. Un score de similarité entre ce nouvel embedding et le *template* est alors calculé par le système. Si le score est supérieur à un seuil propre au système, alors on considère qu'il s'agit bien du même utilisateur. *Par exemple, pour un système parfait* : si on compare le *template* de l'utilisateur u à un embedding e_{v,x_v}, si le score est supérieur au seuil τ, alors on considère que les utilisateurs u et v sont bien les mêmes :

$$\overline{e_u} := \frac{1}{Card(\mathcal{E}_u^{enroll})} \times \sum_{e \in \mathcal{E}_u^{enroll}} e, \forall u \in \mathcal{U} \tag{I.2}$$

$$score(e_{v,x_v}, \overline{e_u}) \geq \tau \Leftrightarrow v = u \tag{I.3}$$

I.1.2.1 - E Les erreurs Mais un système n'est pas toujours parfait, et peut donc commettre des erreurs. Un système d'authentification peut commettre deux erreurs lorsqu'il authentifie un utilisateur :

- S'il laisse passer un utilisateur non habilité, on parle de "*Faux positif*" ou de "*Fausse acceptation*".
- S'il ne laisse pas passer un utilisateur habilité, on parle de "*Faux négatif*" ou de "*Faux rejet*".

La quantité de faux positifs croît avec une baisse du seuil du système, en même temps que la quantité de faux négatifs décroît. Ces quantités sont utilisées pour mesurer certaines métriques, présentées partie I.3.2. Ces erreurs peuvent avoir de nombreuses causes, elles peuvent être *naïves*, c'est-à-dire résulter d'un défaut du système en fonctionnement normal, ou être dues à une attaque. Nous allons explorer ces causes ci-dessous.

I.1.3 La sécurité d'un système d'authentification biométrique

Un système d'authentification peut commettre des erreurs. Pour améliorer la sécurité d'un système, en fonctionnement normal comme face à une attaque, on cherche volontairement les cas dans lesquels il va défaillir, et les conditions associées, pour pouvoir ensuite corriger les erreurs ou établir une stratégie de défense. On peut catégoriser plusieurs cas suivant plusieurs les conditions présentées ici :

- La partie du système en cause lors de l'attaque, présentée dans la figure I.2.
- Les informations sur le système connues par l'attaquant.
- Les conditions de fonctionnement du système :
  - Pour vérifier son fonctionnement normal
  - Pour vérifier sa résistance à un utilisateur mal intentionné.
- L'intention de l'attaquant :
  - Si l'attaquant veut juste provoquer une erreur dans le système, on parle de *perturbation*.
  - Si l'attaquant veut se faire passer pour un autre utilisateur, on parle de *fraude*.

I.1.3.1 Les parties sensibles aux attaques

L'architecture standard d'un système d'authentification biométrique contient un ensemble d'éléments, chacun pouvant présenter une faille de sécurité, ainsi que chacune de leurs interfaces [6], comme présenté dans la figure I.2.

Les composants du système sont habituellement répartis sur plusieurs supports physiques par sécurité. Le capteur et l'application attendant la décision finale sont sur la

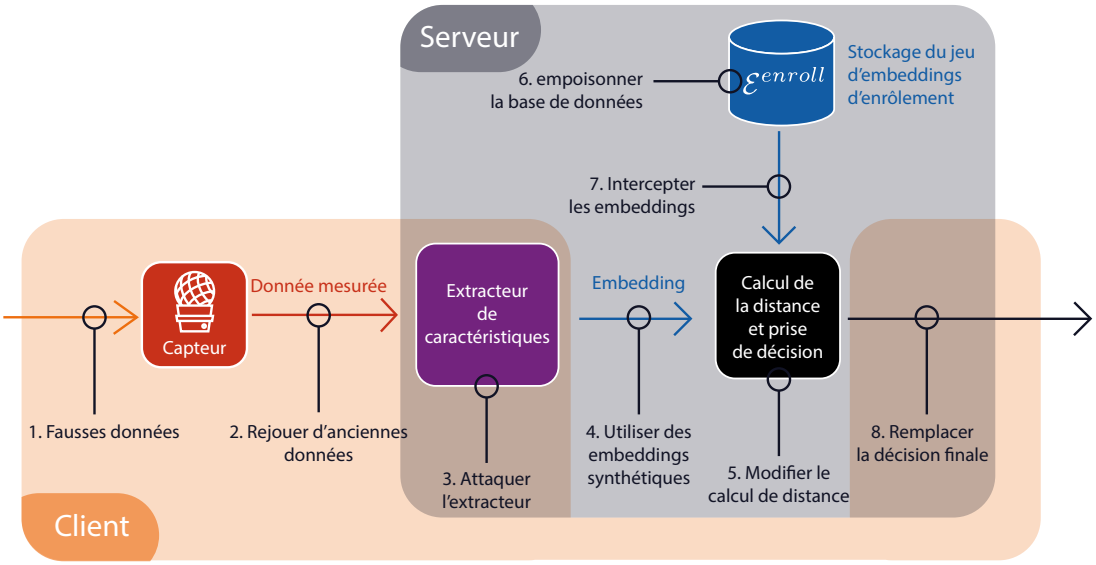

FIGURE I.2 – Failles potentielles d'un système d'authentification biométrique [6]

partie *client*, le support physique détenu par l'utilisateur. Le stockage des embeddings et la prise de décision se font sur la partie *serveur*, le support de l'organisme de confiance proposant le service. L'extracteur peut soit se trouver chez le client pour éviter le transfert de données personnelles entre supports, soit sur le serveur pour éviter de laisser au client un accès à l'extracteur, et pour permettre l'utilisation d'extracteurs demandant plus de puissance de calcul.

Quand on décide de simuler l'attaque d'un système, on peut décider d'attaquer un système avec un accès :

- *Physique* : la donnée d'attaque est appliquée au format analogique, avant le capteur.
- *Logique* : la donnée est appliquée au format numérique à un endroit précis du système, après le capteur.

Dans cette thèse, nous avons choisi de nous concentrer sur la fraude de l'extracteur de caractéristiques, n°3 dans la figure I.2. c'est-à-dire que nous allons chercher des attaques agissant sur l'extracteur, sur ses entrées (données mesurées) ou sur ses sorties (embeddings) directes. Nous supposerons aussi un accès *logique* à l'extracteur, nous permettant d'insérer directement les données de notre choix à l'entrée de l'extracteur.

I.1.3.2 Les informations connues de l'attaquant

Lors d'une attaque, on peut considérer différentes difficultés d'accès à l'encodeur, on étudie alors l'efficacité de l'attaque en fonction de la quantité d'informations accessibles.

- Si l'attaquant a un accès total à l'encodeur, (son architecture, ses paramètres, possibilité de l'utiliser), on parle d'accès en **boite blanche** [26]. L'attaquant est alors qualifié d'attaquant "*oracle*".
- S'il ne connaît rien du système mais a tout de même la possibilité de l'utiliser, on parle d'accès en **boite noire** [14], [26].
  - Dans le cadre d'un accès en boite noire, s'il possède tout de même certaines informations sur l'encodeur (architecture, méthodes d'entraînement), on peut alors parler d'accès en boite noire *informé*.
  - En général, un accès en boite noire n'a pas de condition de quantité, mais si on limite le nombre d'accès autorisés à un nombre prédéfini, on parle d'accès en boite noire *limité*.
- Si l'attaquant n'a pas accès à l'encodeur, il peut s'appuyer sur d'autres informations (sur le type de données en entrée ou en sortie de celui-ci par exemple).
- Enfin, si l'attaquant n'a aucune information, on parle d'*attaque à l'aveugle*.

Dans cette thèse, nous avons choisi de nous concentrer sur les attaques en boite noire, puis sur les attaques à l'aveugle, pour mesurer les possibilités d'un attaquant le plus indépendamment possible de l'extracteur utilisé.

I.1.3.3 L'accès utilisé

Lors de l'attaque d'un système, on distingue 2 types d'attaques [27] :

- **Les attaques de présentation**, aussi appelées attaques à *accès physique* : l'attaquant accède au système par son capteur, c'est donc une attaque plus réaliste, mais aussi plus difficile à réaliser.
- **Les attaques d'injection**, aussi appelées attaques à *accès logique* : l'attaquant injecte ses données à l'entrée d'un composant précis du système.

Dans cette thèse, nous nous limiterons aux attaques d'injection, en injectant les données d'attaques juste à l'entrée de l'extracteur de caractéristiques.

I.1.3.4 Les conditions de fonctionnement d'un système

On distingue deux type de failles possibles : l'erreur en fonctionnement normal, et l'erreur face à une attaque.

Pour mesurer les erreurs dans son fonctionnement normal, on utilise un attaquant dit *naïf*, c'est-à-dire qu'un attaquant non informé soumet une donnée non modifiée/non rejouée. L'objectif est de vérifier que le système ne confonde pas un utilisateur avec un autre, on fait alors des essais avec des données d'autres utilisateurs.

Pour vérifier sa résistance à une attaque, on définit un attaquant (par ses informations, ses intentions et la stratégie qu'il compte adopter) qui voudrait volontairement le mettre en erreur.

Dans cette thèse, les erreurs en fonctionnement normal seront utilisées pour définir la capacité de fonctionnement du système et on cherchera principalement à évaluer la capacité du système à résister à divers types d'attaques.

I.1.3.5 L'intention de l'attaquant

On peut séparer les attaques sur un système d'authentification en deux types : les Perturbations et les fraudes.

Les premières visent à perturber le fonctionnement du système. Appliquées aux extracteurs de caractéristiques, les *attaque adversarielles* [28], [29] en sont un bon exemple. Elles visent à chercher la perturbation minimale d'une variable d'entrée suffisant à faire changer la sortie (ou la prédiction) d'un système d'apprentissage profond.

Les secondes visent à permettre à un attaquant d'accéder au système, en se faisant passer pour un utilisateur habilité. Une solution simple consiste à rejouer les données de l'utilisateur visé, on parle de *replay attack*. D'autres solutions plus complexes consistent à reconstruire des données pour frauder le système, on parle d'*attaques de reconstruction*. Plus spécifiquement, lorsque la reconstruction est faite à partir des embeddings de l'utilisateur, il s'agit d'*attaques de reconstruction de templates* [9], [14], [30], [31]. L'objectif des challenges ASVspoof [27] et SASV [32] sont justement de trouver des moyens de lutter contre ces différentes attaques.

Dans cette thèse, nous avons surtout utilisé des attaques de reconstruction de templates, dans un objectif de mettre en évidence l'aspect critique de la protection des embeddings.

I.2 Les données

Dans cette section nous faisons la présentation des jeux de données utilisés. Ces jeux seront divisés de différentes manières en fonction des expériences, la répartition étant précisée à chaque fois.

Cette thèse est axée autour des moyens d'authentifications par biométrie comportementale. Afin de présenter des travaux incluant des modalités différentes dans le temps imparti, nous avons choisi les deux présentées ci-dessous :

- Les tracés de chiffres manuscrits, tracés à la main sur écran tactile.
- Les extraits de parole.

I.2.1 Chiffres Manuscrits pour la vérification du scripteur

Les tracés de chiffres manuscrits sont des séquences de $d_T \in \mathbb{N}^*$ points en 2 dimensions, collectés par tracé au doigt sur écran tactile (format smartphone ou tablette). Certains jeux de données contiennent également d'autres informations comme la pression exercée par le doigt en chacun des points mais, pour uniformiser les jeux, seules les positions des points et les moments où le doigt se pose ou se lève de l'écran seront exploités.

Nous avons utilisé 3 jeux de données, présentés dans la table I.1. Les deux premiers, *eBioDigit* [33] et *MobileDigit* [22], ont été collectés par l'université de Madrid, et sont tous les deux distribués sous licence non-commerciale avec attribution. Le dernier jeu, que nous appellerons *Interne*, a été collecté lors d'un salon par une équipe d'Orange Innovation, et n'est disponible que pour un usage interne à l'entreprise.

Table I.1 – Tableau des jeux de tracés de chiffres manuscrits utilisés dans ce manuscrit.

Jeux	Scripteurs	Fichiers par chiffre	Fichiers totaux
eBioDigit [33]	217	846	8460
MobileDigit [22]	93	743	7430
Interne	66	585	5850
Total	376	2174	21740

Chaque tracé de chiffre manuscrit est enregistré dans un fichier, et est constitué d'une séquence de positions de points en 2 dimensions ou plus, ainsi que de l'information du scripteur l'ayant tracé, et du chiffre ayant été tracé. Les séquences utilisées sont de longueur variable (moyenne = 33.5, écart type = 13.0, maximum = 254). La concaténation

de ces jeux sera nommée $\mathcal{D}_{numbers}$ pour la suite. La figure I.3 montre un exemple de 4 chiffres de ce jeu de données.

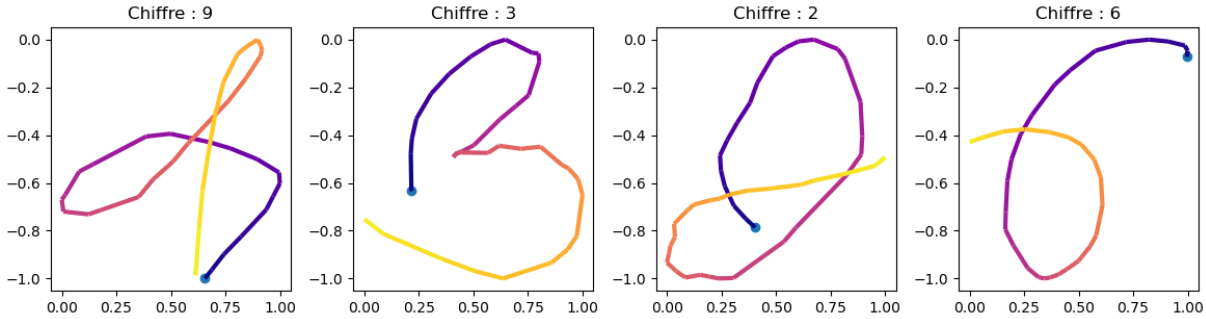

FIGURE I.3 – Graphique montrant 4 tracés de chiffres manuscrits tirés de $\mathcal{D}_{numbers}$. Le point marque le début du tracé, il est toujours placé à l'origine. Le dégradé du violet au jaune permet de percevoir l'ordre du tracé.

I.2.2 La parole pour la vérification du locuteur

Pour la vérification du locuteur, nous utilisons des extraits de parole de 2 secondes ou plus, enregistrés dans des fichiers individuels avec un échantillonage de 16kHz en mono-canal. Nous avons utilisé 3 jeux de données, présentés dans la table I.2.

TABLE I.2 – Tableau des jeux de parole utilisés au cours de la thèse.

Jeux	locuteurs	fichiers	fichiers par locuteur	heures totales
VoxCeleb1 [34]	1251	148 642	118.8	352h
VoxCeleb2 [35]	5994	1 045 732	174.4	2442h
VCTK [36]	110	42 264	384.2	+35h

VoxCeleb 1 [34] et VoxCeleb 2 [35] (*Vox* latin pour "la voix" et *Celeb* contraction de "Célébrité") sont composés d'extraits de vidéos Youtube de célébrités, qui ont été collectés pour l'entraînement de systèmes de vérification du locuteur.

VCTK [36] (*Voice Conversion ToolKit*), a été collecté en enregistrant des utilisateurs lisant des passages de journaux dans conditions contrôlées. L'objectif de la collecte était l'établissement d'une base de données pour entraîner des systèmes à la conversion de voix (voir partie III.3.1). Il contient également certaines phrases en commun prononcées par tous les utilisateurs.

Lors de l'utilisation de ces différents jeux de données, leur répartition dans les différents sous-ensembles utilisés sera précisé pour chaque expérience. La figure I.5 montre un

exemple de signal audio de parole de 2 secondes. Ce signal étant difficilement interprétable tel quel, il est généralement pré-traité en utilisant des opérations propres à chaque tâche. Différentes opérations utilisées pour les tâches de vérification du locuteur sont décrites dans la partie I.4.1.1.

I.3 Les métriques

Pour évaluer les performances d'un système, son bon fonctionnement en conditions normales ou sa résistance à une attaque, on ne peux pas se permettre de considérer individuellement les résultats sur des milliers de fichiers, nous passerons donc par des métriques pour évaluer les performances d'un système pour une tâche donnée. Plusieurs métriques vont être utilisées au cours de cette thèse, nous allons les expliciter ici.

I.3.1 L'Accuracy

L'*accuracy* est une mesure de performance de classification. C'est le pourcentage de données correctement classifiées. Soit un ensemble de données $\mathcal{D}$, leurs étiquettes associées $\mathcal{N}$ et une fonction de classification qui prédit une étiquette $n_x \in \mathcal{N}$ à partir de la donnée $x \in \mathcal{D} : f : x \mapsto \widehat{n_x}$. on définit alors l'*Accuracy* de f comme :

$$Accuracy(f, \mathcal{D}, \mathcal{N}) = 100 \times \frac{Card(\{x \in \mathcal{D} \mid f(x) = n_x\})}{Card(\mathcal{D})} \tag{I.4}$$

Par exemple : un modèle de classification qui prédit la bonne classe 9 fois sur 10 a une accuracy de 90%.

I.3.1.1 Top k accuracy

La Top k *accuracy* est une variante de l'*accuracy* souvent utilisée pour les classifieurs proposant une liste ordonnée des prédictions possibles. Si la bonne prédiction est parmi les k premières prédictions ($k \in \mathbb{N}^*$), alors la donnée est considérée correctement classifiée. Elle est notamment utilisée pour l'identification d'un utilisateur.

I.3.2 L'Equal Error Rate

Parmi les métriques qui permettent d'évaluer l'efficacité d'un système d'authentification, le taux d'égal erreur (**EER** [37], *Equal Error Rate* en anglais) donne rapidement un ordre

de grandeur de l'efficacité d'un système par la somme pondérée des erreurs qu'il commet. Un système d'authentification commet deux types d'erreurs :

- Laisser passer un utilisateur non habilité : on parle de *Faux Positif* ou de *Fausse acceptation.*
- Ne pas laisser passer un utilisateur habilité : on parle de *Faux Négatif* ou de *Faux rejet.*

La proportion de fausses acceptations parmi les résultats positifs est appelée *FAR* (en anglais, voir l'équation I.7). De la même manière on parle de *FRR* pour les rejets(en anglais, voir l'équation I.8).

I.3.2.1 Le calcul des FAR et FRR

Soit un système d'authentification $f(.,.)$ qui produit à partir d'une donnée d'enrôlement x_u et d'une donnée d'authentification y_v un score de similarité pour l'authentification. Dans le cas d'un système d'authentification biométrique à base d'embeddings, les données traitées sont des données biométriques, et la fonction f calculerait les embeddings liés aux 2 données avant de calculer un score entre les deux. Soit g la fonction qui produit un embedding à partir d'une donnée, on peut alors définir la fonction de similarité f comme :

$$f(x_u, y_v) = \frac{g(x_u).g(y_v)}{||g(x_u)|| \times ||g(y_v)||} \tag{I.5}$$

Ce qui revient à faire le produit scalaire des embeddings normalisés (c'est-à-dire dont la norme euclidienne est unitaire).

Considérons une liste de N utilisateurs $\mathcal{U} = (u_i)_{1 \leq i \leq N}$, et deux jeux de données produits par ces utilisateurs : $\mathcal{D}_x = \{x_u \mid u \in \mathcal{U}\}$ et $\mathcal{D}_y = \{y_v \mid v \in \mathcal{U}\}$. Pour savoir s'il laisse passer un utilisateur, le système d'authentification utilise un seuil $\tau \in \mathbb{R}^+$: si le score est sous le seuil, les 2 données évaluées n'appartiennent pas au même utilisateur, et inversement :

$$f(x_u, y_v) \geq \tau \Leftrightarrow u = v \tag{I.6}$$

Pour un système (f), deux ensembles de données mesurées ($\mathcal{D}_x$ et $\mathcal{D}_y$) et un seuil (τ) donnés, on a donc une mesure du FAR et du FRR :

$$FAR(f, \mathcal{D}_x, \mathcal{D}_y, \tau) = \frac{Card(\{(x_u, y_v) \in \mathcal{D}_x \times \mathcal{D}_y \mid f(x_u, y_v) \geq \tau \wedge u \neq v\})}{Card(\mathcal{D}_x) \times Card(\mathcal{D}_y)} \tag{I.7}$$

$$FRR(f, \mathcal{D}_x, \mathcal{D}_y, \tau) = \frac{Card(\{(x_u, y_v) \in \mathcal{D}_x \times \mathcal{D}_y \mid f(x_u, y_v) < \tau \wedge u = v\})}{Card(\mathcal{D}_x) \times Card(\mathcal{D}_y)} \tag{I.8}$$

Ces mesures nous permettent de caractériser la performance du système avec 2 métriques. Lorsqu'on augmente le seuil τ, on restreint l'accès au système, donc le FAR descend et le FRR monte. On parle de système sécurisé lorsque les deux taux sont le plus bas possible.

I.3.2.2 Le calcul de l'EER

Soit τ_0 le seuil pour lequel les deux taux sont égaux, alors l'EER est défini comme la valeur de ces taux d'erreurs pour ce seuil.

$$EER(f, \mathcal{D}_x, \mathcal{D}_y) = FAR(f, \mathcal{D}_x, \mathcal{D}_y, \tau_0) = FRR(f, \mathcal{D}_x, \mathcal{D}_y, \tau_0) \tag{I.9}$$

Comme l'EER est calculé sur un jeu de données contenant un nombre d'éléments fini, alors il n'existe qu'un ensemble limité de valeurs pouvant être prises par le FAR et le FRR, il est donc habituel de ne trouver des cas où aucune valeur de τ ne permet une réelle égalité. Il est cependant possible d'interpoler sa valeur, comme proposé par exemple dans le toolkit Bosaris [38]. Cependant, le jeu de données sur lequel il a été calculé peut être de grande taille, rendant son calcul précis très long pour une métrique.

On l'utilise pour comparer les performances de deux systèmes d'authentification, mais lors de leur utilisation, ce n'est pas le seuil τ_0 qui est utilisé. Comme expliqué dans [39], l'EER est une métrique indépendante du système d'authentification : elle ne prend pas en compte le coût relatif que représente chacune des erreurs (le faux rejet et la fausse acceptation).

I.3.2.2 - A Le DCF - Detection Cost Function Si on souhaite un système privilégiant la sécurité, car le coût d'une fausse acceptation est élevé, le seuil choisi sera alors plus élevé, pour restreindre l'accès. Dans le cadre d'un système privilégiant l'accessibilité, où le coût d'un rejet serait plus élevé, on choisira alors un seuil plus bas. Pour un seuil donné τ et un système donné, on peut calculer la probabilité d'un faux rejet P_{fr}^{τ}, celle d'une fausse acceptation P_{fa}^{τ}. Dans un cadre d'application donné, on peut connaître la probabilité que la tentative de connexion soit effectuée par un utilisateur habilité P_{leg} et associer à chaque type d'erreur un coût C (C_{fr} et C_{fa}). On peut calculer le DCF pour un système fonctionnant avec un seuil donné dans un cadre d'application donné en suivant

la formule suivante [39] :

$$DCF^{\tau} = C_{fa} P_{fa}^{\tau} (1 - P_{leg}) + C_{fr} P_{fr}^{\tau} P_{leg} \tag{I.10}$$

Cette fonction permet donc d'évaluer pour un cadre donné si le système est performant et si le seuil a bien été choisi. Elle fait office de fonction d'évaluation du système et de mesure de la calibration. Les valeurs utilisées habituellement dans les campagnes d'évaluation, telles que le *Speaker Recognition Evaluation plan* [40] organisé par le NIST (Institut National Américain des normes technologiques), sont de $P_{leg} = 1\%$, $C_{fr} = 10$ et $C_{fa} = 1$.

I.3.2.2 - B Le minDCF Pour évaluer un système indépendamment du seuil choisi, on peut calculer la valeur minimale de DCF possible, le **minDCF** [39] :

$$minDCF = \min_{\tau \in \mathbb{R}} DCF^{\tau} \tag{I.11}$$

Le minDCF, comme l'EER est une mesure de discrimination entre les systèmes, mais à la différence de ce dernier, il permet une évaluation dépendante de l'application.

I.3.2.3 Mesurer la fraude avec un FAR

Si on souhaite mesurer les performances en fraude lors de l'attaque d'un système, on peut utiliser le *FAR*. En effet, si l'attaquant cherche à se faire passer pour quelqu'un d'autre, il souhaite provoquer une erreur de fausse acceptation par le système. Pour mesurer la proportion d'attaques réussies, on définit un taux de fausses acceptations de fraudes, *SFAR* [14], [41] (*Spoofing False Acceptation Rate* en anglais, voir équation I.12). Ce taux est défini par le FAR d'un système lorsqu'il est attaqué, pour un seuil τ_0 déterminé à l'avance, et un jeu de données de fraude $\mathcal{D}_f$.

$$sFAR_{\tau_0}(f, \mathcal{D}_f, \mathcal{D}_y) = FAR(f, \mathcal{D}_f, \mathcal{D}_y, \tau_0) \tag{I.12}$$

Le seuil est généralement défini soit comme le seuil de l'EER (τ_{EER}, qui donne $sFAR_{EER}$), soit comme le seuil assurant un FAR choisi ($\tau_n \mid n \in [0, 100]$, qui donne $sFAR_n$)

I.3.2.3 - A Un exemple Considérons un système entraîné, qui a un EER de 3% sur les données de test. Les FAR et FRR au seuil τ_{EER} sont donc tous les deux de 3%. Si on attaque ce système avec un jeu de données destiné à le frauder, et qu'on mesure

un SFAR à 97%, alors la fraude fonctionne beaucoup mieux qu'un attaquant naïf. En revanche, si on obtient 4% de SFAR, alors cela voudrait dire que la fraude ne fonctionne pas, une majorité des intrusions venant des imperfections du système. Revenons à notre cas fonctionnel : si on veux aller plus loin, on pourra mesurer l'efficacité de la fraude face à des systèmes plus contraints. Dans l'exemple ci-dessus, $\tau_{EER} = \tau_3$, mais si on imagine un système plus contraint, configuré pour laisser passer moins d'utilisateurs non habilités, on pourrait regarder des seuils plus bas, comme τ_1 ou $\tau_{0.01}$ (qui ne laisserait passer que 1% ou 0.01% des utilisateurs naïfs). La métrique montrerait alors l'évolution de la résistance du système à la fraude en fonction du seuil choisi.

I.3.2.3 - B Mesurer la fraude pour une application : le t-DCF Pour mesurer l'impact de la fraude dans un cas d'application donné, de la même manière que pour le DCF, on peut considérer les coûts d'une fausse acceptation et d'un faux refus, la probabilité d'avoir un utilisateur habilité et la probabilité d'avoir un cas de fraude pour calculer le **t-DCF** [42]. Dérivée du DCF, cette métrique a été créée pour évaluer des systèmes comprenant une partie de vérification du locuteur et une partie de contre-mesure à la fraude. Utilisée pour différents systèmes avec un même jeu d'évaluation pour les exemples de fraudes, elle permet de comparer les performances en résistance à la fraude et vérification du locuteur. Utilisée pour un même système face à plusieurs type de fraude différentes, elle permet de comparer la performance de plusieurs attaques.

I.3.3 La Log Vraisemblance

La log vraisemblance est une mesure de comparaison entre des ensembles de points ou des distributions statistiques. Si on veut représenter un ensemble de points comme une distribution statistique, on peut utiliser un modèle de mélange de gaussiennes.

I.3.3.0 - A Le Modèle de Mélange de Gaussiennes Soit $A = \{a \in \mathbb{R}^d\}$ un ensemble de vecteurs en dimension $d \in \mathbb{N}^*$ et $k \in \mathbb{N}^*$ un nombre donné de gaussiennes, on peut calculer une distribution statistique décrivant l'ensemble des points de A par une somme pondérée de k gaussiennes, appelé *Gaussian Mixture Model* (ou *Gaussian Mixture Model* en anglais, abrégé en *GMM*).

Un GMM est défini par les poids p (scalaires entre 0 et 1), moyennes μ (vecteurs) et covariancsteames Σ (matrices) de ses k gaussiennes $\mathcal{G}$. On peut donc définir le GMM de

A comme :

$$GMM_A = \{\mathcal{G}_i = (p_i, \mu_i, \Sigma_i) \in]0, 1] \times \mathbb{R}^d \times \mathbb{R}^{d \times d} \mid i \in [\![1, k]\!]\} \tag{I.13}$$

I.3.3.0 - B La définition de la Log vraisemblance On peut utiliser la Log vraisemblance pour mesurer la distance entre un ensemble de points et une distribution statistique. Soit $A = \{a \in \mathbb{R}^d\}$ et $B = \{b \in \mathbb{R}^d\}$ deux ensembles de vecteurs de dimension $d \in \mathbb{N}^*$. Soit $GMM_A = \{\mathcal{G}_i \mid i \in [\![1, k]\!]\}$ le GMM calculé à partir des vecteurs de A.

La Log vraisemblance entre un embedding b de B et une gaussienne $\mathcal{G}_i$ de GMM_A est définie par :

$$\log \mathcal{N}(b \mid \mathcal{G}_i) = -\frac{1}{2}(k \log(2\pi) + \log(\mid \Sigma_i \mid) + (b - \mu_i)^t \Sigma_i^{-1}(b - \mu_i)) \tag{I.14}$$

La Log vraisemblance entre un embedding b de B et l'intégralité du GMM_A est alors la somme pondérée par les poids p_i de chaque gaussienne.

$$\log \mathcal{N}(b \mid GMM_A) = \log \sum_{i=1}^{k} p_i \mathcal{N}(b \mid \mathcal{G}_i) = \log \sum_{i=1}^{k} \exp(\log(p_i)) + \log \mathcal{N}(b \mid \mathcal{G}_i) \tag{I.15}$$

On définit ensuite la Log vraisemblance entre l'ensemble B et le GMM_A comme la moyenne des Log vraisemblance pour chacun des points :

$$\log \mathcal{N}(B \mid GMM_A) = \frac{1}{Card(B)} \sum_{b \in B} \log \mathcal{N}(b \mid GMM_A) \tag{I.16}$$

La Log vraisemblance permet d'obtenir une comparaison entre deux ensembles, mais cette comparaison n'est pas symétrique, et ne peux donc pas être qualifiée de distance. Pour la rendre symétrique, on définit donc en fonction des jeux A et B la Log vraisemblance symétrique, comme le maximum de A par rapport à GMM_B et de B par rapport à GMM_A :

$$LogVraisemblance_S(A, B) = \max(\log \mathcal{N}(A \mid GMM_B), \log \mathcal{N}(B \mid GMM_A)) \tag{I.17}$$

I.3.3.0 - C Mesurer la précision d'un alignement Il existe certaines situations, explorées dans les chapitres IV et V, dans lesquelles on peut souhaiter modifier la Log vraisemblance entre deux jeux de données. Pour ce faire, on peut utiliser une fonction dite

"*d'alignement*", aussi appelée *un alignement*, dont l'effet est de modifier les vecteurs d'un jeu pour le rapprocher de l'autre. La Log vraisemblance symétrique peut être utilisée pour mesurer à quel point cet alignement rapproche effectivement les deux jeux de données.

Dans cette thèse, seules les fonctions d'alignements linéaires seront considérées, elles peuvent donc être représentées dans l'espace des vecteurs par une matrice. Soit W une matrice de rotation à coefficients réels $W \in \mathbb{R}^{d \times d}$. Une matrice W à coefficients réels est une matrice de rotation si et seulement si :

- Elle est inversible et sa transposée est son inverse : $W \times W^t = I \Leftrightarrow W^{-1} = W^t$.
- Son déterminant est unitaire : $det(W) = 1$.

On définit A_W comme l'ensemble des vecteurs de A ayant subi une rotation par W, et à l'inverse B_{W^t} l'ensemble des vecteurs de B ayant subi la transformation inverse.

Soit W notre fonction d'alignement, on peut réutiliser la Log vraisemblance symétrique comme une fonction de coût à minimiser, le maximum de A_W par rapport à GMM_B et de B_{W^t} par rapport à GMM_A :

$$LogVraisemblance_S(A, B, W) = \max(\log \mathcal{N}(A_W \mid GMM_B), \log \mathcal{N}(B_{W^t} \mid GMM_A)) \tag{I.18}$$

Pour des jeux d'embeddings A et B fixés, la fonction $LogVraisemblance_S(A, B, W)$ ne dépend donc que de la matrice de rotation W. Nous utiliserons donc cette fonction pour évaluer les performances d'alignement entre deux jeux pour une matrice de rotation donnée W.

I.4 De la donnée mesurée à l'embedding

Une fois une donnée mesurée, il existe de nombreuses façons de la traiter pour en extraire les informations voulues. Dans cette thèse, il est question de traiter des données biométriques, et si l'état de l'art propose plusieurs solutions d'architectures pour l'extracteur de caractéristiques, ce sont actuellement celles à base de réseaux de neuronaux qui donnent les meilleurs résultats, nous avons donc choisi dans cette thèse de nous limiter à l'étude de ceux-ci.

Pour un système d'authentification par embedding, l'objectif du réseau qui les produit, que nous appellerons extracteur, est toujours d'encoder la donnée d'un utilisateur dans un vecteur de dimension fixée $d_E \in \mathbb{N}^*$ de manière à extraire les informations discriminant les utilisateurs autant que possible. Pour commencer, nous allons examiner le cas de la

parole, avec son pré-traitement spécifique et un exemple d'architecture pour un extracteur résiduel, utilisé dans cette thèse, puis nous verrons le cas de l'analyse de l'écriture manuscrite, utilisant des réseaux récurrents.

I.4.1 Le traitement de la parole

Dans le cas du traitement de la parole, on traite un signal sonore, les sons qui le composent sont des vibrations à différentes fréquences. Si certains systèmes utilisent directement le signal brut, la solution plus classique pour faire une analyse des différents sons est de décomposer le signal en une somme de fréquences, puis d'appliquer des transformations choisies sur ces fréquences en fonction de la tâche à réaliser, pour obtenir une représentation temps/fréquence plus facile à analyser.

I.4.1.1 Les transformations temps/fréquence et spectrogrammes

Dans cette section nous allons voir différentes transformations utilisées pour traiter des segments audio (en une dimension), afin d'obtenir des représentations temps/fréquence (en 2 dimensions) et faciliter le traitement ultérieur. C'est la phase dite de pré-traitement du signal.

I.4.1.1 - A La Transformation de Fourier L'objectif d'une transformation de Fourier est de décomposer localement une fonction continue f par une somme pondérée de fonctions périodiques intégrables $c_n(f)$. Pour une fonction périodique f de période T, continue en un réel x, et dérivable à droite et à gauche en x, le théorème de Dirichlet affirme la convergence de sa série de Fourier évaluée en x et donne l'égalité

$$f(x) = \sum_{n \in \mathbb{N}} \alpha_n c_n(f) e^{inx\frac{2\pi}{T}} \tag{I.19}$$

La suite formée par les coefficients réels $(\alpha_n)_{n\in\mathbb{N}}$ est appelée *Série de Fourier*.

I.4.1.1 - B La transformation de Fourier discrète (DFT) On peut généraliser le calcul des séries de Fourier aux fonctions non périodiques intégrables sur $\mathbb{R}$, c'est la transformation de Fourier. Comme on traite ici de signaux numériques, la fonction n'est pas continue, mais échantillonnée et à valeurs discrètes. Soit un signal s échantillonné sur

N valeurs, on décompose s comme :

$$s(n) = \frac{1}{N} \sum_{k=0}^{N-1} \mathcal{S}(k) e^{2i\pi n \frac{k}{N}} \mid \forall n \in [\![0, N-1]\!] \tag{I.20}$$

Pour estimer rapidement les valeurs $\mathcal{S}(k)$, on utilise la *Transformée de Fourier Rapide* (*Fast Fourier Transform* ou FFT).

I.4.1.1 - C La transformée de Fourier Locale (STFT) Habituellement on calcule les coefficients $s(n)$ sur des petites fenêtres glissantes du signal mesuré, c'est ce qu'on appelle une STFT [43]. Si on calcule le carré des coefficients de la STFT pour chaque fréquence, on obtient des *spectrogrammes*. Le calcul de spectrogrammes sur d_T fenêtres glissantes et d_X fréquences donne donc un spectrogramme de dimensions $d_X \times d_T$.

Ces spectrogrammes peuvent être utilisés directement pour des tâches de traitement audio, ou modifiés pour des besoins spécifiques.

I.4.1.1 - D L'échelle de Mel L'oreille humaine ne perçoit pas les différentes fréquences de manière linéaire, mais logarithmique. Pour se rapprocher de cette perception, il existe une échelle de fréquences différentes, appelée échelle de Mel. En passant les spectrogrammes à l'échelle de Mel, avec la formule ci-dessous, on obtient des *Mel-Spectrogrammes*, visibles sur la figure I.5.

$$mel = 2595 \log_{10}(1 + \frac{f}{700}) \tag{I.21}$$

I.4.1.1 - E Les MFCC Les *MFCC* [44] (Mel Frequency Cepstrums Coeficients) font partie des transformations utilisées pour les tâches de reconnaissance du locuteur. Pour calculer le MFCC d'un signal, la Transformation Cosinus Discrète (ou *Discrete Cosine Transform* en anglais, abrégé en *DCT*) est calculée à partir du logarithme d'un Mel-Spectrogramme. Les MFCC sont les amplitudes des signaux calculés par la DCT.

Les différentes transformations temps/fréquence introduites ici sont représentés dans la figure I.5, appliquées sur un segment de 2 secondes d'audio.

I.4.1.2 Le système WavLM

Récemment, la production de représentations acoustiques directement depuis le signal brut en utilisant des systèmes à base de réseaux de neurones convolutionnels a fait son apparition, et représente actuellement l'état de l'art dans l'extraction de caractéristiques

pour l'analyse de la parole. Inspiré de HuBERT [45], le meilleur système actuel de ce type est Wavlm [46], ayant montré des performances dépassant l'état de l'art pour 14 tâches de reconnaissance de la parole. Ce système est composé d'un encodeur convolutionnel à 7 couches, suivi d'un transformer [47] avec un système d'embeddings de positions relatives fermées [46]. L'encodeur convolutionnel produit les représentations acoustiques à partir d'un signal bruité, puis certaines sont masquées et le transformer apprend à reconstruire ces représentations masquées. L'architecture du système est présentée dans la figure I.4.

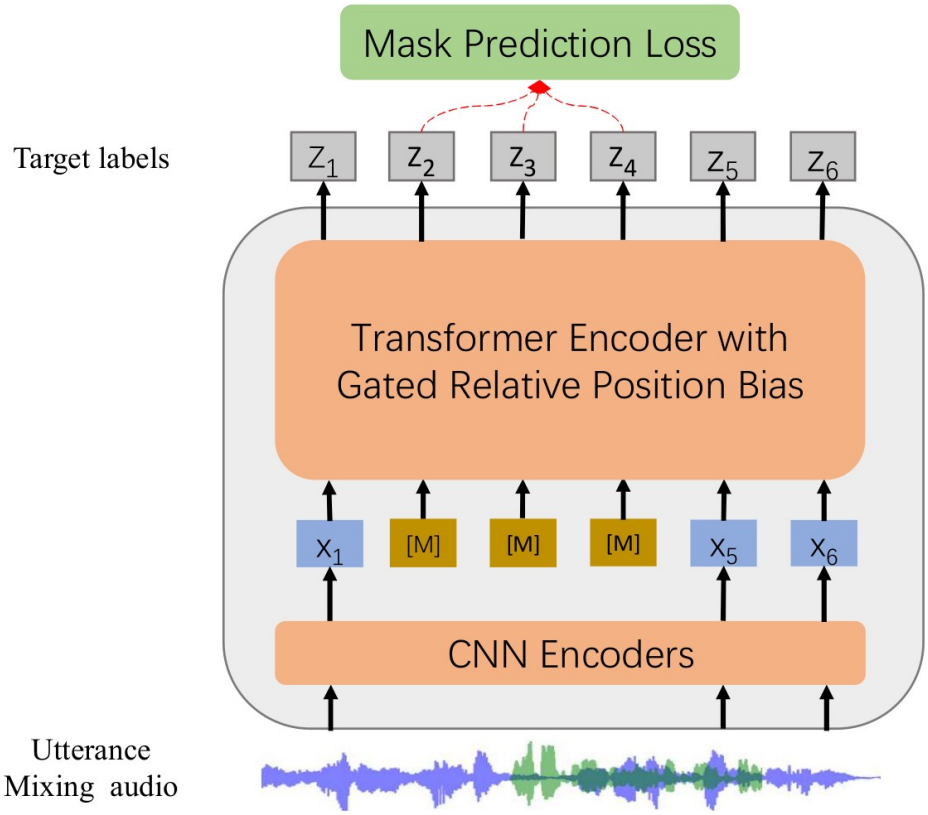

FIGURE I.4 – Architecture du système WavLM, tirée de [46].

Un exemple de transformation utilisant wavlm est également présenté dans la figure I.5

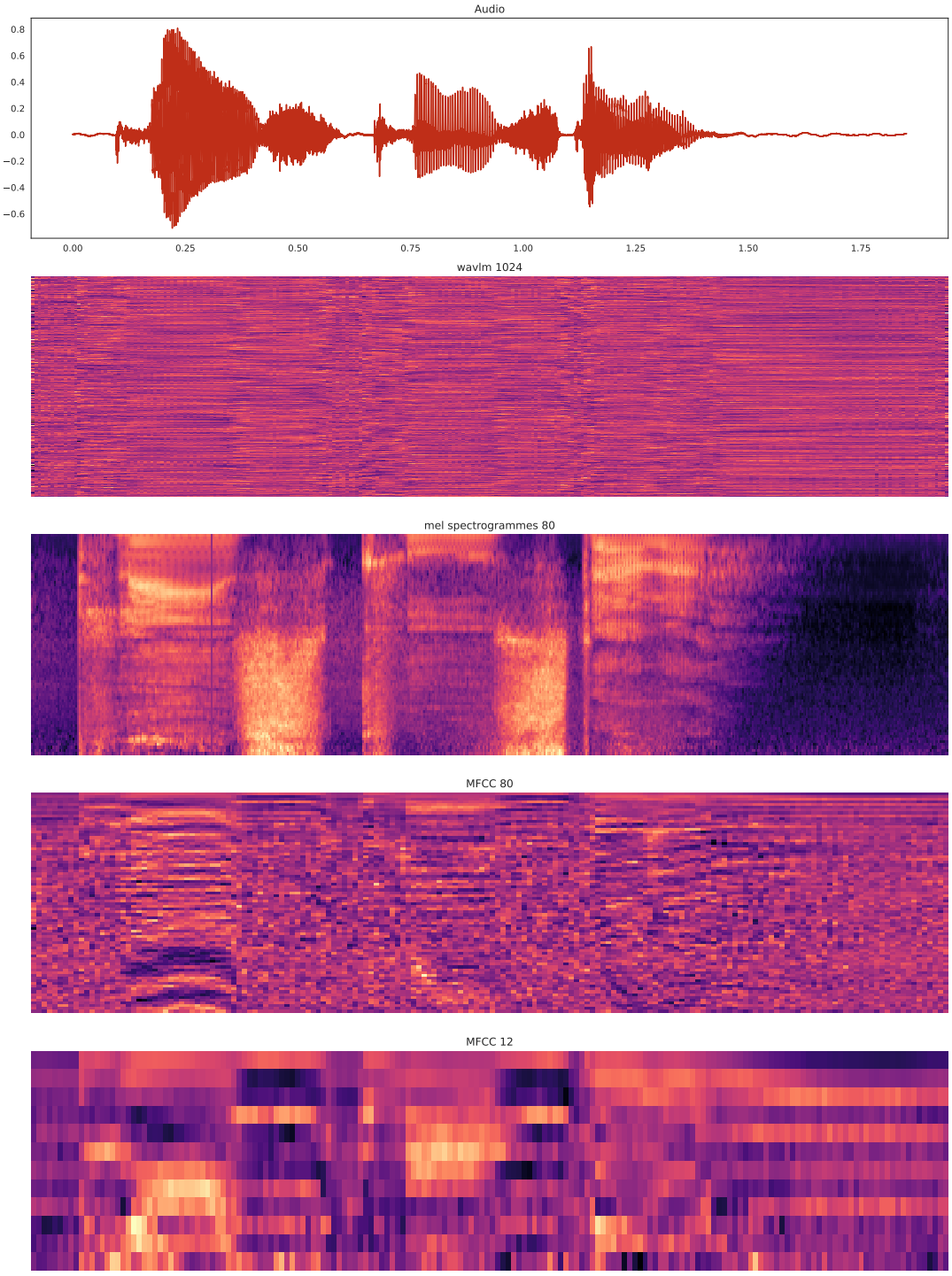

FIGURE I.5 – Graphique représentant différentes représentations possibles (Mel-Spectrogrammes, MFCC, WavLM) sur un même signal audio.

I.4.2 Les systèmes de vérification automatique du locuteur

Il existe beaucoup de systèmes de vérification et reconnaissance de locuteurs utilisant des représentations en hautes dimensions. Entre 2010 et 2016, c'est principalement le système i-vecteurs [48] (Utilisant des modèles de mélange gaussiens et un modèle du monde, dit "*GMM-UBM*") qui s'est imposé, avant de se faire remplacer par les x-vecteurs [49] (on désigne en général par x-vecteur la représentation en haute dimension d'un locuteur calculée par un réseau de neurones. D'autres architectures ont depuis été proposées, utilisant notamment des réseaux de neurones résiduels (*ResNet*) [50], [51] et des mécanismes d'attention [47] comme l'architecture *ECAPA TDNN* [52].

I.4.2.1 Comparaison des systèmes existants

Le tableau I.3 ci-dessous présente les performances de différents systèmes utilisés dans cette thèse (et ECAPA-TDNN [52] par comparaison), calculées sur VoxCeleb1 [34].

Table I.3 – Tableau des taux d'égale erreur (EER) pour plusieurs systèmes de vérification du locuteur à base de réseaux de neurones, calculés sur VoxCeleb1 test [34] avec sidekit [53] en utilisant une similarité cosinus comme score.

Systèmes	EER sur VoxCeleb1 O [34]	Utilisés
système x-vector	4.35%	✓
Fast-ResNet34	2.78%	✓
Half-ResNet34	1.67%	✓
Ecapa TDNN	1.87%	✗
Ecapa TDNN + WavLM	**0.70%**	✗

I.4.2.2 Un exemple d'architecture : le Fast ResNet 34

Le Fast ResNet 34¨ [51] est une version allégée d'un réseau de neurones constitué de 34 couches résiduelles, pour la vérification du locuteur.

Face aux problèmes de disparition du gradient [54] rencontré avec les réseaux de neurones très profonds, une solution proposée consiste à utiliser des blocs dits "*Résiduels*" [55]. Un bloc résiduel, tel que représenté dans la figure I.6, est constitué de deux couches de convolutions, et fait la somme de la sortie de ces couches et de leur entrée pour obtenir son résultat.

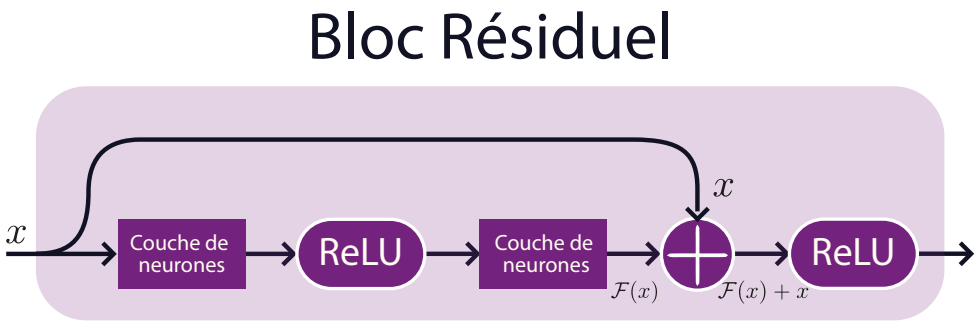

FIGURE I.6 – Schéma d'un ResBloc [55]

Le ResNet34 [55] est composé de 34 couches de convolution, soit 17 blocs résiduels, présentés dans la table I.4. Ce réseau a été conçu pour analyser des images en 32×32 dimensions, pour le challenge ImageNet [56].

L'architecture du ResNet34 a ensuite été adaptée pour être utilisée sur des tâches de vérification du locuteur [35] en lui fournissant en entrée des MFCC [44] à 40 dimensions. Cette variation est présentée dans la deuxième colonne de la table I.4. Sa version plus légère (le *Fast* ResNet 34 [51]) est présentée dans la troisième colonne de la table I.4.

C'est cette version que nous avons utilisée dans les premières expériences impliquant des systèmes de vérification du locuteur.

I.4.2.3 Sidekit

Dans ce manuscrit, les systèmes de vérification du locuteur utilisés sont tous entraînés via la librairie python sidekit [53].

I.4.3 Le traitement d'une donnée temporelle

Les biometries comportementales ont la particularité de mesurer une action réalisée dans le temps, elles nécessitent donc un traitement spécifique. Une donnée temporelle peut présenter une durée variable et des informations qui doivent être considérées chronologiquement, relativement à ce qu'il s'est passé avant ou après dans la séquence. Pour traiter cette longueur variable, il existe plusieurs possibilités, les plus utilisées étant :

- Le découpage de segments de longueur constante, utilisé surtout pour l'entraînement des systèmes [50]-[52].
- L'utilisation de statistiques calculées sur toute la longueur temporelle. [19], [51]
- L'utilisation de réseaux récurrents [23], [45], [57]

TABLE I.4 – Tableau présentant les architectures des ResNet34 [55] et FastResNet34 [53]. La composition des blocs résiduels est définie entre crochets, avec le nombre de blocs de chaque type utilisé. Pour passer à une dimension réduite, un *stride* de 2 est utilisé entre 2 lignes. Soit s le *stride* et p le *padding* utilisés, lorsqu'ils ne sont pas précisés c'est qu'ils sont respectivement à $s = 1$ et $p = 0$. A la sortie de chaque bloc résiduel $conv\_n\_x, n \in [\![2, 5]\!]$, on additionne l'entrée à la sortie avant de passer au bloc suivant.

	I	II	III
Nom	ResNet34 [55]	ResNet34 [55]	FastResNet34 [51]
Entraîné sur	ImageNet [56]	VoxCeleb2 [55]	VoxCeleb2 [55]
conv_1	$[7 \times 7] \times 64$ $s = 2$, $p = 1$ Maxpool $[3 \times 3]$	3×3, 128 $s = 2$, $p = 1$ BatchNorm 128	7×7, 16 $s = (1, 2)$, $p = 3$ BatchNorm 16
conv_2_x	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 3 \times 3, 16 \\ 3 \times 3, 16 \end{bmatrix} \times 3$
conv_3_x	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 3 \times 3, 32 \\ 3 \times 3, 32 \end{bmatrix} \times 4$
conv_4_x	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 6$
conv_5_x	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 3$
fc_1 pooling	 avg pool	9×1, 512 $1 \times N$, avg pool	9×1, 512 $1 \times N$, avg pool
fc_2 Softmax	1×1, 1000 Softmax	1×1, 5994 Softmax	1×1, 5994 Softmax
Paramètres	3.6×10^9	22×10^6	1.4×10^6

Nous allons explorer plus en détail ci-dessous la solution des réseaux récurrents.

I.4.3.1 Les réseaux de neurones récurrents

Ces réseaux sont appelés ainsi car ils possèdent au moins un élément qui sera réutilisé en boucle, ce qui permet de répéter une opération plusieurs fois [58]. Cet élément réutilisé est souvent appelé "*Cellule*". Le nombre d'itérations n'est pas fixé pour une architecture donnée, mais peut dépendre de chaque donnée, ce qui permet de traiter des données temporelles de durée variable.

Soit une donnée temporelle $x = (x_t)_{1 \leq t \leq d_T} \mid x_t \in \mathbb{R}^{d_X}$, définie comme une suite de d_T vecteurs x_t de dimension d_X. Chaque cellule d'un réseau récurrent peut s'écrire sous la forme d'une fonction RNN_{cell} telle que :

$$RNN_{cell} : x_t, h_{t-1} \mapsto h_t, o_t \mid \forall t \in [\![1, d_T]\!] \tag{I.22}$$

A chaque étape t, la cellule produit un vecteur de sortie o_t et un vecteur caché h_t à partir du vecteur caché précédent h_{t-1} et d'une donnée x_t. De la même manière qu'on a défini la série x, on peut définir les suites $h := (h_t)_{1 \leq t \leq d_T}$ et $o := (o_t)_{1 \leq t \leq d_T}$, composées respectivement de vecteurs cachés et de vecteurs de sorties. Si on fixe un premier vecteur caché h_{-1} à l'avance (comme un vecteur nul ou tiré aléatoirement d'une distribution gaussienne multivariée) et qu'on applique d_T fois la fonction RNN_{cell} aux vecteurs de x, alors on obtient les suites h et o, on peut donc écrire le réseau récurrent comme :

$$RNN_{h_{-1}} : x \mapsto h, o \tag{I.23}$$

Ce calcul est illustré dans la figure I.7. Si on souhaite obtenir un vecteur de taille fixée en sortie à partir d'une séquence, on considère soit le dernier vecteur de h, h_{d_T} comme sortie du réseau, soit sa moyenne $\overline{h}$. Si on souhaite à l'inverse obtenir une séquence à partir d'un vecteur, on peut utiliser une suite de valeurs constantes $x = (\overline{x})_{1 \leq t \leq d_T}$ en entrée, et considérer les séquences h ou o comme sortie.

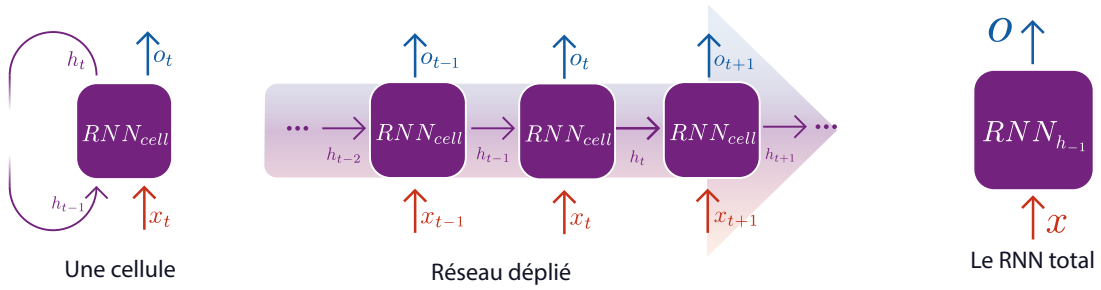

FIGURE I.7 – Schéma présentant le principe de récurrence d'un réseau de neurones récurrent

I.4.3.2 Les réseaux récurrents LSTM

Les *LSTM* (Long Short Term Memory [59]) sont des réseaux récurrents qui possèdent deux vecteurs internes : un vecteur de contexte c_t et un vecteur caché h_t pour chaque

étape t, permettant une analyse plus complète d'une séquence longue. Dans ce cas, on utilise les vecteurs h_t comme sorties. La composition de la cellule d'un LSTM lui permet de garder une mémoire à court (c_t) et à long (h_t) terme des informations vues dans la séquence. Ces réseaux ont montré une certaine efficacité il y a quelques années pour la reconnaissance de parole [60] ou l'analyse d'écriture manuscrite [61].

La composition d'une cellule LSTM, présentée dans la figure I.8, peut être définie par la fonction $LSTM_{cell}$:

$$LSTM_{cell} : x_t, h_{t-1}, c_{t-1} \mapsto h_t, c_t \tag{I.24}$$

Qui se généralise après d_T répétitions, en fixant h_{-1} et c_{-1}, par :

$$LSTM_{(h_{-1},c_{-1})} : x \mapsto h, c \tag{I.25}$$

Le contenu de la cellule du LSTM est présenté dans la figure I.8.

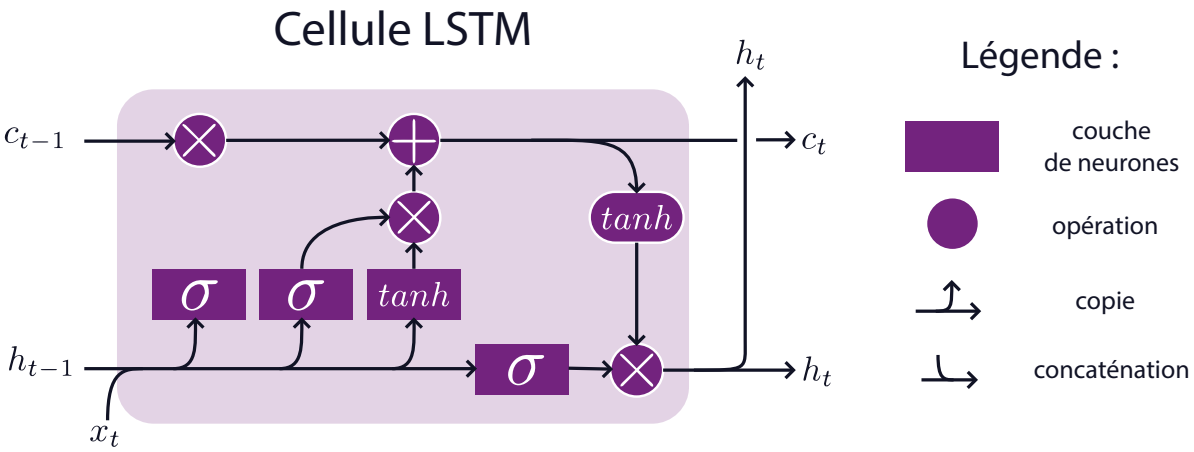

FIGURE I.8 – Schémas d'une cellule d'un réseau LSTM

I.4.3.3 Les réseaux récurrents BiLSTM

Les réseaux BiLSTM peuvent lire une séquence x dans un sens $\overrightarrow{x}$ ou dans l'autre $\overleftarrow{x}$, pour considérer l'information de la fin de la séquence en fonction de son début, et inversement. Lorsqu'on utilise deux LSTM pour lire la séquence dans les deux sens, avant de concaténer ou moyenner les sorties, on appelle ce réseau un *BiLSTM* [57]. Le concept de réseaux récurrents utilisant une séquence dans les 2 sens remonte aux BiRNN [62], proposés en 1997.

I.4.3.3 - A Les BiLSTM pour l'analyse de tracés de chiffres Dans le cas de l'analyse de tracés de chiffres, on analyse une suite de points de longueur variable en deux dimensions ou plus (certains jeux de données incluent des informations sur la pression exercée par le stylet ou le doigt, et on peut également considérer la vitesse et l'accélération du mouvement, voir partie I.2.1). Dans la majorité des cas, on lit ces séquences dans les 2 sens pour en extraire plus d'informations [23], [24], [61].

I.4.3.3 - B Les performances d'un BiLSTM pour la reconnaissance de scripteur En analysant conjointement 4 chiffres tracés à la main sur un écran tactile, [23] descend à 4.9% d'EER pour la reconnaissance de l'utilisateur (12.5% d'EER pour un unique chiffre, voir I.3 pour la définition de l'EER).

I.4.3.4 Les réseaux à densité mixtes : MDN

Les *MDN* (Mixture Density Networks) sont des réseaux qui ne prédisent pas un vecteur x en d_X dimensions, mais un ensemble de k gaussiennes $(\mathcal{G}_i)_{1\leq i\leq k}$, caractérisées par leurs coefficients (*poids*, p^i), vecteurs de moyennes (*means*, μ^i) et matrices de covariances (*covariance*, Σ^i), à chaque instant $t \in [\![1, d_T]\!]$:

$$\mathcal{G}_i = (p_t^i, \mu_t^i, \Sigma_t^i) \mid p_t^i \in [0,1], \mu_t^i \in \mathbb{R}_X^d, \Sigma_t^i \in \mathbb{R}^{d_X \times d_X}, t \in [\![1, d_T]\!] \tag{I.26}$$

Un MDN est donc composé de 3 blocs élémentaires qui produiront ces éléments à partir d'un vecteur intermédiaire h_t. On peut redéfinir des blocs élémentaires comme :

$$poids : h_t \mapsto p_t := \{p_t^i \mid i \in [\![1, k]\!]\} \tag{I.27}$$

$$means : h_t \mapsto \mu_t := \{\mu_t^i \mid i \in [\![1, k]\!]\} \tag{I.28}$$

$$covariances : h_t \mapsto \Sigma_t := \{\Sigma_t^i \mid i \in [\![1, k]\!]\} \tag{I.29}$$

I.4.3.4 - A Les réseaux LSTM-MDN Si on souhaite produire une séquence de points à partir d'un LSTM, on a vu qu'il suffisait de lui donner une suite de vecteurs de valeur constante $\overline{v}$. Si on souhaite obtenir la distribution de probabilité de présence des points dans l'espace, on joint un MDN à la fin des cellules du LSTM, ce qui nous donne cette équation pour une cellule :

$$LSTM\_MDN_{cell} : \overline{v}, h_{t-1}, c_{t-1} \mapsto \{(p_t^i, \mu_t^i, \Sigma_t^i) \mid i \in [\![1, k]\!]\} \tag{I.30}$$

Toujours en fixant les vecteurs h_{-1} et c_{-1}, on peut alors définir notre fonction LSTM-MDN comme :

$$LSTM\_MDN_{(h_{-1},c_{-1})} : \overline{v} \mapsto \{(p^i, \mu^i, \Sigma^i) \mid i \in [\![1, k]\!]\} \tag{I.31}$$

La figure I.9 illustre la structure globale du LSTM-MDN.

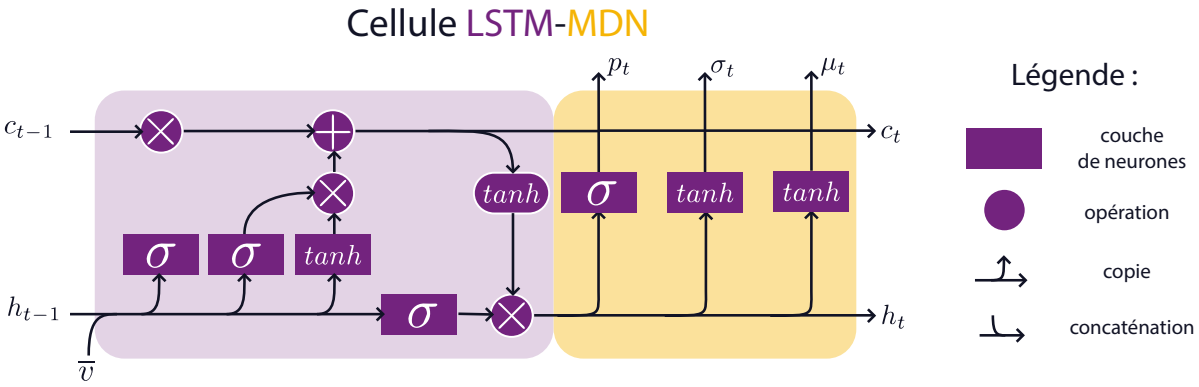

FIGURE I.9 – Schéma d'une cellule de LSTM-MDN.

I.4.3.4 - B Revenir à une séquence points après un MDN Ce réseau est entraîné de manière supervisée avec un nombre de gaussiennes fixé, en comparant la distribution de probabilité produite avec le point qu'on désirait trouver. Après l'entraînement, si on souhaite avoir la suite des points la plus probable x, on prends le centre μ_t^i de la gaussienne i_t donc le coefficient p_t^i est le plus grand, à chaque étape t :

$$LSTM\_MDN : \overline{v} \mapsto x := (\mu_t^{j_t} \mid j_t = argmax_{i \in [\![1,k]\!]}(p_t^i))_{1 \leq t \leq d_T} \tag{I.32}$$

I.4.3.4 - C L'utilisation du LSTM-MDN Ces réseaux sont utilisé pour la reconstruction de l'écriture manuscrite, car la construction des suites de points par un modèle de gaussiennes permet de prédire plusieurs emplacements concurrents pour le prochain point, et donc de modéliser beaucoup plus facilement les brusques changement de direction ou les levés de crayon. [63] propose une méthode pour générer des tracés manuscrits à partir de LSTM-MDN, qui permet de synthétiser des lettres assez fidèlement. La figure I.10 montre une illustration de champ de probabilité produit pour l'écriture du mot "under".

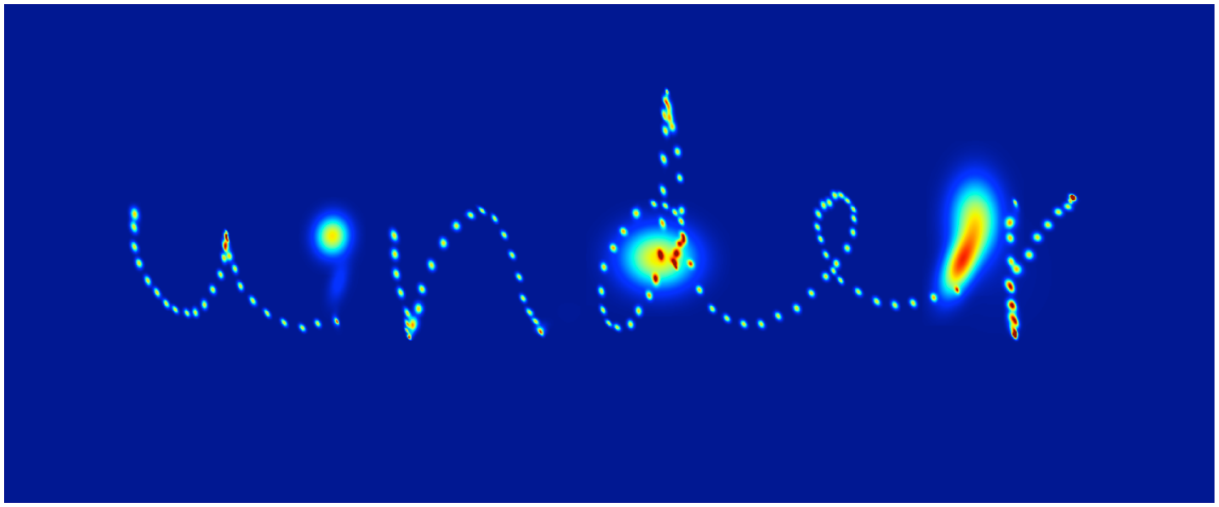

FIGURE I.10 – Illustration de champ de probabilité produit pour l'écriture du mot "under" par un LSTM-MDN dans [63].

I.4.4 Les systèmes de vérification automatique du scripteur

Parmi les systèmes de vérification du scripteur [64], analysant l'écriture manuscrite, les plus récents utilisent tous des architectures à base de réseaux de neurones récurrents, que cela soit pour une analyse de l'écriture [24], [64], des signatures [61] ou des chiffres [23], [33].

I.4.4.1 Comparaison des systèmes existants

Les systèmes de vérification automatique du scripteur utilisant uniquement des chiffres [23], [33] sont peu nombreux. La table I.5 présente les résultats des deux principaux systèmes existants (par TOLOSANA, VERA-RODRIGUEZ, FIERREZ et al. [33] et par LE LAN et FREY [23]), évalués sur une partition du jeu eBioDigit [33].

TABLE I.5 – Tableau des performances de vérification du scripteur mesuré par l'EER sur le jeu $eBioDigit_{test}$ [33].

Système	Données d'entraînement	Nombre de sessions	EER 1 chiffre	EER 4 chiffres
[33]	$eBioDigit_{train}$	4000	18.6%	9.3%
[23]	$eBioDigit_{train}$	4000	15.1%	6.6%
	$eBioDigit_{train} + internal$	8750	12.5%	4.9%

C'est le système *Bi-LSTM* [23] que nous utiliserons pour la suite de la thèse, bien que entraîné sur des données différentes en fonctions des expériences et des données disponibles.

I.5 Conclusion du chapitre

Dans ce chapitre, nous avons introduit les concepts clés liés aux systèmes d'authentification biométriques, certaines de leurs faiblesses, ainsi que les métriques, les données et les encodeurs de caractéristiques qui seront utilisés au cours de cette thèse. Ces systèmes utilisent des représentations en hautes dimensions caractéristiques des utilisateurs (aussi appelées *embeddings*), produites par un encodeur à partir des données biométriques. Ces embeddings peuvent présenter une faille de sécurité dans le système. Le prochain chapitre présente plusieurs attaques exploitant le vol d'un ensemble d'embeddings.

Chapitre II

Attaquer un système par ses représentations en hautes dimensions

Cette thèse étudie les vulnérabilités des systèmes d'authentification biométriques, utilisant la parole ou les chiffres manuscrits. Elle se concentre sur les systèmes d'extraction de caractéristiques et sur les représentations en hautes dimensions qu'ils produisent, appelées *embeddings*. Ces embeddings étant calculés à partir de données personnelles, pouvant être échangés ou stockés à long terme, ils peuvent représenter une faille de sécurité. Dans ce chapitre, nous nous plaçons du point de vue d'un attaquant qui souhaiterait exploiter ces embeddings pour en déduire des informations sur les données à leur origine. Pour commencer, j'explique les différentes informations qui pourraient être contenues dans un embedding utilisé dans le cadre de la vérification automatique de Scripteur (par les tracés de chiffres manuscrits sur écran tactile) ou de locuteur (par la parole). Ensuite, j'explique comment un attaquant pourrait retrouver des informations indépendantes de l'identité de l'utilisateur, ainsi que les limites de la méthode pour les informations propres à l'identité.

II.1 Les informations contenues dans les données et leurs embeddings

Un système d'authentification biométrique fonctionne en extrayant à partir de données biométriques (par définition des données personnelles) des informations discriminantes, permettant de faire la distinction entre les utilisateurs. Ces informations discriminantes sont compressées dans un embedding. Pour savoir quelles informations peuvent être retrouvées à partir d'un embedding, il faut d'abord savoir quelles informations sont contenues dans les données à son origine.

II.1.1 Les informations contenues dans une donnée biométrique

Une donnée biométrique étant mesurée par un capteur numérique, les informations qu'elle contient sont limitées par ce capteur. Cette limite n'empêche pas ces données de contenir beaucoup d'informations personnelles. Par exemple : une photo de votre visage ne permettra pas de compter le nombre de vos cheveux ou de savoir ce qui ce passe dans votre crâne, mais pourra permettre de savoir si c'est bien vous, de connaître la couleur de vos cheveux ou de vos yeux, votre genre ou de déduire de votre expression faciale votre état psychologique au moment de la prise de la photo. Les deux types de données utilisées lors de cette thèse sont l'écriture manuscrite et la parole, nous allons donc lister ci-dessous les informations que nous pourrions y trouver.

II.1.1.1 Les informations contenues dans l'écriture manuscrite

L'écriture manuscrite est un moyen de communication, l'écriture contient donc au minimum l'information qui a été écrite, en fonction des symboles utilisés (système de numération utilisé, alphabet utilisé, signatures,...). L'analyse de l'écriture manuscrite, aussi appelée *graphologie* [65] permet également de révéler un certain nombre d'informations complémentaires, dont certaines sont caractéristiques de l'identité du scripteur ou de son état médical et émotionnel.

II.1.1.2 Les informations contenues dans la parole

La parole est par définition :

> *L'expression et la communication de la pensée au moyen du système des sons du langage articulé émis par les organes phonateurs.* [66]

Elle contient donc au minimum une information linguistique : les mots et la langue dans laquelle ils ont été produits. Grâce aux performances des systèmes de reconnaissance du locuteur (présentés dans la section I.4.2), on peut également affirmer qu'elle contient une information suffisante pour identifier son locuteur. On sait également qu'elle peut contenir des informations comme l'âge, le genre, l'origine ethnique et géographique, l'état médical et émotionnel d'un locuteur [67].

Par souci d'éthique, on se limitera dans cette thèse aux notions d'*informations linguistiques*, d'*informations discriminant le locuteur* et de *genre* (ce dernier étant nécessaire pour évaluer et corriger les biais de genre liés à des jeux de données non équilibrés).

II.1.1.3 De la donnée mesurée à l'embedding

Un des avantages de la structure des réseaux de neurones est la variété et la taille des données d'entrée admissibles, en revanche, leur construction rend difficile l'interprétation de leur fonctionnement. Un réseau de neurones est entraîné pour une tâche ou un ensemble de tâches.

Lorsqu'on entraîne un système pour une tâche de vérification de l'utilisateur, dans un système d'authentification, on lui demande de produire des représentations des utilisateurs qui soient discriminantes. C'est-à-dire qu'on souhaite que les embeddings d'un même utilisateur soient semblables (on parle de similarité intra-classe), quand ceux d'utilisateurs différents ne doivent pas l'être (dis-similarité inter-classes).

La production des embeddings pose plusieurs questions : Si les embeddings sont calculés à partir des informations discriminant les utilisateurs, est-ce que cela veut dire qu'ils ne contiennent que les informations discriminantes contenues dans la donnée d'entrée ? Quelles sont exactement ces informations discriminantes dans les listes vues précédemment, et est-ce qu'il est possible de toutes les retrouver à partir des embeddings ? Est-il possible de retrouver des informations indépendantes de l'utilisateur, non nécessaires pour la distinction des utilisateurs, dans les embeddings ?

Il est très difficile d'affirmer qu'une information est totalement indépendante de l'identité d'un utilisateur, mais on peut déjà distinguer plusieurs types d'informations :

- **Les informations connues ou non**. *Par exemple : si la parole ou son embedding peuvent contenir l'orientation politique d'un utilisateur, sans données étiquetées avec ces informations, on ne peut pas vérifier sa présence effective.* On suppose donc qu'on ne considéra que les informations **connues**, pour lesquelles on possède au moins une base de données étiquetée avec les informations recherchées.

- **Les informations ciblées ou non**. Si on demande à un réseau de s'entraîner conjointement à extraire une information spécifique, est-ce que celle-ci a plus de chance d'être présente dans l'embedding ?

C'est cette distinction sur les objectifs de l'extracteur d'embeddings qu'on explore dans la section ci-dessous.

II.1.2 Informations ciblées et non ciblées par l'entraînement d'un réseau

L'entraînement d'un réseau de neurones cible toujours un ou plusieurs objectifs déterminés. Il peut s'agir de séparation des utilisateurs, qu'ils soient des locuteurs ou des scripteurs ; il peut s'agir de classification, comme pour les chiffres ou les phonèmes (si on veut faire de la reconnaissance automatique de la parole). On évalue la différence entre la production réelle d'un réseau de neurones et la production ciblée par une fonction, appelée *fonction de coût*. Lors de l'entraînement d'un réseau, on cherche donc à minimiser une ou plusieurs fonctions de coûts, pour adapter le comportement du réseau à un ou plusieurs objectifs.

Dans certains cas, on pourra avoir plusieurs tâches à assurer conjointement à partir d'une même donnée, comme pour la vérification du scripteur [23], en utilisant alors plusieurs fonctions de coût : une pour la reconnaissance du chiffre et une pour la vérification du scripteur. Dans d'autres, les données pourront contenir plusieurs informations dont seules certaines seront ciblées, comme pour la vérification du locuteur, où la parole contient une information linguistique et une information discriminant les locuteurs, mais où seule la dernière est ciblée. Nous allons examiner comment la présence ou non d'objectif défini favorise la transmission de ces informations annexes.

II.1.3 La reconnaissance conjointe du scripteur et des chiffres

Quand on analyse des tracés de chiffres manuscrits pour l'authentification, on analyse également le chiffre tracé, le code formé par un ensemble de chiffres pouvant servir comme une sécurité supplémentaire (soit comme un mot de passe à usage unique (*One Time Password* en anglais, abrégé en OTP), soit comme un code PIN). On demande donc la plupart du temps à un système de prédire à la fois l'information du chiffre tracé et de l'identité du scripteur. Ici le réseau cible deux informations lors de son entraînement :

1. Le chiffre tracé, dont les possibilités sont limitées et connues ($n \in [\![0, 9]\!]$)
2. l'identité du scripteur, dont les possibilités sont théoriquement limitées au nombre d'utilisateurs enrôlés, mais qui peuvent être considérées comme infinies (on attribue à chaque utilisateur une étiquette unique, un chiffre entier $u \in \mathbb{N}$)

Utiliser un unique réseau pour prédire deux informations plutôt que d'en entraîner 2 séparément permet également de gagner du temps de calcul à l'entraînement et à l'utilisation.

Mais quid de l'impact sur les performances des deux tâches ?

II.1.3.1 Présentation d'un système de vérification du scripteur

Le Lan et Frey [23] ont proposé un système permettant de prédire les chiffres et le scripteur à partir de tracés de chiffres. Ce système, un BiLSTM RNN [57], est entraîné sur une combinaison des bases de données de tracés de chiffres *E-BioDigit* [33] et un jeu de données interne (décrits section I.2). Pour entraîner ce système, deux fonctions de coût (des entropies croisées, ou *cross entropy*) peuvent être utilisées, pour entraîner le système à prédire le scripteur, à reconnaître les chiffres, ou les deux simultanément si utilisées conjointement. Ces fonctions sont respectivement décrites dans les équations II.1 et II.2.

Uniquement lors de l'entraînement du réseau, deux couches de neurones indépendantes suivies d'une fonction *softmax* sont ajoutées après la couche produisant les embeddings pour prédire respectivement la probabilité qu'un tracé ait été fait par un utilisateur, et la probabilité qu'il s'agisse d'un chiffre donné. Soit U l'ensemble des scripteurs $(u)_{u\in[1,n_U]}$ et D l'ensemble des chiffres $(d)_{d\in[0,9]}$, soit S_k le tracé d'un chiffre (pour un ensemble des tracés de taille M, $k \in [1, M]$) : la probabilité qu'un tracé S_k ait été tracé par un utilisateur u est définie par $p_{S_k\in u}$, et elle est obtenue en lisant la u^{eme} dimension du vecteur produit par la couche softmax prédisant les utilisateurs. C'est la même chose pour les chiffres, avec $p_{S_k\in d}$ la probabilité associée. Soit $\mathbb{1}_{[S_k\in u]}$ un opérateur qui vaut 1 si S_k a été tracé par l'utilisateur u, 0 sinon, on peut définir la fonction de coût servant à entraîner le système à prédire les utilisateurs comme :

$$(L)_u = -\frac{1}{M}\sum_{k=1}^{M}\sum_{u=1}^{n_U}\mathbb{1}_{[S_k\in u]}\log(p_{S_k\in u}) \tag{II.1}$$

De la même manière en utilisant l'opérateur $\mathbb{1}_{[S_k\in d]}$ pour les chiffres, on définit la fonction de coût suivante :

$$(L)_d = -\frac{1}{M}\sum_{k=1}^{M}\sum_{d=0}^{9}\mathbb{1}_{[S_k\in d]}\log(p_{S_k\in d}) \tag{II.2}$$

Si on souhaite entraîner le système aux deux tâches, on utilise donc la somme des deux fonctions :

$$(L)_{u+d} = (L)_u + (L)_d \tag{II.3}$$

II.1.3.2 Les performances en vérification du scripteur avec et sans reconnaissance du chiffre

Nous avons commencé par comparer les performances de deux systèmes de vérification du scripteur, avec ou sans reconnaissance du chiffre conjointe, pour examiner les variations de performances liées à l'ajout d'une tâche.

Les résultats sur les performances de vérification du scripteur après entraînement (évalués sur la partition d'évaluation de eBioDigit [33]) sont présentés sous forme d'EERs dans la table II.1 :

TABLE II.1 – Tableau des EER obtenus sur le sous-ensemble d'évaluation [23] du jeu eBioDigit [33] par l'entraînement de systèmes avec ou sans fonction de coût sur les chiffres [23].

Objectif(s)	Fonction de coût	EER sur 1 chiffre	EER sur 4 chiffres
Scripteur	$(L)_u$	18.0%	9.9%
Scripteur et chiffre	$(L)_{u+d}$	**12.5**%	**4.9**%

Ces résultats montrent que le système obtient de meilleurs résultats sur la vérification du scripteur en prédisant conjointement le chiffre et l'identité qu'en prédisant uniquement l'identité du scripteur.

II.1.3.3 Les performances en reconnaissance du chiffre avec et sans vérification du scripteur

Ensuite, nous avons cherché à vérifier l'impact de la vérification du scripteur sur les performances en reconnaissance du chiffre. Nous avons entraîné 2 réseaux BiLSTM sur les séquences de chiffres manuscrits disponibles $\mathcal{D}^{numbers}$ pour mesurer la différence de précision en reconnaissance du chiffre avec et sans utilisation d'une fonction de coût pour la vérification du scripteur. $\mathcal{D}^{numbers}$ est séparé en $\mathcal{D}^{train}$ et $\mathcal{D}^{test}$, contenant respectivement 300 et 74 utilisateurs disjoints (80%/20% du jeu de données). Les deux réseaux sont entraînés pour 100 époques sur $\mathcal{D}^{train}$ avec comme fonction de coût une entropie croisée, un optimiseur Adam [68] et un learning rate de 10^{-4}. Leurs performances sur le jeu $\mathcal{D}^{test}$ sont présentées dans le tableau II.2 :

TABLE II.2 – Tableau des performances (voir section I.3) sur la reconnaissance des chiffres obtenues sur $\mathcal{D}^{test}$, avec et sans entraînement conjoint sur la reconnaissance du scripteur.

Objectif(s)	Fonction de coût	Accuracy
Chiffre	$(L)_d$	96.81%
Sscripteur et chiffre	$(L)_{u+d}$	**97.41**%

On remarque que les performances en reconnaissance du chiffre sont légèrement meilleures lorsque le système est conjointement entraîné pour une tâche de vérification du scripteur. On fera donc le choix d'entraîner toujours conjointement la vérification de scripteur et la reconnaissance de chiffre dans la suite de ce document.

II.1.4 La place de la parole dans la vérification du locuteur

Quand on analyse des segments de parole en vérification du locuteur, les informations connues sont souvent uniquement le genre et l'identité du locuteur, et le seul objectif du système est d'extraire l'identité du locuteur. Cependant : la donnée analysée est de la parole, et comme vu précédemment, même sans en connaître la transcription, un segment de parole contient toujours une information linguistique. Cette information ne peut donc pas être un objectif lors de l'entraînement d'un réseau sans en connaître la transcription, mais cela ne nous assure pas qu'elle ne soit pas partiellement présente dans les embeddings produits par un système de vérification locuteur. Nous ne ferons pas d'expériences pour mesurer l'impact de la tâche de transcription sur les performances en vérification du locuteur dans la suite de cette thèse.

Nous savons qu'il existe de nombreuses informations dans une donnée biométrique [69], mais que seules une poignée d'entre elles sont mesurables (en fonction des informations présentes dans les bases de données, et de la volonté de faire apprendre au réseau comment extraire cette information ou pas), nous allons donc examiner quelles informations il est possible de retrouver à partir des embeddings de chiffres manuscrits et de parole.

II.2 Les informations disponibles à partir des embeddings

On a vu que les données utilisées étaient composées d'une multitude d'informations, pas toujours connues, étiquetées ou ciblées par les réseaux qui les traitent. En revanche, la présence de ces informations a un impact sur les embeddings produits par un réseau de neurones. Dans le cas d'un système d'authentification, elles produisent des variations entre les embeddings d'un utilisateur, qu'on appellera *variance intra-utilisateur*.

Dans cette section, nous cherchons s'il est possible de retrouver certaines des informations composant la donnée d'entrée, à partir d'un ensemble d'embeddings. Dans un système d'authentification biométrique utilisant des embeddings, l'ensemble des embeddings d'enrôlement peut présenter une faille de sécurité, car ils sont échangés et stockés, chaque échange ou enregistrement pouvant présenter une faille de sécurité potentielle.

Notre première hypothèse est qu'un attaquant a pu se procurer un jeu d'embeddings d'enrôlement non chiffrés, provenant tous d'un extracteur auquel il n'a qu'un accès restreint. Pour simplifier le problème dans un premier temps, on supposera que l'attaquant connaît la structure de l'extracteur qu'il attaque, car on utilisera des structures classiques pour les tâches étudiées. En suivant le *principe de Kerckhoffs* [70], qui suppose que la sécurité d'un système ne repose pas sur le secret de son architecture, nous motivons notre connaissance de l'architecture des réseaux attaqués.

II.2.1 L'attaque des embeddings de chiffres : retrouver les chiffres

Les systèmes pour l'authentification à partir de chiffres manuscrits sont entraînés pour prédire à la fois la valeur du chiffre et l'identité du scripteur. Il parait normal qu'on puisse trouver une information sur le chiffre dans les sorties de l'extracteur. Nous avons donc cherché quelles informations sur les chiffres étaient déductibles d'un ensemble d'embeddings de chiffres, de manière non-supervisée (expérience II.2.1.1) puis supervisée (expérience II.2.1.2).

II.2.1.1 Le regroupement non supervisé des embeddings de chiffres

La première expérience a consisté à analyser les embeddings de chiffres manuscrits de manière non supervisée du point de vue d'un attaquant. Si l'attaquant sait qu'il attaque un système analysant des chiffres, alors il sait devoir trouver 10 groupes de vecteurs (pour

10 chiffres), donc un clustering (en utilisant l'algorithme des K-Means [71]) devrait lui permettre de regrouper en fonction de leur chiffre les vecteurs.

Pour cette expérience, on utilise les données de tracés de chiffres manuscrits $\mathcal{D}_{numbers}$ présentés partie I.2.1. $\mathcal{D}_{numbers}$ est séparé en 2 jeux composés d'utilisateurs disjoints ($\mathcal{D}^{test}_{numbers}$ et $\mathcal{D}^{train}_{numbers}$, contenant chacun une moitié des utilisateurs). Un réseau BiLSTM avec une couche cachée de dimension 256 a été entraîné sur $\mathcal{D}^{train}_{numbers}$ à reconnaître les chiffres (avec une précision de 97.32% sur $\mathcal{D}^{test}_{numbers}$) et les scripteurs (avec un EER de 12.72% sur $\mathcal{D}^{test}_{numbers}$). Ensuite, on passe $\mathcal{D}^{test}_{numbers}$ dans le réseau entraîné pour obtenir un ensemble d'embeddings $\mathcal{E}^{test}_{numbers}$, et on effectue un *clustering* avec 10 *clusters* en utilisant K-means [71]. L'expérience est résumée dans le schéma présenté dans la figure II.1 Les opérations de clustering utilisées dans cette thèse seront représentées en **vert** sur les schémas.

On observe que chaque groupe est composé d'une majorité d'embeddings correspondant à un même chiffre, permettant d'attribuer à chaque *cluster* une étiquette correspondant à la valeur du chiffre présent majoritairement. On évalue la précision du *clustering* en faisant le rapport entre les embeddings associés au *cluster* ayant la bonne étiquette (majoritairement constitué de même chiffre qu'eux) et le total des embeddings.

On trouve une précision de **98.96%**. Ce résultat est à mettre en comparaison avec la précision de prédiction du réseau lui-même sur le même ensemble : **97.32%**.

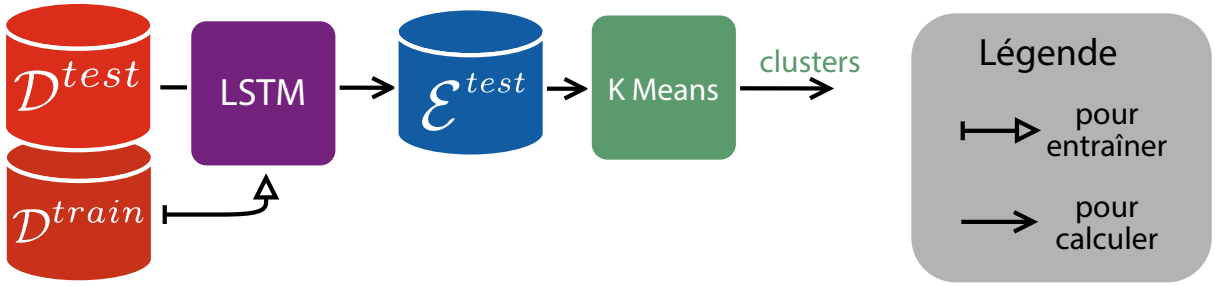

FIGURE II.1 – Schéma résumant l'expérience n°II.2.1.1. $\mathcal{D}^{train}$ entraîne l'encodeur (LSTM), puis $\mathcal{D}^{test}$ est utilisé pour produire les embeddings composants $\mathcal{E}^{test}$, qui seront ensuite regroupés par $K-means$

La figure II.2 montre la matrice de confusion pour le clustering effectué avec K-means, où l'on peut voir que, même sans connaître la valeur des chiffres de chaque cluster, la démarcation est bien nette.

Pour aller plus loin, après la publication de ces résultats à ACNS 2020 [72] (*International Conference on Applied Cryptography and Network Security*), nous avons décidé

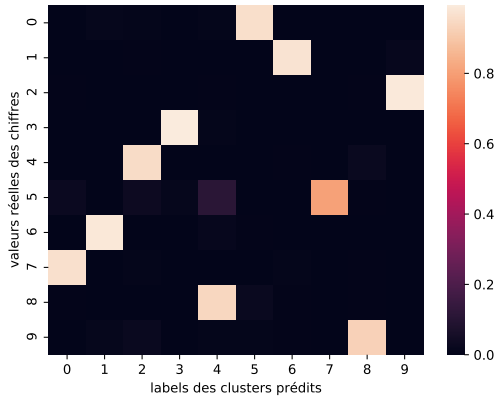

FIGURE II.2 – Matrice de confusion lors du clustering des embeddings de chiffres manuscrits par K-means.

de comparer la précision du clustering obtenu avec celui d'autres méthodes. Les résultats sont présentés dans la table II.3.

TABLE II.3 – Tableau des résultats de précision du clustering sur un jeu d'embeddings de chiffres manuscrits.

Méthode	Précision du clustering
KMeans [71]	98.96%
Spectral clustering [73]	98.95%
Agglomerative Clustering [74]	98.95%
DBSCAN [75]	98.10%
En utilisant les prédictions du LSTM	97.32%

La méthode employée pour faire un regroupement semble peu influer sur la précision du clustering, la table II.3 montrant des résultats proches des résultats de l'encodeur, sûrement limités aux performances de ce dernier. Une fois que nous avons des clusters non étiquetés de chiffres, la question est de savoir si nous pouvons savoir à quel chiffre correspond quel cluster de manière non supervisée.

II.2.1.2 Etiquetage non supervisé des clusters d'embeddings

Nous avons montré qu'il était possible de former des clusters d'embeddings pertinents vis-à-vis de leurs chiffres d'origine. Une fois les groupes formés, est-il possible de trouver

la valeur du chiffre associé à chaque groupe ? Notre hypothèse est qu'il est possible d'inférer des informations sur les clusters de chiffres à partir de leur distribution individuelle (variance dans l'espace) et de la topologie de l'ensemble de clusters (position relative des clusters : un 6 est peut-être plus proche d'un 9 ou d'un 0 que d'un 4, voir figure II.3). Pour exploiter cette information, nous avons décidé de passer par la comparaison avec d'autres ensembles d'embeddings.

Pour commencer, nous avons donc séparé $\mathcal{D}_{numbers}$ (présenté section I.2.1) en 4 jeux composés d'utilisateurs disjoints ($\mathcal{D}^{test}_{target}$, $\mathcal{D}^{train}_{target}$, $\mathcal{D}^{test}_{attack}$ et $\mathcal{D}^{train}_{attack}$, contenant chacun un quart des utilisateurs). On commence par entraîner deux réseaux BiLSTM ($BiLSTM_{target}$ et $BiLSTM_{attack}$) avec une couche cachée de dimension 256 sur $\mathcal{D}^{train}_{attack}$ et $\mathcal{D}^{train}_{target}$ à reconnaître les chiffres et les scripteurs. En passant les jeux de données $\mathcal{D}^{test}_{attack}$ et $\mathcal{D}^{test}_{target}$ dans ces deux réseaux, on obtient respectivement les jeux d'embeddings $\mathcal{E}_{attack}$ et $\mathcal{E}_{target}$, composés chacun de 5600 embeddings issus de 94 scripteurs distincts. Pour cette expérience, on suppose que le jeu cible (*target*) $\mathcal{E}_L$ est étiqueté, et que l'autre, celui qui servira pour l'*attaque*, ne l'est pas.
On montre une projection des embeddings de $\mathcal{E}_{attack}$ en 2 dimensions dans la figure II.3.

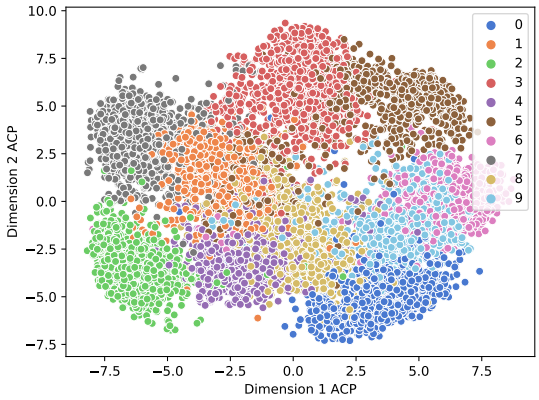

FIGURE II.3 – Projection par ACP en 2 dimensions des embeddings de $\mathcal{E}_{attack}$, extraits par le réseau $BiLSTM_{attack}$ à partir des données de $\mathcal{D}^{test}_{attack}$. Couleurs par chiffre associé.

On cherche la valeur du chiffre associé à chacun des embeddings à partir de la structure des clusters. Pour trouver cette valeur, on va aligner un ensemble d'embeddings de valeur de chiffres connues à un ensemble aux valeurs de chiffres inconnues. On évalue les performances de l'alignement avec une fonction de log vraisemblance.

Pour trouver cet alignement optimal, on procède par étapes, résumées dans la figure

II.4 :

1. On commence par former des groupes de même chiffres
   (a) avec les valeurs connues pour $\mathcal{E}_{attack}$
   (b) avec un clustering K-Means pour $\mathcal{E}_{target}$
2. On normalise les embeddings de chaque jeu.
3. On pré-aligne les jeux en faisant une Analyse en Composantes Principales [76] en 10 dimensions sur chacun.
4. On fait un alignement en utilisant l'analyse de Procrustes [77] et un Algorithme génétique [78]. Les opérations d'alignement utilisées dans cette thèse sont représentées en **vert** sur les schémas.
   (a) Si on connaissait la correspondance 1 à 1 entre les clusters des 2 jeux, on pourrait utiliser l'*analyse de Procrustes* [77] pour calculer la rotation qui rapprocherait de manière optimale les deux espaces.
   (b) Comme il existe 10 clusters, on a 10 !=3628800 combinaisons possibles.
   (c) Pour chaque combinaison possible, on calcule la rotation associée, on projette un jeu sur l'autre, et on mesure la similarité entre les deux avec une métrique de *log vraisemblance* (voir paragraphe I.3.3).
   (d) Finalement, on utilise un algorithme génétique pour trouver la combinaison qui donne le score de similarité le plus haut sans explorer toutes les combinaisons.

Une fois cet algorithme testé plusieurs fois sur 100 découpes différentes du jeu initial, nous avons trouvé que la combinaison en première position était la bonne dans 62% des cas. Une analyse plus poussée nous a montré que dans 99 cas sur 100 la combinaison recherchée était classée au pire 12e, et dans le dernier cas elle était classée 20e.

Ces résultats s'expliquent peut-être car les matrices de rotations candidates ne sont produites qu'à partir des centres des 10 clusters, sans prendre en compte la variance des clusters dans l'espace des embddings. Nous avons donc ajouté une étape supplémentaire à notre algorithme d'étiquetage : Une fois qu'on a la liste des combinaisons avec les meilleurs scores, on affine chacune des 20 meilleures combinaisons candidates :

1. Pour une combinaison donnée, on peut utiliser l'analyse de Procrustes et calculer la matrice de rotation associée.
2. La matrice peut- être considérée comme un ensemble de paramètres variables, qu'on peut alors affiner avec une descente de gradient stochastique [79] en utilisant la *log vraisemblance* comme fonction de coût à minimiser.

3. Une fois affinée, on peut utiliser sa valeur de *log vraisemblance* comme nouveau score.

La combinaison avec le meilleur score après affinage est la vraie combinaison.

Une fois cette dernière étape ajoutée, sur 100 découpes différentes du jeu initial, on a à chaque fois la bonne combinaison qui arrive première, soit une précision de 100% sur l'étiquetage des clusters.

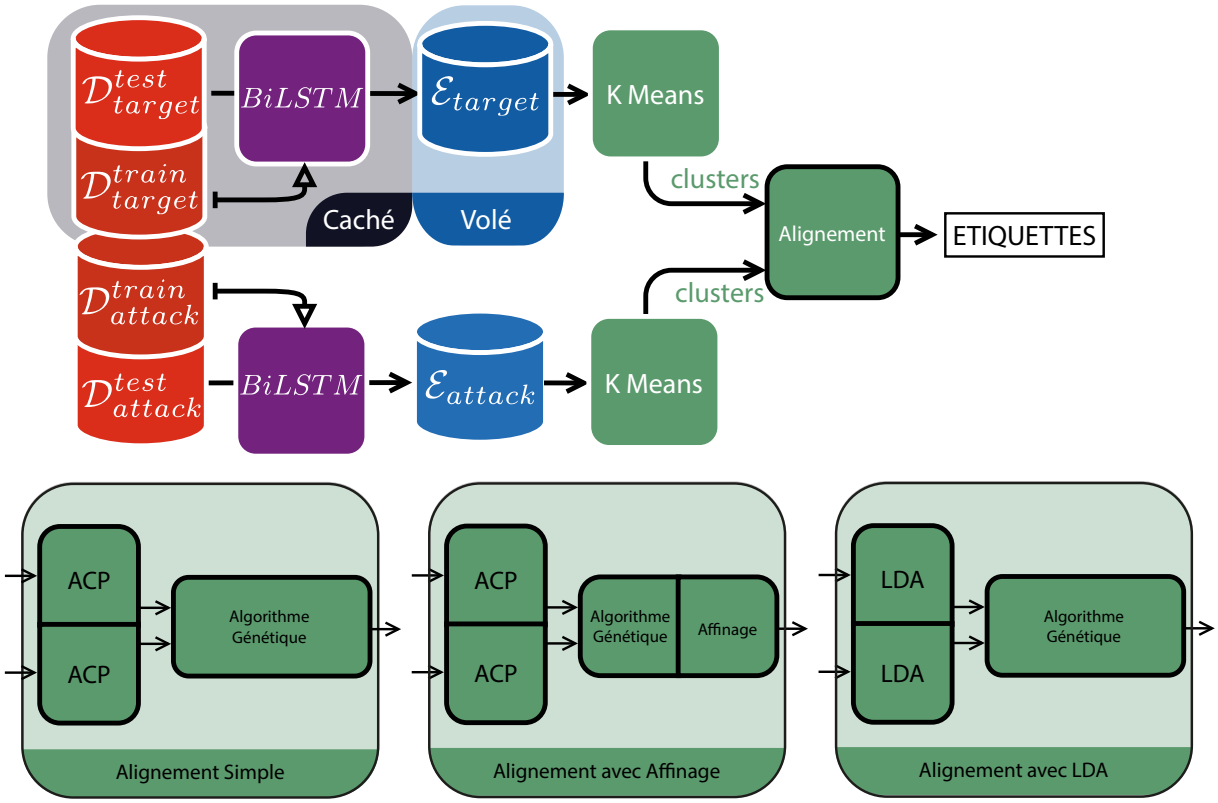

FIGURE II.4 – Schéma représentant l'expérience n°II.2.1.2. 3 méthodes d'alignement sont présentées : la méthode initiale utilisant un algorithme génétique à la suite d'une ACP, sa dérivée ajoutant un affinage sur les meilleures solutions candidates, ainsi que l'amélioration utilisant une LDA.

II.2.1.2 - A Les limites de la méthode Si cette méthode montre de bons résultats, elle reste limitée, car elle impose de pouvoir utiliser un réseau de même architecture pour créer un jeu d'embeddings similaires, elle ne fournit qu'un étiquetage par cluster, donc véridique uniquement pour les embeddings correctement classifiés par les algorithmes de clustering. Enfin, elle ne permet pas de traiter un nombre plus grand de classes (si on souhaitait faire la même chose avec les lettres manuscrites, on passerait à 26 classes, donc

26 ! possibilités) ni de deviner les utilisateurs (beaucoup plus de classes, un nombre de classes inconnu, composé d'utilisateurs tous différents).

Les travaux présentés dans cette section II.2.1 ont fait l'objet d'une publication [72] dans le workshop "*International Workshop on Security in Machine Learning and its Applications (SiMLA)*", satellite de la conférence "*International Conference on Applied Cryptography and Network Security (ACNS)*", lors de sa 18e édition, en 2020.

II.2.1.3 L'Analyse Linéaire Discriminante pour l'alignement

Après avoir expérimenté avec différentes méthodes d'alignement, nous avons constaté qu'appliquer une LDA (Analyse Linéaire Disciminante) à la place d'une PCA sur les clusters permettait d'obtenir un meilleur étiquetage.

II.2.1.3 - A L'Analyse Linéaire Discriminante *Linear Discriminant Analysis* en anglais, abrégée LDA [80], est une technique de réduction dimensionnelle supervisée. Elle est utilisée sur un ensemble de vecteurs en $N \in \mathbb{N}^*$ dimensions étiquetés comme appartenant à $P \in \mathbb{N}^*$ catégories ($P \leq N$). Si on connaît l'étiquette associée à chacun des vecteurs, alors la LDA va projeter linéairement l'ensemble de vecteurs dans un espace en $P - 1$ dimensions maximisant la distance entre les différents groupes de vecteurs.

II.2.1.3 - B Utilisation sur les embeddings de chiffres manuscrits Dans notre cas, on utilisera la LDA avec les ensembles $\mathcal{E}_{attack}$ et $\mathcal{E}_{target}$ et les valeurs des chiffres associés comme étiquettes. Pour le jeu $\mathcal{E}_{attack}$, on connaît déjà la valeur des chiffres, donc on peut l'utiliser pour la LDA, ce qui nous donne donc un ensemble de vecteurs en 9 dimensions (10 chiffres, donc 10 catégories, on les projette donc dans un espace en 9 dimensions). Pour le jeu $\mathcal{E}_{target}$, on ne connaît pas les valeurs des chiffres, mais nous avons le résultat du clustering, qui a déjà montré son efficacité dans l'expérience II.2.1.1. On utilisera donc les valeurs attribuées par le clustering, qui bien que ne correspondant pas au chiffre réel, permettront au moins de séparer les clusters appartenant à des chiffres différents.

II.2.1.3 - C Résultats En utilisant la LDA à la place de la PCA dans la méthode présentée à l'expérience II.2.1.2, on a mesuré qu'en sortie de l'algorithme génétique, la première combinaison trouvée était la bonne dans 100% des cas. Ce qui signifie que nous

pouvons donc étiqueter des embeddings de chiffres de manière non supervisée sans avoir besoin d'un affinage, qui était une opération longue à réaliser.

Cette découverte n'a pas été utilisée ni publiée avant les travaux présentés à la section IV.1.

II.2.2 L'attaque des embeddings de chiffres : retrouver les scripteurs

Si on arrive à retrouver les chiffres associés aux clusters, on ne peut pas retrouver les scripteurs avec la même méthode, pour plusieurs raisons :

- Il faut connaître le nombre de scripteurs dans l'ensemble d'enrôlement, ou le déduire de manière non supervisée à partir du jeu d'embeddings.
- Si on a le nombre de scripteurs ou une méthode de clustering qui ne le nécessite pas, il faut encore que le clustering des scripteurs soit précis.
- Si on a un clustering précis, notre méthode d'alignement rencontre 2 difficultés :
    1. Sa complexité dépend exponentiellement du nombre de classes à aligner, pour un nombre trop grand de scripteurs elle ne fonctionnera pas ou dans un temps trop long.
    2. L'alignement se fait avec un autre ensemble présentant les mêmes classes, on saura donc à qui appartient quel cluster d'embeddings uniquement à condition de déjà posséder un ensemble d'embeddings de chaque utilisateur déjà identifié. L'attaque n'aurait donc pas lieu d'être.

Pour illustrer la répartition de l'ensemble d'embeddings vis-à-vis des scripteurs, nous avons décidé de reprendre le jeu d'embeddings $\mathcal{E}_{target}$ utilisé à la sous-section II.2.1.2. On peut voir dans la figure II.5 la répartition des embeddings en fonction du scripteur, projeté en 2 dimensions grâce à une analyse en composantes principales.

II.2.2.1 Vers la reconstruction de chiffres manuscrits

On a vu que trouver les alignements de clusters avait ses limites, notamment pour le vol d'identité des utilisateurs. Une solution que nous proposons pour obtenir plus d'informations sur les scripteurs est de reconstruire les données originales (tracés de chiffres manuscrits) à partir de leurs embeddings pour voir la quantité d'information sur l'utilisateur présente.

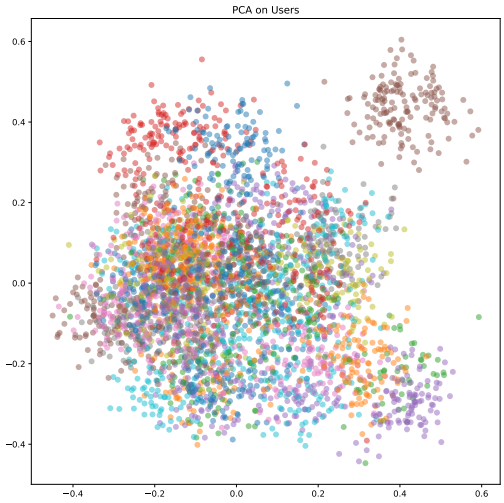

FIGURE II.5 – Graphique montrant les embeddings de $\mathcal{E}_{target}$. Différentes couleurs représentent différents scripteurs. (94 scripteurs au total)

Si les tracés sont réalistes et contiennent une bonne information sur le scripteur, alors on pourra affirmer d'un coup que les embeddings contiennent une information discriminante sur les scripteurs, et que cette information est reconstructible efficacement. Nous verrons la reconstruction plus en détail dans le chapitre III.

II.2.3 L'attaque des embeddings de parole

Nous savons qu'un extrait de parole contient au moins des informations discriminant les locuteurs et une information linguistique. Les systèmes de vérification du locuteur sont entraînés pour maximiser la présence de ces informations discriminantes dans les embeddings de parole, aussi appelés *x-vectors*. En se plaçant du point de vue d'un attaquant, nous avons cherché à retrouver un maximum d'informations à partir de ces x-vectors.

II.2.3.1 Retrouver de l'information linguistique

Pour commencer, nous nous sommes intéressés aux informations indépendantes du locuteur, comme les informations linguistiques.

RAJ, SNYDER, POVEY et al. [69] expliquent que les x-vectors (produit par un réseau x-vector [19] entraîné sur le jeu de données RedDot [81]) contiennent tout de même une

information sur le contenu linguistique, l'encodage ou la longueur du segment de parole à leur origine. Si cette étude montre la présence de ces contenus dans les x-vectors, nous ne pouvons pas pour autant essayer la méthode utilisée dans l'expérience n°II.2.1.2. En effet, comme nous l'avons expliqué au paragraphe II.2.1, celle-ci repose sur un nombre de classes défini,connu et assez faible (10 chiffres), or le nombre de mots du dictionnaire anglais le dépasse largement, ce qui rendra la méthode présentée inefficace.

Nous avons donc décidé de passer directement à l'extraction d'informations sur les locuteurs.

II.2.3.2 Retrouver des informations sur les locuteurs

Notre objectif dans l'analyse des embeddings de parole a donc été de retrouver toute information possible sur les locuteurs. En revanche, comme nous l'avons expliqué au paragraphe II.2.2, utiliser une méthode d'alignement pour retrouver à quel utilisateur correspond quel embedding ne donnera que des résultats limités, Nous avons donc décidé de passer directement à la reconstruction d'extraits de parole à partir des embeddings locuteurs, à voir dans le chapitre III.

II.3 Conclusion du chapitre

Dans ce chapitre, nous avons présenté et exécuté plusieurs attaques exploitant le vol d'un ensemble d'embeddings de tracés de chiffres manuscrits. Dans le cas de chiffres manuscrits, il est possible d'inférer les chiffres liés aux embeddings, mais pas de connaître les utilisateurs sans plus d'informations. Nous allons donc chercher dans le prochain chapitre à reconstruire directement les données biométriques à partir de leurs embeddings, pour pouvoir frauder un système d'authentification en utilisant ces données reconstruites.

Dans le cas du traitement de la parole pour la vérification du locuteur, il n'est pas possible d'inférer les mots prononcés en utilisant la même technique que celle utilisée pour déduire les chiffres, car la variété des mots prononçables dans un extrait de parole de 2 secondes est beaucoup trop importante. Les premières attaques sur les systèmes de vérification automatique du locuteur ne seront donc présentées qu'à partir du chapitre suivant.

Chapitre III

La reconstruction de données

Dans le chapitre précédent, nous avons vu qu'une donnée biométrique était une donnée personnelle, devant être protégée des vols par définition, car pouvant contenir beaucoup d'informations privées de différentes natures. Ces informations peuvent être présentes dans les *embeddings* produits par les extracteurs de caractéristiques dans le cadre d'un système d'authentification biométrique. Nous avons également vu qu'en utilisant un alignement entre plusieurs jeux d'*embeddings*, on pouvait retrouver des informations sur un jeu non étiqueté, comme la valeur des chiffres associés aux embeddings dans le cadre de l'authentification par analyse de tracés de chiffres manuscrits. En revanche cette méthode trouve ses limites sur les attaques visant à retrouver des informations sur les utilisateurs : il n'est pas possible de retrouver les utilisateurs sans connaître préalablement la liste des utilisateurs, ce qui limite fortement les possibilités d'actions.

Une solution possible face au manque d'information sur les utilisateurs est de passer par la reconstruction de leurs données personnelles. Si la donnée d'entrée peut être reconstruite fidèlement à partir de son *embedding*, alors celui-ci peut être considéré comme aussi sensible que la donnée en question, et elle peut être utilisée pour frauder un système d'authentification. Ce chapitre va étudier les différentes techniques de reconstruction de la parole et des chiffres manuscrits, et leur utilisation pour la fraude de systèmes d'authentification biométriques, du point de vue de l'attaquant. La première partie présente des architectures connues pour la reconstruction de données, puis nous nous focalisons sur les LSTM pour la reconstruction de chiffres (déjà présentés partie I.4.3.2) et le système AutoVC pour la reconstruction de la parole.

III.1 Les architectures connues pour la reconstruction

Dans un système d'authentification, un extracteur de caractéristiques, qu'il soit à base de réseaux de neurones ou non, est conçu pour encoder la donnée d'un utilisateur dans un vecteur, avec les propriétés suivantes :

- Les vecteurs d'un même utilisateur doivent être similaires
- Les vecteurs d'utilisateurs différents doivent être dissimilaires

La similarité entre deux vecteurs étant mesurée par un score ou une distance. Tous les extracteurs n'étant pas inversibles, la reconstruction de la donnée d'entrée à partir des embeddings de sortie devra se faire via un autre système. Dans cette section, nous allons voir des solutions classiques utilisées pour la reconstruction et la génération de données.

III.1.1 Les Auto Encodeurs

Une solution pour faciliter le décodage d'une donnée, est d'inscrire cet objectif dans l'entraînement de son encodeur. C'est le principe de l'*Auto Encodeur* [82] : on entraîne conjointement l'encodeur et le décodeur pour faciliter la reconstruction. L'efficacité de la reconstruction en fonction de la taille du vecteur d'encodage permet de trouver la taille nécessaire pour enregistrer toute l'information contenue dans la donnée originale en limitant les pertes.

Une variante connue est l'Auto Encodeur Variationnel (VAE) [83], qui génère une distribution gaussienne multi-dimensionnelle (un vecteur de moyenne et un vecteur de variance) à la place d'un vecteur, ce qui permet grâce à de multiples tirages de générer de nouvelles données, et ainsi de caractériser l'espace latent des vecteurs compressés, mais également de générer de nouvelles données. Ces réseaux peuvent notamment être utilisés pour reconstruire efficacement des dessins [84].

Il est possible de s'inspirer des architectures de certains auto encodeurs pour avoir des solutions permettant de reconstruire une donnée à partir d'un vecteur de taille fixe.

III.1.2 Les Réseaux Adversariels Génératifs (GAN)

Pour aller plus loin dans la génération de nouvelles données, on peut utiliser des Réseaux Adversariels Génératifs [85], [86], aussi appelés *GAN*. Les GAN sont composés de réseaux "*Générateurs*" et "*Discriminateurs*". L'objectif des premiers est d'apprendre à générer une donnée respectant certaines caractéristiques à partir d'un vecteur en grande dimension, quelle que soit la structure de l'espace d'où sont tirés ces vecteurs (en général on utilise une répartition gaussienne dans un espace à D dimensions). On parle de réseaux adversariels car leur entraînement se fait en compétition face au deuxième, qui lui apprend à différencier les données synthétiques (produites par le générateur) des données réelles. Le générateur et le discriminateur sont entraînés conjointement jusqu'à ce que le générateur finisse par produire des données indistinguables des données réelles par le discriminateur.

Ces réseaux ont montré qu'ils pouvaient générer efficacement des données synthétiques de visage [87], [88] pouvant être utilisées pour des attaques de reconstruction de *templates* [14].

III.1.3 Les *Flows*

Les *Flows* [89] sont des systèmes composés uniquement d'éléments réversibles, permettant donc d'utiliser le réseau dans les deux sens. Leur utilisation permet donc de généraliser le passage d'une donnée vers un vecteur (et inversement) en assurant à la fois les fonctions d'encodeur et de décodeur, en créant une transformation bijective entre 2 espaces. De la même manière qu'il est utilisé pour faire correspondre un ensemble de données et un ensemble d'embeddings, il peut également être utilisé pour aligner les distributions de deux ensembles d'embeddings. L'utilisation d'opérations réversibles dans la construction de ces systèmes permet de cartographier l'espace latent des vecteurs encodés, et donc de l'étudier et d'y opérer des transformations beaucoup plus facilement.

La figure III.1 résume schématiquement les différentes architectures présentées ci-dessus. Les encodeurs y sont représentés en **violet** et les décodeurs en **jaune**. Les flows pouvant être vus comme l'alignement entre l'espace des **données** et celui des **embeddings**, ils sont représentés en **vert**.

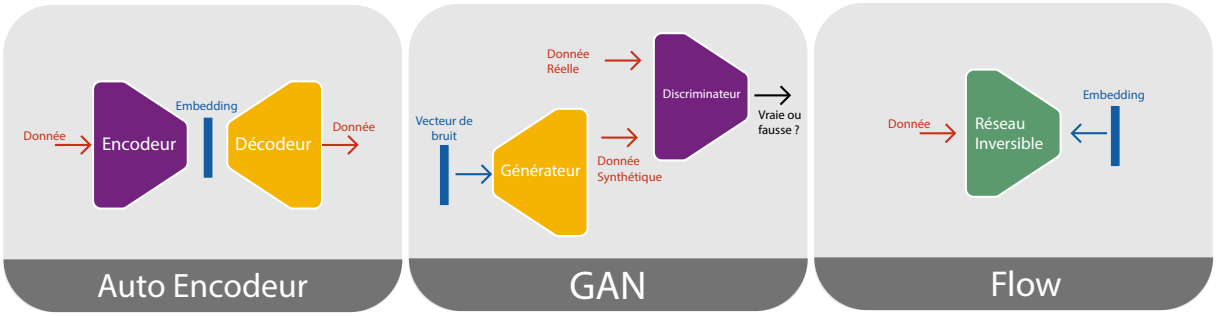

FIGURE III.1 – Schéma simplifié des architectures pour la reconstruction et la génération

III.2 La reconstruction de tracés de chiffres

Comme expliqué dans la section II.2.1, pour aller plus loin dans la récolte d'informations sur les scripteurs à partir des embeddings, nous devons passer par la reconstruction des données originales, c'est-à-dire des tracés de chiffres manuscrits.

Il s'agit de passer d'embeddings vers une séquence de points en 2 dimensions de longueur variable et inconnue. L'information est donc peu compressée.

III.2.1 Les LSTMs pour la reconstruction de chiffres

Les embeddings des scripteurs sont générés à partir de réseaux de neurones récurrents appelés LSTM [59], présentés section I.4.3.2. La première idée a donc été d'utiliser des réseaux de même architecture comme décodeurs, comme dans [90]. Nous avons donc essayé de reconstruire les représentations produites par un encodeur LSTM avec un décodeur LSTM, entraîné à postériori.

III.2.1.1 Reconstruction de chiffres à l'aide d'un LSTM

L'objectif de cette première expérience de reconstruction est de mesurer l'efficacité de la reconstruction de chiffres manuscrits à l'aide d'un décodeur LSTM. Les chiffres ont une valeur connue et sont tracés par un scripteur défini, l'efficacité est ainsi définie selon les deux critères suivants :

1. Le chiffre reconstruit est détecté à sa bonne valeur par l'extracteur original
2. Le chiffre reconstruit est détecté comme appartenant au bon scripteur par l'extracteur original, c'est-à-dire que ses embeddings sont assez proches des embeddings produits à partir des tracés originaux.

III.2.1.1 - A Le protocole Pour commencer, nous avons séparé $\mathcal{D}_{numbers}$ (présenté section I.2.1) en 2 jeux composés d'utilisateurs disjoints ($\mathcal{D}^{test}$ et $\mathcal{D}^{train}$, contenant chacun la moitié des utilisateurs).

Un réseau BiLSTM (qu'on appellera *encodeur*) avec une couche cachée de dimension 256 a été entraîné sur $\mathcal{D}^{train}$ à reconnaître les chiffres (avec une précision de 96,83% sur $\mathcal{D}^{test}$) et les scripteurs (avec un EER de 12,72% sur $\mathcal{D}^{test}$). L'encodeur est entraîné comme dans [23].

La fonction qui produit un embedding à partir d'une donnée sera appelée enc_{emb} et celle qui prédit le chiffre à partir d'une donnée enc_{num}. L'ensemble des embeddings produits en passant $\mathcal{D}^{train}$ (respectivement $\mathcal{D}^{test}$) dans l'encodeur sera appelé $\mathcal{E}^{train}$ (respectivement $\mathcal{E}^{test}$). Ensuite nous entraînons également un réseau LSTM (qu'on appellera *décodeur*) sur $\mathcal{D}^{train}$ et $\mathcal{E}^{train}$, pour produire des séquences de points en 3 dimensions, de longueur maximum 256 (la longueur maximum des chiffres en entrée étant de 254, nous avons donc choisi une puissance de 2 qui soit proche et supérieure) à partir de vecteurs de dimension 256. Les deux premières dimensions servent à prédire la position des points, la troisième à déterminer la fin de la séquence. Une *sigmoïde* est appliquée sur cette troisième dimension, et le réseau doit produire un 1 lorsqu'il doit s'arrêter, et des 0 sinon. On a alors un vecteur décrivant la longueur prédite en *onehot*. Le décodeur est entraîné avec l'optimiseur Adam [68], en utilisant 2 fonctions de coûts :

- $Loss_{len}$ pour la prédiction de la longueur des tracés. Soit L la longueur réelle et $\widehat{L}$ la longueur prédite , alors on défini $Loss_{len} : L, \widehat{L} \mapsto (onehot(L) - onehot(\widehat{L}))^2$.
- $Loss_{trc}$ pour la prédiction des tracés. Soit $x = (x_i)_{1 \geq i \geq L}$ le tracé réel et $\widehat{x} = (\widehat{x_i})_{1 \geq i \geq \widehat{L}}$ le tracé prédit, alors on défini $Loss_{trc} : L, x, \widehat{x} \mapsto \frac{1}{L} \log \sum_{i=1}^{L} \exp(x_i - \widehat{x_i})$

Après entraînement, on va tester notre système avec les embeddings de $\mathcal{E}^{test}$. Les données reconstruites par le décodeur seront donc appelées $\widehat{\mathcal{D}^{test}}$, les embeddings recalculés par l'encodeur $\widehat{\mathcal{E}^{test}}$, et la fonction qui reconstruit une donnée à partir d'un embedding : *dec*. Le fonctionnement de l'expérience est explicité dans la figure III.2.

III.2.1.1 - B Les résultats Les performances sont évaluées sur les données reconstruites à partir de $\mathcal{D}^{test}$, grâce à trois métriques expliquées section I.3.

- L'*Accuracy* sur les chiffres va servir à mesurer si les données reconstruites permettent de retrouver la valeur du chiffre.
- L'*EER* par rapport aux utilisateurs va permettre de savoir si les données reconstruites conservent la même discriminabilité entre utilisateurs.

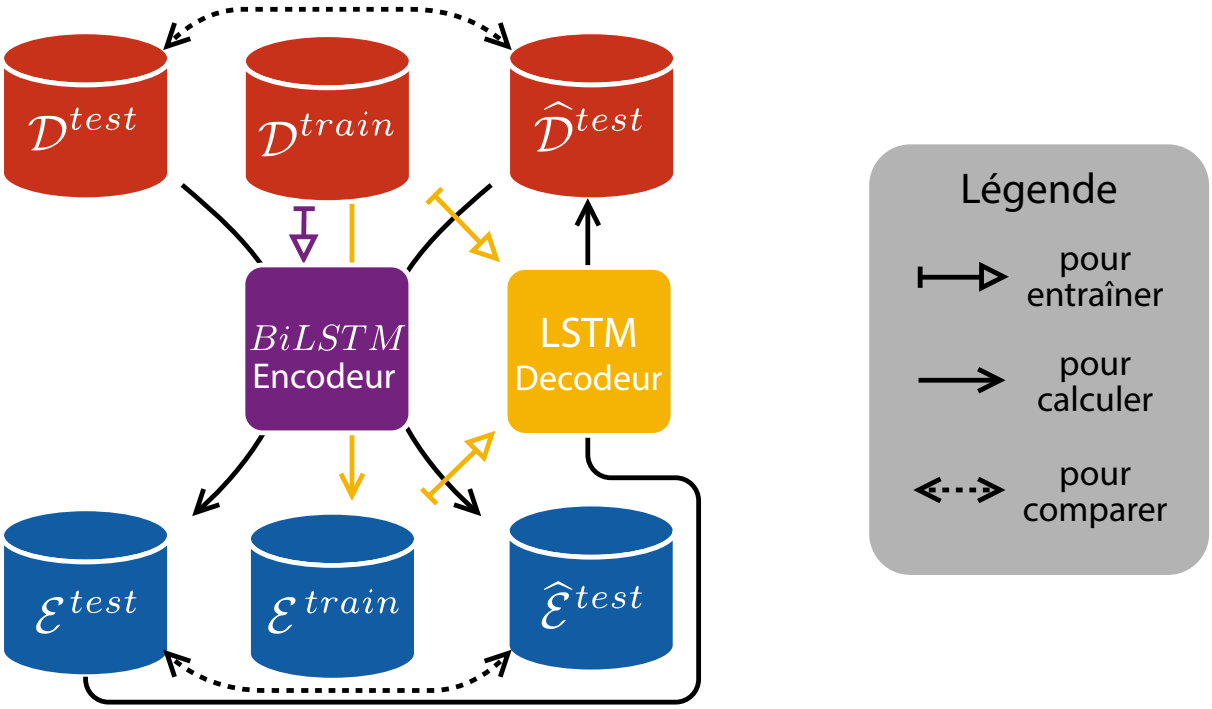

FIGURE III.2 – Schéma résumant l'expérience III.2.1.1.

- La *SFAR* comparant les données reconstruites aux données $\widehat{\mathcal{D}^{test}}$ va permettre de mesurer si l'attaquant peut frauder le système, en considérant des contraintes variables.

Les résultats sont présentés dans la table III.1 à partir de ces trois métriques, et comparés aux résultats des expériences III.2.2.1 et III.2.2.2. On remarque que l'attaque fonctionne assez bien tant que l'attaquant a un accès à l'encodeur, car on a un $sFAR_{EER}$ assez haut : 68,22%, même si celui-ci baisse drastiquement lorsqu'on réduit le seuil d'acceptation : le $sFAR_{1\%}$ descend à 33,41%.

Les chiffres reconstruits par ce décodeur LSTM sont représentés dans la figure III.3. Ceux-ci sont trop "ronds", les brusques variations sont mal modélisées : notre décodeur n'est donc peut-être pas assez bon, et les bonnes performances pourraient n'être valables que pour cet encodeur précis.

III.2.2 Les LSTM-MDN pour la reconstruction de chiffres

Face aux problèmes rencontrés sur l'apparence des chiffres avec le décodeur LSTM, nous avons décidé d'adapter son architecture en cherchant dans la bibliographie des références de réseaux pour la reconstruction de l'écriture manuscrite.

TABLE III.1 – Résultats des expériences III.2.1.1 et III.2.2.1, en terme d'*Accuracy* sur les chiffres, EER et de $SFAR$. Les encodeurs utilisés ont tous la même architecture, mais seul le décodeur de l'expérience III.2.2.1 est un LSTM-MDN.

Système	*Accuracy* ↑	EER ↓	$SFAR_{EER}$ ↑	$SFAR_{1\%}$ ↑	$SFAR_{0.1\%}$ ↑
Exp III.2.1.1					
Enc. LSTM	96,22%	12,72%	-	-	-
Déc. LSTM	84,79%	17,72%	95,76%	68,22%	33,41%
Exp. III.2.2.1					
Enc. LSTM	96,22%	12,72%	-	-	-
Déc. LSTM MDN	85,44%	17,71%	97,15%	75,20%	44,31%

[63] introduit l'utilisation de Mixture Density Networks pour la génération d'écriture manuscrite. Ces réseaux, présentés section I.4.3.4, semblent plus adaptés à la reconstruction de chiffres manuscrits.

III.2.2.1 Reconstruction de chiffres à l'aide d'un LSTM-MDN

L'objectif de cette expérience est le même que l'expérience III.2.1.1, mais nous utilisons ici un encodeur différent : un LSTM MDN.

III.2.2.1 - A Le protocole Le protocole est le même que pour l'expérience III.2.1.1, les mêmes données sont utilisées, seul le décodeur change. Le protocole est présenté schématiquement dans la figure III.4.

III.2.2.1 - B Les résultats Nous utilisons les mêmes métriques que dans l'expérience III.2.1.1, EER, Accuracy et SFAR, présentées dans la partie I.3. Les résultats sont présentés dans la table III.1, et ils sont légèrement meilleurs que pour le décodeur précédent :

- l'Accuracy sur la détection des chiffres monte à 85.44% (+0.65%)
- L'EER reste à 17.71%
- Le SFAR monte pour tous les seuils :
  - 97.15% pour un seuil à l'EER (+1.39%)
  - 44.31% pour un seuil de 0.1% (+10.90%)

Le décodeur LSTM-MDN fonctionne donc mieux que le LSTM.

On peut alors en déduire que, individuellement, les embeddings de chiffres manuscrits contiennent assez d'informations sur le scripteur pour être utilisés pour la fraude. Cette mesure a été faite avec un accès illimité en boite noire à l'encodeur (voir partie I.1.3.2). Il s'agit d'une attaque de reconstruction de gabarit similaire à des attaques déjà effectuées pour d'autres biométries, comme l'analyse d'images de visages dans [14].

III.2.2.2 Reconstruction de chiffres manuscrits sans accès à l'encodeur

Pour aller plus loin, nous avons cherché s'il était possible de reconstruire des embeddings de chiffres manuscrits sans avoir accès à l'encodeur qui les avait produit. Si l'attaquant simulé dans cette expérience n'a pas accès à l'encodeur, on suppose à priori qu'il sait attaquer un système de reconnaissance du scripteur à partir de chiffres manuscrits. La littérature n'est pas très développée sur ce sujet, et le nombre d'architectures disponible est limité, toujours à base de LSTM ou BiLSTM. On pose donc que l'attaquant a une connaissance de l'architecture de l'encodeur attaqué. Pour attaquer cet encodeur cible, il devra donc construire et entraîner lui-même son encodeur d'attaque, avec des données différentes.

III.2.2.2 - A Le protocole Pour cette expérience, on utilise deux encodeurs de même architecture, entraînés sur des données d'utilisateurs disjoints. Le premier sera nommé *encodeur cible*, seul ses embeddings de sortie seront accessibles. Le deuxième sera nommé *encodeur d'attaque*, et lui sera accessible en boite blanche.

Pour commencer, le jeu $\mathcal{D}_{numbers}$ (présenté section I.2.1) est séparé en 4 jeux composés d'utilisateurs disjoints ($\mathcal{D}^{test}_{attack}$, $\mathcal{D}^{train}_{attack}$, $\mathcal{D}^{test}_{target}$ et $\mathcal{D}^{train}_{target}$, contenant chacun le quart des utilisateurs). Comme nous utilisons ici deux fois moins de données par rapport aux encodeurs des expériences III.2.1.1 et III.2.2.1, nous avons besoin d'un modèle capable de mieux généraliser (en gardant le même modèle, nous obtenions un EER de 28.45% sur le jeu de test). Les 2 encodeurs ont donc presque la même architecture que dans l'expérience III.2.1.1, et sont entraînés de la même manière, mais la dimension des embeddings passe de 256 à 1024, ce qui permettra d'inclure plus d'informations sur les tracés. L'*encodeur cible* est entraîné avec $\mathcal{D}^{train}_{target}$ et testé sur $\mathcal{D}^{test}_{target}$. De même pour l'*encodeur d'attaque*, mais respectivement avec les jeux $\mathcal{D}^{test}_{attack}$ et $\mathcal{D}^{train}_{attack}$. Les performances des deux encodeurs en précision et accuracy sont présentées dans la table III.1. Le jeu d'embeddings $\mathcal{E}^{test}_{attack}$ (respectivement $\mathcal{E}^{test}_{target}$) est calculé en utilisant l'*encodeur d'attaque* (resp. l'*encodeur cible*) avec le jeu de données $\mathcal{D}^{test}_{attack}$ (resp. $\mathcal{D}^{test}_{target}$)

Ensuite, un décodeur avec une architecture LSTM-MDN est entraîné comme dans l'expérience III.2.2.1, sur $\mathcal{E}^{test}_{attack}$ et $\mathcal{D}^{test}_{attack}$, jusqu'à obtenir 85.29% de précision sur la détection du chiffre reconstruit. Le protocole de l'expérience est explicitée dans la figure III.5.

III.2.2.2 - B Les résultats Pour évaluer le fonctionnement, on utilise le décodeur entraîné pour reconstruire les embeddings de $\mathcal{E}^{test}_{target}$, ce qui nous donnera un jeu de données reconstruites $\widehat{\mathcal{D}^{test}_{target}}$. C'est sur ces données reconstruites qu'on évalue la précision de l'attaque sur les scripteurs et les chiffres, en utilisant l'*encodeur cible*

Les résultats sur les encodeurs cible et source sont disponibles dans la table III.2 : Nous utilisons les mêmes métriques que dans l'expérience III.2.1.1 : EER, Accuracy et SFAR.

TABLE III.2 – Résultats de l'expérience III.2.2.2, en terme d'*Accuracy* sur les chiffres, EER et de $SFAR$. Les encodeurs utilisés ont tous la même architecture et le décodeur est un LSTM-MDN. Il est testé fonctionnant avec l'encodeur utilisé pour son entraînement (source), et avec l'encodeur visé par l'attaque (cible).

Système	$Accuracy \uparrow$	$EER \downarrow$	$SFAR_{EER} \uparrow$	$SFAR_{1\%} \uparrow$	$SFAR_{0.1\%} \uparrow$
Exp. III.2.2.2					
Enc. cible LSTM	94,64%	14,13%	-	-	-
Enc. source LSTM	94.76%	13.21%	-	-	-
Déc. LSTM MDN (sur l'enc. source)	78,51%	13,53%	95,62%	67,88%	34,55%
Déc. LSTM MDN (sur l'enc. cible)	9,45%	39.01%	1,04%	0,02%	0%

Elles sont présentées dans la partie I.3. Les résultats évalués sur l'encodeur qu'on attaque sont cependant très variables :

- l'Accuracy sur la détection des chiffres est à 9.45% (au niveau de l'aléatoire, sur 10 chiffres)
- L'EER est à 13.05%
- Le SFAR est proche de 0 pour tous les seuils.

Si le décodeur conserve bien une information concernant le scripteur, les chiffres reconstruits ne sont pas du tout les bons, et les embeddings issus de ses chiffres ne correspondent

pas du tout à ceux utilisés en entrée, ce qui fait que l'attaque échoue presque systématiquement. Deux théories sont alors possibles :

1. Soit le décodeur ne généralise pas, il n'effectue qu'une attaque adversarielle sur l'encodeur d'attaque, non transposable à d'autres encodeurs.
2. Soit le décodeur généralise, mais les embeddings produits par un autre encodeur ne sont pas dans le même espace vectoriel, ou *domaine*.

Au vu de l'échec de la reconstruction sans alignement préalable, nous allons devoir trouver une fonction qui rapproche le plus possible les embeddings produits par l'encodeur cible de l'espace de ceux produits par l'encodeur d'attaque. La recherche de cette **fonction d'alignement** sera explorée dans le chapitre IV : Alignements.

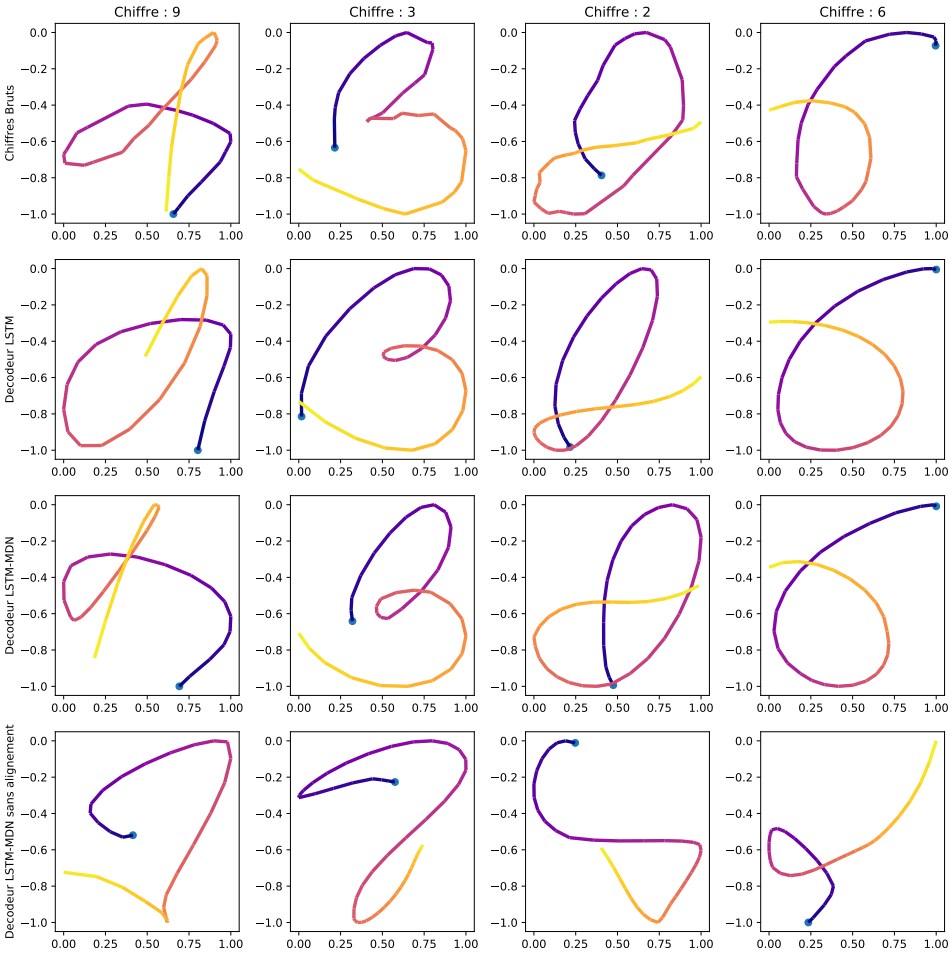

FIGURE III.3 – Tracés de chiffres manuscrits. La première ligne de chiffres sont les tracés originaux, la deuxième les tracés reconstruits par un LSTM (expérience III.2.1.1), la troisième par un LSTM-MDN (expérience III.2.2.1) et la quatrième par un LSTM-MDN n'ayant pas accès à l'encodeur (expérience III.2.2.2). Le point bleu marque le début du tracé et le dégradé du violet au jaune permet de percevoir l'ordre du tracé.

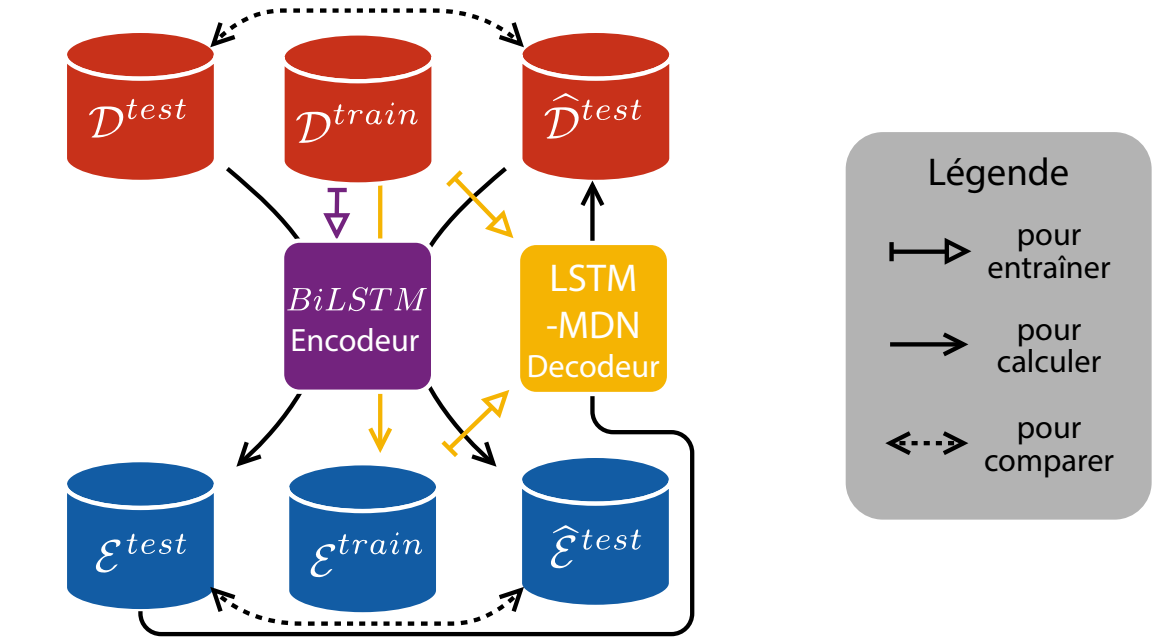

FIGURE III.4 – Schéma résumant l'expérience III.2.2.1.

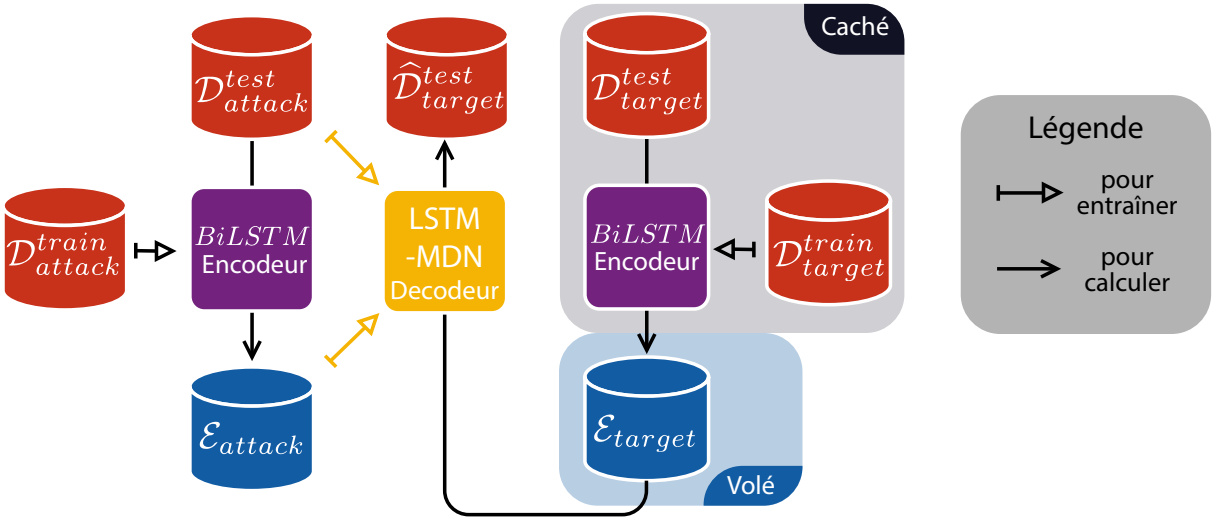

FIGURE III.5 – Schéma résumant l'expérience III.2.2.2.

III.3 La reconstruction d'extraits de voix

Suite à la section II.2.3, nous avons décidé de nous intéresser directement à la reconstruction de la voix. La première interrogation a été de savoir s'il était possible de reconstruire de la parole uniquement à partir d'un embedding de parole, aussi appelé *x-vector*.

Comme nous l'avons vu précédemment, la parole est un signal complexe, qui peut contenir des informations variées. En l'occurrence, nous savons que la parole naturelle contient à la fois des informations discriminantes du locuteur et des informations linguistiques. Les x-vectors sont des vecteurs de caractéristiques produits par un extracteur dont l'objectif est d'ignorer la partie linguistique pour extraire un maximum d'informations sur le locuteur. Si on suppose que la partie linguistique est indépendante de la partie locuteur et que l'extracteur utilisé est parfait, alors les x-vectors ne contiennent aucune information linguistique, et il est donc impossible de reconstruire autre chose que du bruit adversariel à partir de ceux-ci.

Dans le cas où l'information contenue dans les x-vectors ne suffirait pas, nous avons donc exploré des techniques de *conversion de voix* (voir section III.3.1) pour modifier un extrait de parole de manière à la faire prononcer par un locuteur cible. Les solutions explorant ces techniques seront présentées section III.3.2.

III.3.1 La Conversion de voix

La conversion de voix [91], [92] est un ensemble de techniques visant à transformer un extrait de parole pour donner l'impression qu'il est prononcé par un autre locuteur. La figure III.6 montre un schéma du fonctionnement d'un système de conversion de voix. L'extracteur linguistique peut être un simple encodeur, un système de reconnaissance de la parole [57] ou un extracteur plus complexe séparant les informations d'une manière plus complexe [93]. L'extracteur de caractéristiques locuteurs est la plupart du temps un système de vérification de locuteur (voir section I.4.2).

III.3.1.1 Les différents systèmes de conversion de voix

Il existe plusieurs propriétés caractéristiques des système de conversion de voix : Si un système est entraîné pour un seul locuteur source vers un seul locuteur cible, on parle de système *one-to-one* [94]. Dans le cas de plusieurs locuteurs sources (resp. plusieurs

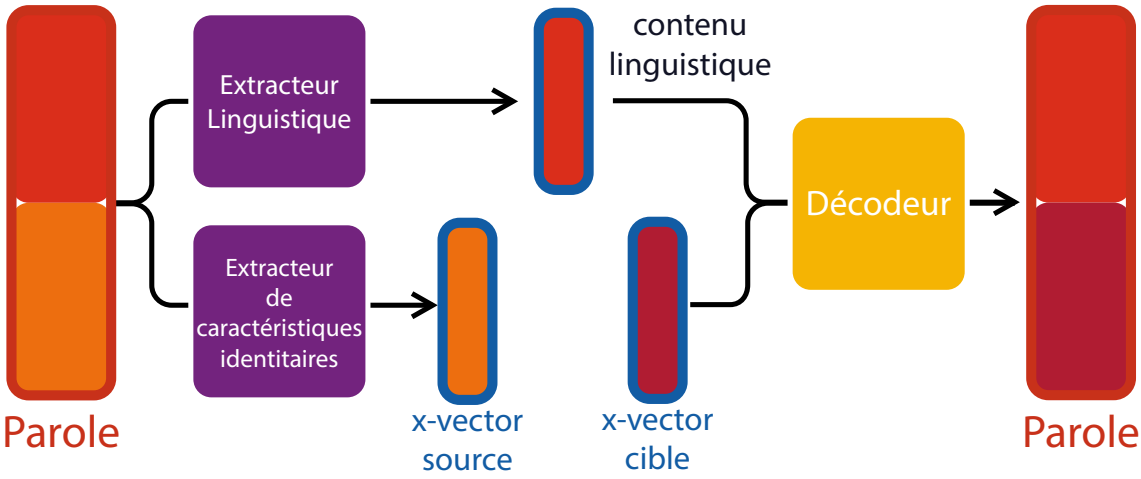

FIGURE III.6 – Schéma du fonctionnement d'un système de conversion de voix

locuteurs cibles), on parle de système *many-to-one* [95], [96] (resp. *one-to-many* [96]). Le cas qui nous intéresse le plus ici est celui des systèmes *many-to-many* [85], [97]. Plus spécifiquement, dans le cadre d'attaques sur des locuteurs inconnus, nous avons cherché des systèmes dits "*Zero-shot*" [93], [98], [99], c'est-à-dire des systèmes fonctionnant pour des locuteurs cible jamais vus lors de l'entraînement du réseau. Lors de cette thèse, le premier système *many-to-many zero shot* disponible était AutoVC [99]. D'autres systèmes sont parus par la suite [97], mais nous sommes restés sur AutoVC ou des dérivées de son architecture par cohérence avec les expériences précédemment réalisées.

III.3.1.2 AutoVC

[99] est un système de conversion de voix *many-to-many zero shot* qui utilise comme encodeur linguistique un réseau convolutionnel, et comme encodeur d'identité un réseau de vérification du locuteur x-vector [19]. Ce système prend en entrée des Mel-Spectrogrammes (voir section I.4.1.1) ayant D_T dimensions temporelles et D_S dimensions spectrales, et reconstruit des Mel-Spectrogrammes ayant le même format.

Ce système est originalement entraîné avec une unique fonction de coût, utilisant la technique dite de la "*copy synthesis*" : lors de l'entraînement, le système est utilisé avec le même locuteur source et cible, et on cherche à minimiser la distance entre la parole d'origine et la parole reconstruite. Les compositions de l'encodeur linguistique et du décodeur sont présentées respectivement dans les colonnes I et II de la table III.3.

Dans le système original, la dimension de l'embedding de parole $D_P \in \mathbb{N}^*$ est fixée à 256

et la dimension de l'embedding linguistique $D_L \in \mathbb{N}^*$ est fixée à 16. D_L est aussi appelé goulot d'étranglement, ou *bottleneck*, car c'est cette dimension qui filtre l'information locuteur pour ne laisser passer que l'information linguistique, ce qui oblige le décodeur à se servir de l'embedding cible pour la reconstruction. Si le bottleneck est trop grand, alors les vecteurs linguistiques pourront contenir également de l'information du locuteur à la source, et s'il est trop petit, l'information linguistique subira trop de pertes pour pouvoir reconstruire un segment de parole réaliste.

La fonction de coût qui mesure la différence entre la parole reconstruite et la parole source est appelée $\mathcal{L}_{copy}$. Cette différence est mesurée avec l'erreur moyenne au carré entre les deux éléments, aussi appelée *MSE*.

Lors de l'entraînement, on lui ajoute également une dernière partie : un réseau convolutionnel qui va retraiter la parole après sa reconstruction, sans en changer la taille. Son architecture est présentée dans la colonne III de la table III.3. L'ajout de ce réseau à posteriori, dit *PostNet*, permet d'augmenter le nombre de paramètres pendant l'apprentissage et ainsi rendre celui-ci plus facile. L'utilisation de ce réseau provoque l'ajout d'une fonction de coût supplémentaire $\mathcal{L}_{pn}$ pour s'assurer que les sorties et les entrées du PostNet soient égales, pour pouvoir ne plus l'utiliser après l'entraînement. L'équilibre entre les deux fonctions de coût est assuré par le scalaire $\lambda_{pn} \in \mathbb{R}^+$, fixé à 1 initialement. On a ainsi la fonction de coût globale définie comme :

$$\mathcal{L}_{tot} = \mathcal{L}_{copy} + \lambda_{pn} \times \mathcal{L}_{pn} \tag{III.1}$$

Un schéma clarifiant son entraînement et sa composition est présenté dans la figure III.7.

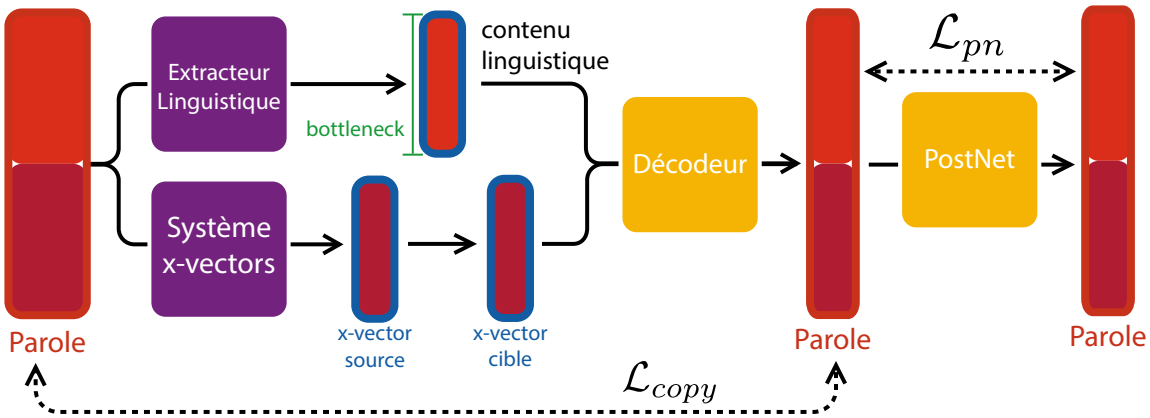

FIGURE III.7 – Entraînement et composition d'AutoVC.

TABLE III.3 – Tableau présentant l'architecture de l'encodeur linguistique, du décodeur et du PostNet utilisés dans AutoVC [99]. Les dimensions sont présentées pour une dimension de l'embedding de parole $D_P \in \mathbb{N}^*$ et une dimension de l'embedding linguistique $D_L \in \mathbb{N}^*$ non connues.

	I	II	III
Réseau	Encodeur Linguistique	Décodeur	PostNet
Entrée	Mel-Spect : $D_T \times D_S$ Emb. de parole : D_P	Emb. linguistiques : $D_T \times D_L$ Emb. de parole cible : D_P	Mel-Spect : $D_T \times D_S$
	Concaténation : $D_T \times (D_S + D_P)$	Concaténation : $D_T \times (D_L + D_P)$	
		LSTM $\mapsto D_T \times 512$	
	Conv $(D_S + D_P) \times 512$, 5 stride 1, padding 2	Conv 512×512, 5 stride 1, padding 2	Conv 80×512, 5 stride 1, padding 2
	Conv 512×512, 5 stride 1, padding 2	Conv 512×512, 5 stride 1, padding 2	Conv 512×512, 5 stride 1, padding 2
	Conv 512×512, 5 stride 1, padding 2	Conv 512×512, 5 stride 1, padding 2	Conv 512×80, 5 stride 1, padding 2
	BiLSTM $\mapsto D_T \times D_L$	BiLSTM $\mapsto D_T \times 1024$ Linear $1024 \times D_S$	
Sortie	$D_T \times D_L$	$D_T \times D_S$	$D_T \times D_S$

C'est ce système qui servira de base pour la conversion de voix au cours de cette thèse : le système de vérification du locuteur pourra être modifié, et d'autres fonctions de coût ajoutées.

III.3.2 La Conversion de voix pour la fraude

Dans cette section, nous nous plaçons du point de vue d'un attaquant désirant frauder un système de vérification du locuteur en utilisant uniquement des embeddings de parole. Pour frauder le système, l'attaquant essaye de reconstruire des extraits de parole tels que prononcés par les locuteurs ayant produit les embeddings de parole. Pour cela, on utilisera un système de conversion de voix : AutoVC [99] pour reconstruire des segments de parole à partir d'embeddings et de segments de paroles produits par d'autres locuteurs.

TABLE III.4 – Tableau présentant les résultats des expériences III.3.2.1, III.3.2.2 et III.3.2.3, en terme d'EER_{cible}, EER_{source} et de SFAR. La valeur maximale d'un EER est de 50%, on cherche donc à avoir EER_{source} aussi proche de 50 que possible. Pour rappel, l'EER de l'encodeur utilisé dans ces expériences est de 2.31% sur $\mathcal{D}^{test}_{encoder}$. AutoVC "Base" fait référence au système AutoVC original, AutoVC "Optimisé" fait référence au système après optimisation des hyper-paramètres. Les fonctions de coût utilisées, ou *Losses*, sont décrites dans les sections III.3.1 et III.3.2.3.

Système	Losses	Données	EER_{cible}	EER_{source}	$SFAR_1$	$SFAR_{0.1}$
III.3.2.1						
AutoVC	$\mathcal{L}_{copy}/\mathcal{L}_{pn}$	$\mathcal{D}^{test}_{encoder}$	9.51%	45.50%	0.01%	0.00%
Base		$\mathcal{D}^{test}_{decoder}$	38.27%	26.23%	0.23%	0.08%
III.3.2.2						
AutoVC	$\mathcal{L}_{copy}/\mathcal{L}_{pn}$	$\mathcal{D}^{test}_{encoder}$	6.48%	37.24%	0.63%	0.00%
Optimisé		$\mathcal{D}^{test}_{decoder}$	23.50%	30.24%	11.27%	1.75%
III.3.2.3						
AutoVC	$\mathcal{L}_{copy}/\mathcal{L}_{pn}/\mathcal{L}_{rec}$	$\mathcal{D}^{test}_{encoder}$	1.10%	46.15%	60.07%	0.93%
Optimisé		$\mathcal{D}^{test}_{decoder}$	0.25%	49.12%	99.74%	99.04%

III.3.2.1 Reconstruction de parole avec AutoVC

Notre première expérience de reconstruction de parole avec AutoVC vise à mesurer les capacités de ce système pour la fraude d'un système de vérification du locuteur. L'expérience est schématisée dans la figure III.8.

III.3.2.1 - A Le protocole Soit $\mathcal{D}^{train}_{encoder}$ le jeu de données composé de segments de paroles de 2 secondes extraits de voxceleb2 [35] et $\mathcal{D}^{test}_{encoder}$ les données de VoxCeleb1 [34]. Soit $\mathcal{D}^{train}_{decoder}$ le jeu de données composé des 100 premiers locuteurs de VCTK [36] et $\mathcal{D}^{test}_{decoder}$ les données des 10 locuteurs restants. Pour plus de détails concernant ces données, voir section I.2. L'extracteur de caractéristiques locuteur utilisé sera un Fast-ResNet34 [51], entraîné avec $\mathcal{D}^{train}_{encoder}$ jusqu'à atteindre un EER de 2.31% sur $\mathcal{D}^{test}_{encoder}$.

Pour la reconstruction d'extraits de parole, nous avons utilisé l'architecture AutoVC mais avec le même encodeur (un Fast-ResNet34). Le décodeur est entraîné sur les données de $\mathcal{D}^{train}_{decoder}$ et leurs embeddings associés ($\mathcal{E}^{train}_{decoder}$), en utilisant l'extracteur entraîné en boite noire, jusqu'à stabilisation (500 époques).

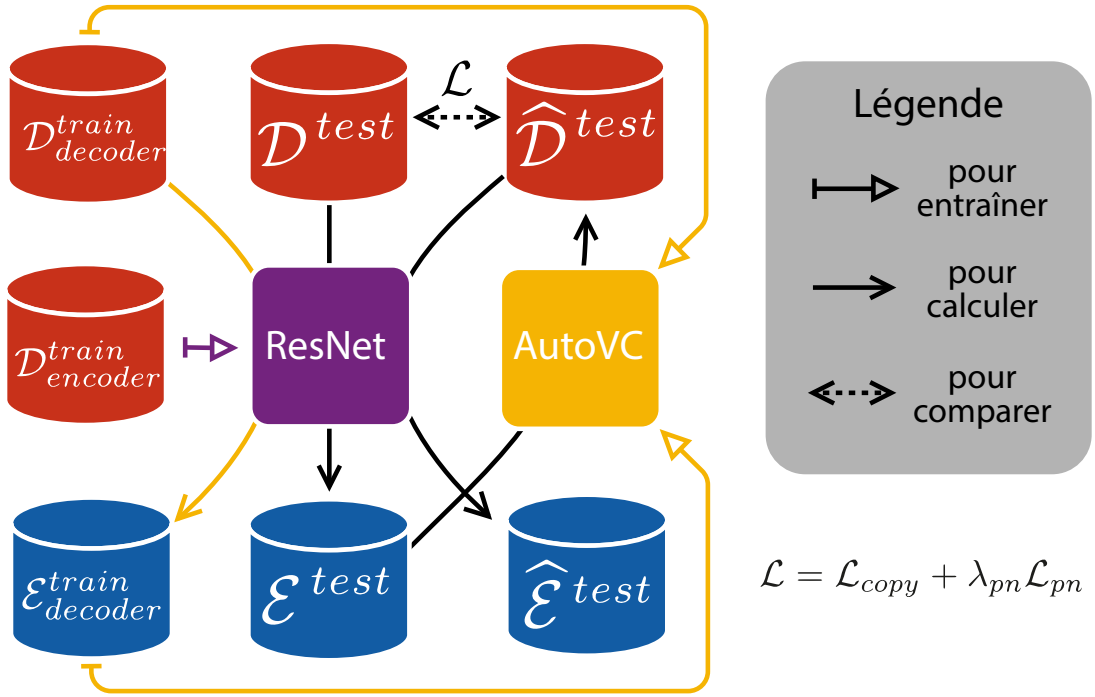

FIGURE III.8 – Schéma résumant l'expérience III.3.2.1. $\mathcal{D}^{test}$ pouvant faire référence à $\mathcal{D}^{test}_{decoder}$ ou $\mathcal{D}^{test}_{encoder}$ en fonction du jeu testé. Même chose pour $\mathcal{E}^{test}$."*Loss*" représente la fonction de coût utilisée pour l'entraînement d'AutoVC [99].

Pour commencer, nous avons suivi les paramètres d'entraînement indiqués dans [99], ce qui nous a donné une base de travail.

III.3.2.1 - B Les résultats Pour mesurer les performances du décodeur entraîné, on utilise un jeu de test $\mathcal{D}^{test}$. On calcule les embeddings représentatifs des locuteurs de $\mathcal{E}^{test}$ avec le fast-ResNet34. Ensuite on re-génère les extraits de voix à partir de ces embeddings et d'extraits de parole des 100 premiers locuteurs de VCTK à l'aide du décodeur entraîné, ce qui nous donne le jeu de données reconstruites $\widehat{\mathcal{D}^{test}}$ Enfin on recompile les embeddings reconstruits ($\widehat{\mathcal{E}^{test}}$) à partir des données reconstruites. Nous mesurons finalement l'EER du jeu $\widehat{\mathcal{E}^{test}}$ par rapport aux locuteurs cibles (du jeu $\mathcal{E}^{test}$) et aux locuteurs sources (du jeu $\mathcal{D}^{train}_{decoder}$), ce qui nous donne respectivement l'EER_{cible} et l'EER_{source}. On calcule également le *SFAR* (voir section I.3) pour mesurer à quel point la fraude a fonctionné, avec un seuil de fausses acceptations à 1% et à 0.1%. Les résultats sont présentés à la ligne 6.1 de la table III.4 en utilisant comme jeu de test soit $\mathcal{D}^{test}_{encoder}$ soit $\mathcal{D}^{test}_{decoder}$.

En examinant la table III.4, on remarque que les EER mesurés sont très hauts comparés à celui de l'encodeur (2.31%) et que le taux de fraudes réussies est quasiment nul.

L'attaque ne fonctionne pas.

III.3.2.2 Optimisation des hyper-paramètres d'AutoVC pour la fraude

Pour améliorer les résultats, nous avons commencé par affiner les paramètres d'entraînement d'AutoVC :

- La taille des batch est passée de 2 à 64.
- Le facteur λ_{pn} est passé de 1 à 0.05.
- Le *learning rate* est passé de 10^{-3} à 10^{-4}.

Les résultats pour ces paramètres optimaux sont présentés à la ligne 7 de la table III.4, et même si les résultats sont meilleurs, on a toujours une efficacité de l'attaque qui reste très limitée.

Face au manque de précision du système, nous avons décidé de le perfectionner spécifiquement pour la tâche de reconstruction pour la fraude, en lui permettant de s'entraîner avec une nouvelle fonction de coût conçue pour optimiser les embeddings produits à partir des extraits de parole reconstruits.

III.3.2.3 Reconstruction de parole avec une nouvelle fonction de coût

Dans cette expérience, nous avons essayé d'améliorer les performances d'AutoVC en ajoutant une nouvelle fonction de coût sur les embeddings reconstruits. Soit u et v deux utilisateurs. Soit d_u et d_v deux segments de paroles, produits respectivement par u et par v. On appellera e_{u,d_u} et e_{v,d_v} les embeddings calculés à partir de ces segments de parole par un extracteur de données entraîné. On utilise AutoVC pour reconstruire un nouvel extrait de parole $d_{u\to v}$ à partir de e_{v,d_v} et d_u, contenant le contenu linguistique de d_u prononcé par v. On compile l'embedding associé à cet extrait de parole reconstruit $e_{v,d_{u\to v}}$, qui est censé être similaire aux autres embeddings de v. La fonction de coût que nous utiliserons ici est la même que celle utilisée initialement pour comparer la parole reconstruite à la parole originale : la différence moyenne au carré entre l'embedding reconstruit et l'embedding original de v, aussi appelée *MSE*.

$$\mathcal{L}_{rec} = \|e_{v,d_{u\to v}} - e_{v,d_v}\|^2 \tag{III.2}$$

Une fois cette fonction pondérée par un scalaire λ_{rec}, la fonction de coût totale devient donc :

$$\mathcal{L}_{tot} = \mathcal{L}_{copy} + \lambda_{pn} \times \mathcal{L}_{pn} + \lambda_{rec} \times \mathcal{L}_{rec} \tag{III.3}$$

L'expérience est représentée schématiquement dans la figure III.9.

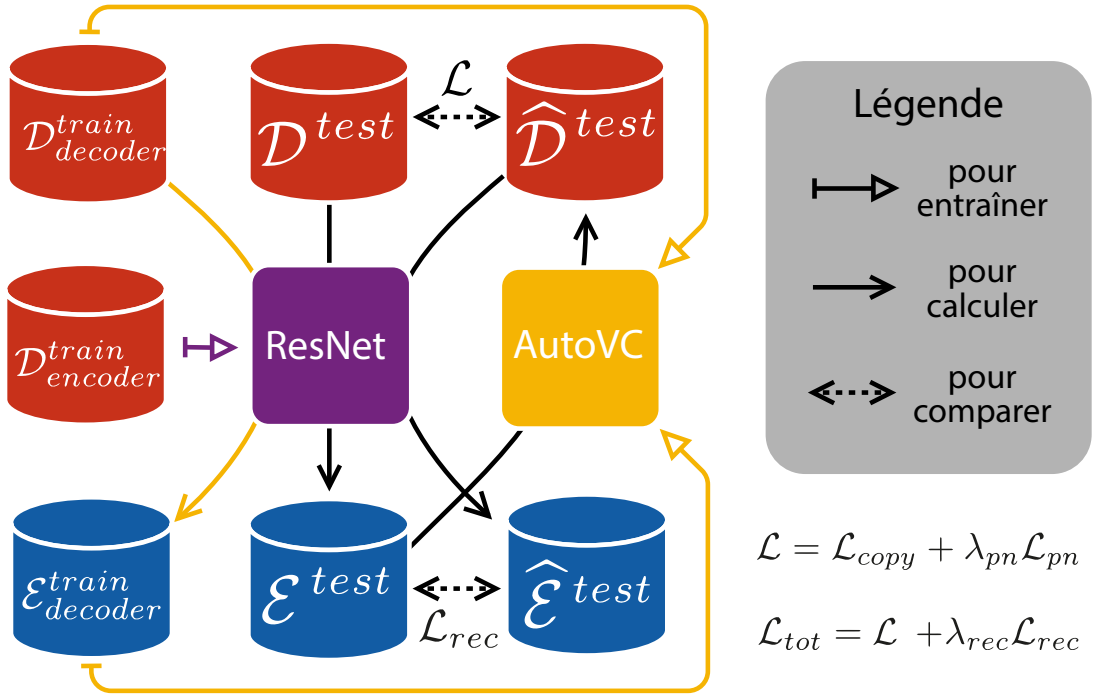

FIGURE III.9 – Schéma résumant l'expérience III.3.2.3. $\mathcal{D}^{test}$ pouvant faire référence à $\mathcal{D}^{test}_{decoder}$ ou $\mathcal{D}^{test}_{encoder}$ en fonction du jeu testé. Même chose pour $\mathcal{E}^{test}$. "*New Loss*" représente la nouvelle fonction de coût ajoutée pour cette expérience.

III.3.2.3 - A Le protocole On réutilisera les jeux de donnés tels que définis dans l'expérience III.3.2.2 : $\mathcal{D}^{train}_{encoder}$, $\mathcal{D}^{test}_{encoder}$, $\mathcal{D}^{train}_{decoder}$ et $\mathcal{D}^{test}_{decoder}$. Pour plus de détails concernant ces données, voir section I.2. L'extracteur et le décodeur sont les mêmes que dans l'expérience III.3.2.2, à la différence près que le décodeur a été entraîné avec la nouvelle fonction de coût présentée ci-dessus. Le coefficient λ_{rec} a été fixé à 2.

III.3.2.3 - B Les résultats Pour mesurer les performances du décodeur entraîné, on utilise comme jeu de test soit $\mathcal{D}^{test}_{encoder}$ soit $\mathcal{D}^{test}_{decoder}$. Les résultats sont présentés dans la table III.4 en utilisant les mêmes métriques que précédemment.

L'attaque semble très bien marcher pour des données du même domaine que celles ayant servies à entraîner le décoder : on fraude un système qui ne laisserait passer que

0.1% de faux positifs dans 99.04% des cas ! En revanche, les résultats sont beaucoup moins bons pour les données dites "hors domaine", c'est-à-dire des segments de parole n'ayant pas été extraits dans les mêmes conditions. Et nous avons également le problème de l'accès à l'encodeur : l'entraînement de notre décodeur nécessite un accès direct à l'encodeur, ce qui constitue une hypothèse très restrictive.

III.3.2.4 Mesure des performances pour des bottlenecks différents

Avant d'aller plus loin, nous avons tenu à vérifier que la dimension du bottleneck était bien optimale. Pour le vérifier, nous avons entraîné plusieurs systèmes avec différentes dimensions de bottleneck, allant de 0 à 128. Pour chaque dimension possible, nous avons ré-entraîné AutoVC en suivant l'expérience III.3.2.3. Lorsque nous reconstruisons un segment de parole avec une dimension linguistique nulle, cela revient à reconstruire un segment de parole uniquement à partir d'un x-vector, sans apport d'information linguistique. Pour chaque système entraîné, nous avons mesuré les EER par rapport aux locuteurs sources et cibles sur le jeu $\mathcal{D}^{test}_{decoder}$. Les résultats de l'expérience sont présentés dans la figure III.10.

En observant le graphique de la figure III.10, on commence par remarquer que le choix d'un bottleneck de 16 est bel et bien optimal, car c'est celui qui nous donne l'EER_{cible} le plus bas et l'EER_{source} le plus haut. Une situation souhaitée, car on veut que le système produise des segments de voix totalement indépendants du locuteur utilisé à la source (donc un EER le plus proche possible des 50%), et en même temps que ceux-ci soient prononcés comme le locuteur cible les aurait prononcés, donc induisant un EER le plus proche possible de 0 pour les cibles.

Quand on s'intéresse aux effets de bords, on remarque 2 éléments :

- Lorsque le bottleneck devient trop grand, la présence d'information sur les locuteurs sources devient de plus en plus marquée, jusqu'à remplacer toute l'information sur les locuteurs cibles, et inverse totalement les deux courbes.
- Lorsque le bottleneck devient trop petit, il y a moins d'information qui est transmise et la détection de l'information locuteur en est impactée : l'EER pour les locuteurs cible monte sans pour autant impacter l'EER des locuteurs sources.

Notre hypothèse est qu'avec un bottleneck trop petit, le vecteur censé contenir les informations de la source (locuteurs et linguistiques) ne contient plus assez d'information. Comme on a montré que l'information du locuteur a déjà presque disparu à partir du bot-

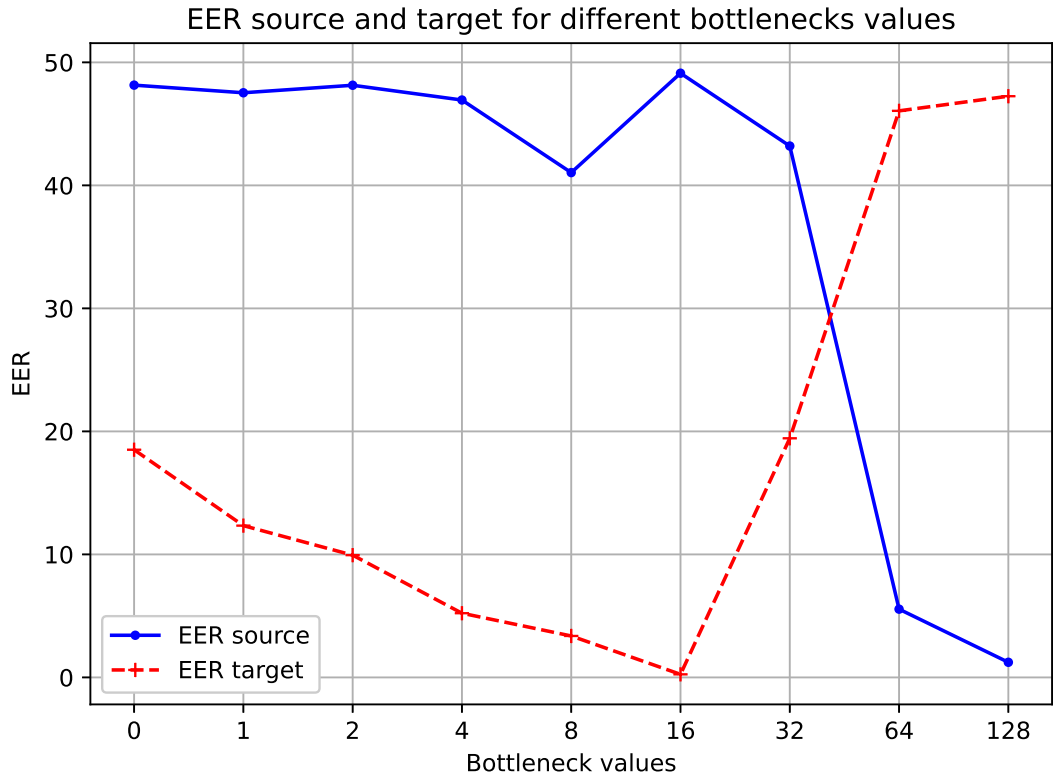

FIGURE III.10 – EER_{source} et EER_{cible} mesurés pour des systèmes avec différents bottlenecks lors de l'expérience III.3.2.4.

tleneck à 16 dimensions, alors l'information qui disparaît ici doit donc être l'information linguistique. La reconstruction de parole sans cette information linguistique apportée par la source donne des performances en vérification du locuteur beaucoup plus faibles.

On remarque également que pour un bottleneck nul, c'est-à-dire lorsqu'aucune information linguistique n'est utilisée pour la reconstruction, on a les plus mauvaises performances : un EER_{cible} à 19% et un EER_{source} à 49%. Cela semble évident, car détecter un locuteur dans un segment ne contenant pas de parole semble évidemment plus dur que dans un segment contenant de la parole.

Maintenant que nous avons confirmé que le bottleneck utilisé était bien optimal, nous allons poursuivre les expériences, en essayant de reproduire l'attaque présentée dans l'expérience III.3.2.3, mais sans accès à l'encodeur.

III.3.2.5 Reconstruction de parole sans accès à l'encodeur

Dans cette expérience, on se place du point de vue d'un attaquant qui voudrait reconstruire des extraits de paroles à partir d'un ensemble d'embeddings, mais sans avoir aucun accès à l'encodeur les ayant produits. Pour ce faire, nous reproduisons l'expérience précédente en nommant notre encodeur "*source*", et nous entraînons un deuxième encodeur, dit "*cible*", sur lequel nous allons mesurer les performances de notre décodeur.

III.3.2.5 - A Le protocole Pour les jeux de données utilisés : $\mathcal{D}^{train}_{encoder}$, $\mathcal{D}^{test}_{encoder}$, $\mathcal{D}^{train}_{decoder}$ et $\mathcal{D}^{test}_{decoder}$, on réutilisera les mêmes définitions que dans les expériences précédentes. Pour plus de détails concernant ces données, voir section I.2.

Les deux premiers extracteurs de caractéristiques locuteur $encoder^{fast}_{train}$ et $encoder^{fast}_{test}$ sont des Fast-ResNet34 [51], alors que le troisième ($encoder^{half}_{test}$) sera un Half-ResNet34 [51]. Ils sont tous entraînés avec $\mathcal{D}^{train}_{encoder}$ jusqu'à atteindre des EER respectifs de 3.11%, 4.06% et 2.85% sur $\mathcal{D}^{test}_{encoder}$.

Pour la reconstruction d'extraits de parole, nous avons utilisé l'architecture AutoVC mais avec le même encodeur (un Fast-ResNet34). Le décodeur est entraîné sur les données de $\mathcal{D}^{train}_{decoder}$ et leurs embeddings associés ($\mathcal{E}^{train}_{decoder}$), en utilisant l'extracteur entraîné $\mathcal{D}^{train}_{encoder}$, jusqu'à stabilisation (500 époques). Nous avons utilisé la même version d'AutoVC que lors de l'expérience III.3.2.3.

Le protocole est représenté schématiquement dans la figure III.11.

TABLE III.5 – Tableau présentant les résultats de l'expérience III.3.2.5, en terme d'EER_{cible}, EER_{source} et de SFAR. La valeur maximale d'un EER est de 50%, on cherche donc à avoir EER_{source} aussi proche de 50 que possible.

Expérience	Encodeur	$EER_{cible} \downarrow$	$EER_{source} \uparrow$	$SFAR_{EER} \uparrow$	$SFAR_1 \uparrow$
III.3.2.5	$encoder^{fast}_{train}$	0.17%	50.00%	100.0%	99.74%
	$encoder^{fast}_{test}$	2.16%	45.97%	81.52%	6.09%
	$encoder^{half}_{test}$	8.65%	44.14%	66.95%	3.15%

III.3.2.5 - B Les résultats Pour mesurer les performances du décodeur entraîné, on utilise comme jeu de test $\mathcal{D}^{test}_{decoder}$. Les résultats sont présentés dans la table III.5 en utilisant les mêmes métriques que précédemment.

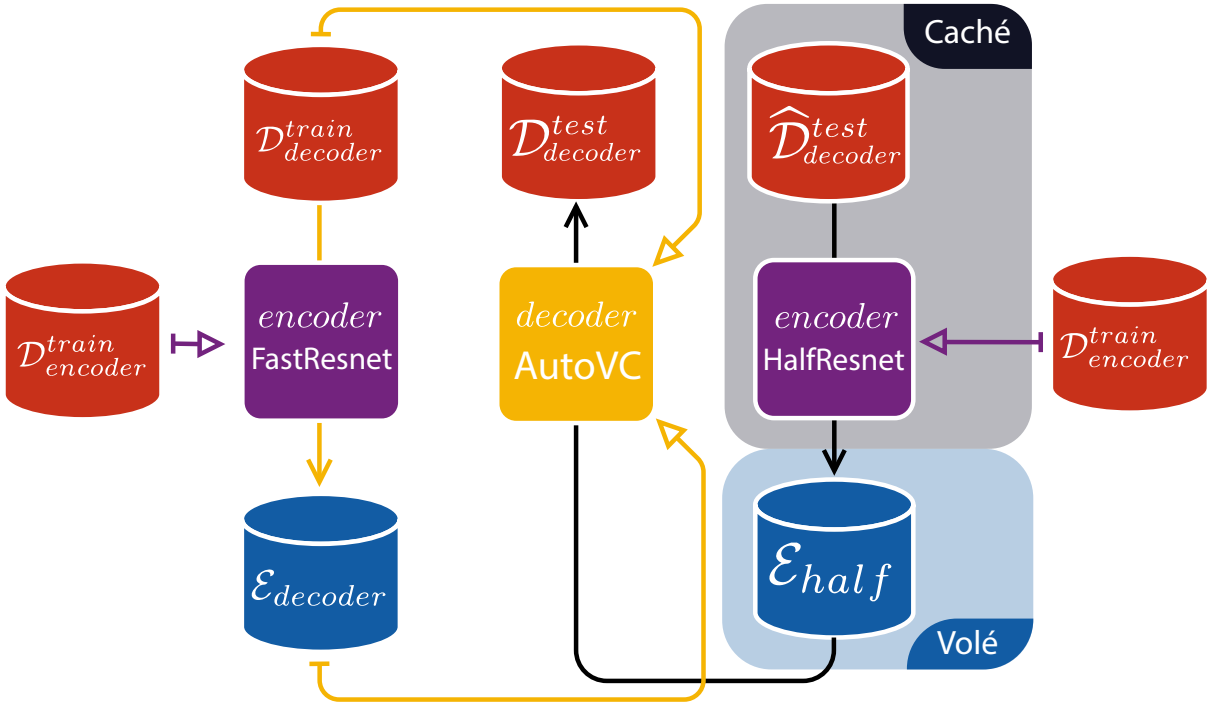

FIGURE III.11 – Schéma résumant l'expérience III.3.2.5.

Pour commencer, on remarque que bien que la reconstruction fonctionne toujours, l'utiliser sur un réseau initialisé différemment (sur $encoder^{fast}_{test}$ par exemple) donne des résultats en terme de SFAR plus faibles. De la même manière, quand on utilise un encodeur ayant une architecture différente, comme $encoder^{half}_{test}$, les résultats sont encore plus mauvais. Cependant, on notera qu'avec 81.52% sur un encodeur de même architecture et 66.95% sur un encodeur différent, on a tout de même un transfert entre les encodeurs significatif, donc une certaine généralisation de notre décodeur.

On retombe sur le même problème que pour la reconstruction de tracés de chiffres manuscrits : l'embedding d'un même utilisateur ne sera pas au même endroit en fonction de l'encodeur qui l'a calculé, donc la reconstruction ne fonctionnera qu'en adaptant le domaine des embeddings du premier encodeur à celui du deuxième. Pour adapter les sorties des deux encodeurs, il va falloir trouver comment les aligner, et c'est ce qu'on verra dans le chapitre IV.

III.4 Conclusion du chapitre

Dans ce chapitre, nous avons présenté et exécuté des attaques de reconstruction de *templates* sous plusieurs variantes, sur des embeddings de tracés de chiffres manuscrits et de parole. Ces attaques consistent à fabriquer un *décodeur*, un système capable de reconstruire une donnée biométrique à partir d'un embedding.

Dans le cas de chiffres manuscrits tracés à la main, nous avons utilisé comme décodeur des réseaux LSTM et LSTM-MDN, qui ont permis une reconstruction conservant l'information caractéristique du scripteur. Cependant, cette reconstruction supposant un accès en boite noire à l'encodeur qui avait produit les embeddings, le modèle d'attaque était très limité. Nous avons donc essayé d'entraîner le décodeur sur un encodeur différent de celui qui avait produit les embeddings, pour élargir le modèle d'attaque. Ce nouveau décodeur n'a pas donné de bons résultats, qu'il s'agisse de la reconstruction du chiffre ou de la conservation des informations du scripteur. Notre hypothèse pour expliquer que le décodeur ne fonctionne plus dans cette configuration est que les embeddings utilisés ne sont pas dans le bon espace vectoriel, et qu'il faudrait les projeter dans l'espace sur lequel le décodeur a été entraîné avant de les décoder. Pour trouver la projection optimale entre les deux espaces, nous allons avoir besoin de trouver l'alignement qui minimisera la distance entre les distributions d'embeddings des deux espaces : ce sera le sujet du chapitre suivant.

Dans le cas du traitement de la parole pour la vérification du locuteur, il n'est pas possible de reconstruire de la parole uniquement à partir d'un embedding, car celui-ci ne contient pas d'information linguistique. Pour ajouter une information linguistique, nous utilisons un système de conversion de voix : AutoVC. Ce système utilise l'information locuteur extraite d'un embedding et l'information linguistique d'un autre extrait de parole pour reconstruire un nouvel extrait de parole. De la même manière que pour les chiffres manuscrits, lorsque le décodeur a accès à l'encodeur, il reconstruit des données efficacement, mais lorsqu'il est entraîné sur un autre encodeur, les résultats sont bien moindres, nous allons donc aussi avoir besoin d'un alignement pour la vérification du locuteur, abordé dans le chapitre suivant.

Chapitre IV

Les alignements entre ensembles d'embeddings

Dans le chapitre précédent, nous avons vu plusieurs techniques de reconstruction pour reproduire une donnée biométrique (des extraits de parole ou des tracés de chiffres manuscrits) à partir d'embeddings. Nous nous sommes placés du point de vue d'un attaquant désirant frauder un système d'authentification biométrique. Nous avons supposé que l'attaquant peut avoir en sa possession un ensemble d'embeddings produits par un encodeur fonctionnant à base de réseaux de neurones, et qu'il possède un certain nombre d'informations sur l'encodeur utilisé. L'attaque a alors consisté à entraîner un décodeur capable de retrouver les données d'entrée à partir de ces embeddings, puis à utiliser ces données reconstruites en entrée du système d'authentification pour mesurer quelle proportion arrivait à le frauder. Les premières attaques, utilisant un accès en boite noire à l'encodeur, ont bien fonctionné. Nous avons ensuite essayé de ne plus supposer un accès à l'encodeur, mais uniquement la connaissance de son architecture. De cette manière, en entraînant un deuxième encodeur de même architecture avec des données différentes, nous avons pu réitérer l'attaque sans avoir besoin d'accéder à l'encodeur ayant produit les embeddings. Ces attaques n'ont pas fonctionné.

Notre hypothèse est que les embeddings produits par deux encodeurs ne sont pas dans le même domaine. Pour utiliser un décodeur entraîné sur un ensemble d'embeddings avec un ensemble d'un autre domaine, nous allons devoir trouver une fonction permettant d'aligner les deux ensembles d'embeddings. Dans ce chapitre, nous étudions plusieurs méthodes supervisées et non supervisées pour calculer une fonction d'alignement rapprochant deux ensembles de vecteurs ou distributions statistiques de manière optimale. Ces fonctions d'alignement seront d'abord utilisées dans le cadre des attaques sur les systèmes de vérification du locuteur et du scripteur présentées précédemment, puis pour vérifier les limites de systèmes d'anonymisation de la parole, en alignant des embeddings anonymisés et non anonymisés.

IV.1 Les alignements pour attaquer un système de vérification du scripteur

Nous avons vu dans la partie II qu'il était possible d'inférer les étiquettes des chiffres tracés à partir de leurs embeddings de manière non supervisée. Pour ça, nous avons aligné les ensembles d'embeddings produits par deux extracteurs différents pour pouvoir les comparer. Si la comparaison nous a permis de savoir à quels chiffres correspondaient quels embeddings, pour faire de même avec les utilisateurs, il aurait fallu déjà posséder un jeu d'embeddings contenant des exemples de tous les utilisateurs que nous souhaitions attaquer. La solution pour attaquer directement les utilisateurs semblait alors de reconstruire la donnée à partir des embeddings, permettant d'attaquer n'importe quel utilisateur dont on posséderait l'embedding.

Dans la partie III, nous avons trouvé des moyens de reconstruire les chiffres en utilisant respectivement des réseaux LSTM-MDN. Cependant, le système de reconstruction est entraîné pour un jeu d'embeddings donné, pour lesquels nous connaissons les données brutes. Lorsqu'utilisé sur un jeu d'embeddings différent, comme dans le cas d'une attaque, la reconstruction ne fonctionne plus. Notre hypothèse est que les embeddings produits par chaque extracteur sont dans un espace qui leur est propre. Pour utiliser les embeddings produits par un extracteur différent, il faudrait les projeter dans le bon espace. Pour ce faire, nous avons choisi de trouver l'alignement qui minimise la distance entre les embeddings des deux espaces, et de l'utiliser pour projeter les embeddings entre espaces.

IV.1.1 Un alignement sur des groupes de vecteurs

Le premier alignement que nous avons réalisé dans cette thèse a en réalité été présenté dans le chapitre II, expérience II.2.1.2. Cet alignement a été réalisé entre deux groupes d'embeddings de chiffres. Ces groupes ont été produits par deux réseaux BiLSTMs de même architecture, mais initialisés différemment et entraînés sur des ensembles de données disjoints.

L'objectif de ce premier alignement était de trouver des informations sur les chiffres. Sachant qu'il existe 10 chiffres dans notre système numérique, et que l'expérience II.2.1.1 a montré qu'on pouvait retrouver 10 groupes d'embeddings pertinents par rapport aux chiffres avec un clustering *K-means*, nous avons donc aligné des groupes d'embeddings et pas directement des embeddings.

Cette méthode a montré son fonctionnement pour l'étiquetage de groupes de chiffres, avec 100% de précision pour leur étiquetage. Cependant, sa complexité et ses hypothèses sur le nombre de classes connues n'ont pas permis de l'utiliser directement pour identifier les utilisateurs.

IV.1.2 Un alignement pour ajuster les utilisateurs

Pour attaquer les utilisateurs, nous avons entraîné dans le chapitre III un système permettant de reconstruire des tracés de chiffres à partir de leurs embeddings. Cependant, sans alignement préalable entre l'espace des embeddings utilisé pour l'attaque et l'espace sur lequel le décodeur a été entraîné, l'attaque ne fonctionnait pas.

TABLE IV.1 – Tableau présentant les résultats des expériences IV.1.2.1, IV.1.2.2, IV.1.2.4 et IV.1.2.5, en terme d'$Accuracy_{chiffre}$, EER et de $SFAR$ (pour des seuils à l'EER (13.53%) et à 1%). Les encodeurs et décodeurs utilisés ont tous la même architecture. Toutes les métriques ont été calculées en utilisant les embeddings calculés à partir des données de $\mathcal{D}_{target}^{test}$ avec l'encodeur cible. Les lignes 1 et 2 sont données à titre indicatif pour mesurer la performance du décodeur sur les données sources (expérience III.2.2.1) et sur les données ciblées, sans alignement (expérience III.2.2.2).

	Expérience	Alignement	$Accuracy \uparrow$	$EER \downarrow$	$SFAR_{EER} \uparrow$	$SFAR_{1\%} \uparrow$
1	III.2.2.1	Aucun	78,51%	13,53%	95,62%	67,88%
2	III.2.2.2	Aucun	9,45%	39.01%	1,04%	0,02%
3	IV.1.2.1	Procrustes sur N	68.14%	36.67%	8.34%	0,00%
4	IV.1.2.2	Procrustes + Affinage	70.91%	30.65%	23.61%	0,01%
5	IV.1.2.4	Wasserstein Procustes	77.38%	24.10%	54.64%	0,55%
6	IV.1.2.5	Wasserstein Procrustes Oracle	76.38%	21.96%	80.36%	5.31%
7	IV.1.2.5	Procrustes Oracle	77.14%	21.66%	81.40%	6.06%

IV.1.2.1 Reconstruction de chiffres avec l'analyse de Procrustes.

Dans cette expérience, nous nous plaçons du point de vue d'un attaquant s'étant procuré un jeu d'embeddings non étiquetés $\mathcal{E}_{target}$, produits par un encodeur LSTM à partir de chiffres manuscrits. En tant qu'attaquant, on cherche à reconstruire les données à l'origine des embeddings, et à les utiliser pour frauder un système d'authentification par vérification automatique du scripteur.

IV.1.2.1 - A Le protocole Dans cette expérience, on utilisera les mêmes réseaux *encodeur d'attaque*, *encodeur cible* et *décodeur* que dans l'expérience III.2.2.2, entraînés de la même manière. On utilisera pour ça 4 jeux de données issus de $\mathcal{D}_{numbers}$,présenté section I.2.1 : $\mathcal{D}^{test}_{attack}$, $\mathcal{D}^{train}_{attack}$, $\mathcal{D}^{test}_{target}$ et $\mathcal{D}^{train}_{target}$, contenant chacun le quart des utilisateurs.

Les *encodeur d'attaque* et *encodeur cible* sont entraînés respectivement avec $\mathcal{D}^{train}_{attack}$ et $\mathcal{D}^{train}_{target}$. Une fois entraînés, leurs tests sur les jeux $\mathcal{D}^{test}_{attack}$ et $\mathcal{D}^{test}_{target}$ donnent respectivement une Accuracy de 94.76% et 94.64%, et un EER de 13.21% et 14.13%. On pose $\mathcal{E}_{attack}$ et $\mathcal{E}_{target}$ les ensembles d'embeddings produits respectivement par les encodeurs d'attaque et encodeur cible à partir des données $\mathcal{D}^{test}_{attack}$ et $\mathcal{D}^{test}_{target}$.

$\mathcal{E}_{target}$ est le jeu d'embeddings que l'attaquant va chercher à reconstruire. On suppose que l'attaquant s'est procuré ses propres données ($\mathcal{D}^{train}$) et a entraîné son propre encodeur avec celles-ci (l'encodeur d'attaque).

On entraîne donc un décodeur LSTM-MDN avec $\mathcal{D}^{train}_{attack}$ et l'*encodeur d'attaque*. Une fois testé sur l'encodeur d'attaque avec les données $\mathcal{D}^{test}_{attack}$, il présente un Accuracy de 71.51% sur la détection de chiffres et un EER de 13.53%.

Comme nous l'avons vu à l'expérience III.2.2.2, pour faire fonctionner le décodeur avec les embeddings de $\mathcal{E}_{target}$, il faut trouver un alignement entre les embeddings produits par les deux encodeurs.

Dans la section II.2.1, nous avons vu des méthodes pour étiqueter les chiffres associés aux embeddings de $\mathcal{E}_{target}$, en utilisant :

- Soit un clustering [71], une PCA [76], un algorithme génétique [78] et un affinage.
- Soit un clustering [71], une LDA [80] et un algorithme génétique [78].

Une fois les chiffres associés aux clusters de $\mathcal{E}_{target}$ connus, on peut utiliser une analyse de Procrustes [77] pour calculer la rotation W qui rapprochera de manière optimale des centres des clusters de $\mathcal{E}_{target}$ des centres des clusters de $\mathcal{E}_{attack}$. On utilise ensuite cette rotation pour projeter le jeu $\mathcal{E}_{target}$ dans l'espace de $\mathcal{E}_{attack}$, ce qui nous donne $\mathcal{E}^{W}_{target}$, et

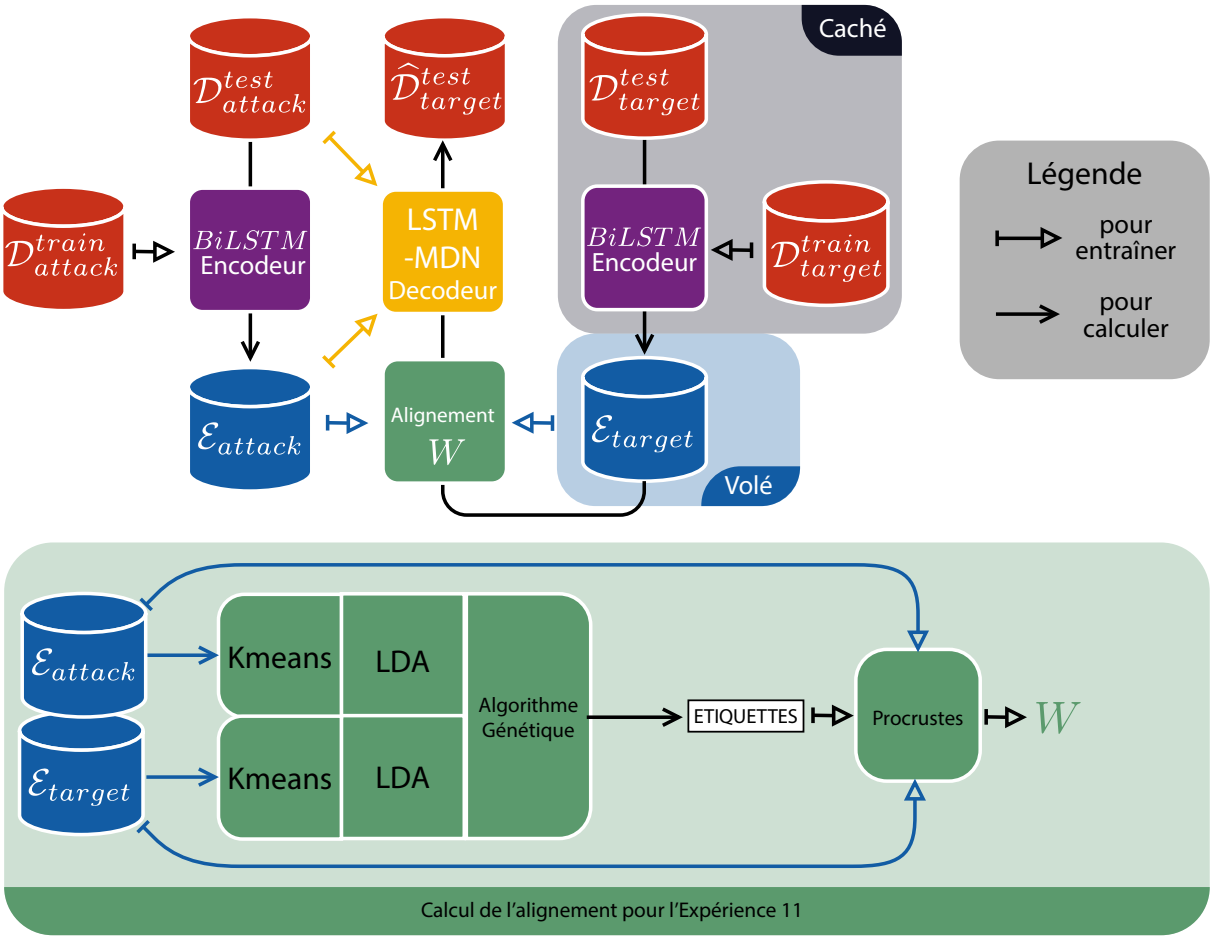

FIGURE IV.1 – Schéma représentant le protocole l'expérience IV.1.2.1.

on fait passer ces embeddings projetés dans le décodeur pour estimer les données $\widehat{\mathcal{D}}^{test}_{target}$.

La figure IV.1 résume le protocole présenté ci-dessus.

IV.1.2.1 - B Les résultats Pour savoir si les données reconstruites pourraient être utilisées pour frauder l'encodeur attaqué, on va rejouer les données reconstruites $\widehat{\mathcal{D}}^{test}_{target}$ et mesurer le SFAR (voir section I.3) pour différents seuils. Les résultats sont présentés dans la table IV.1, ligne 3.

La figure IV.2 montre un exemple de la reconstruction de chiffres manuscrits de $\mathcal{D}^{test}_{target}$ en utilisant le décodeur entraîné sur $\mathcal{D}^{train}_{attack}$ et l'alignement proposé dans cette expérience.

On remarque que, bien qu'il y ait une nette amélioration sur l'Accuracy sur les chiffres, le taux d'attaques réussies (SFAR) reste très faible, et ce quel que soit le seuil.

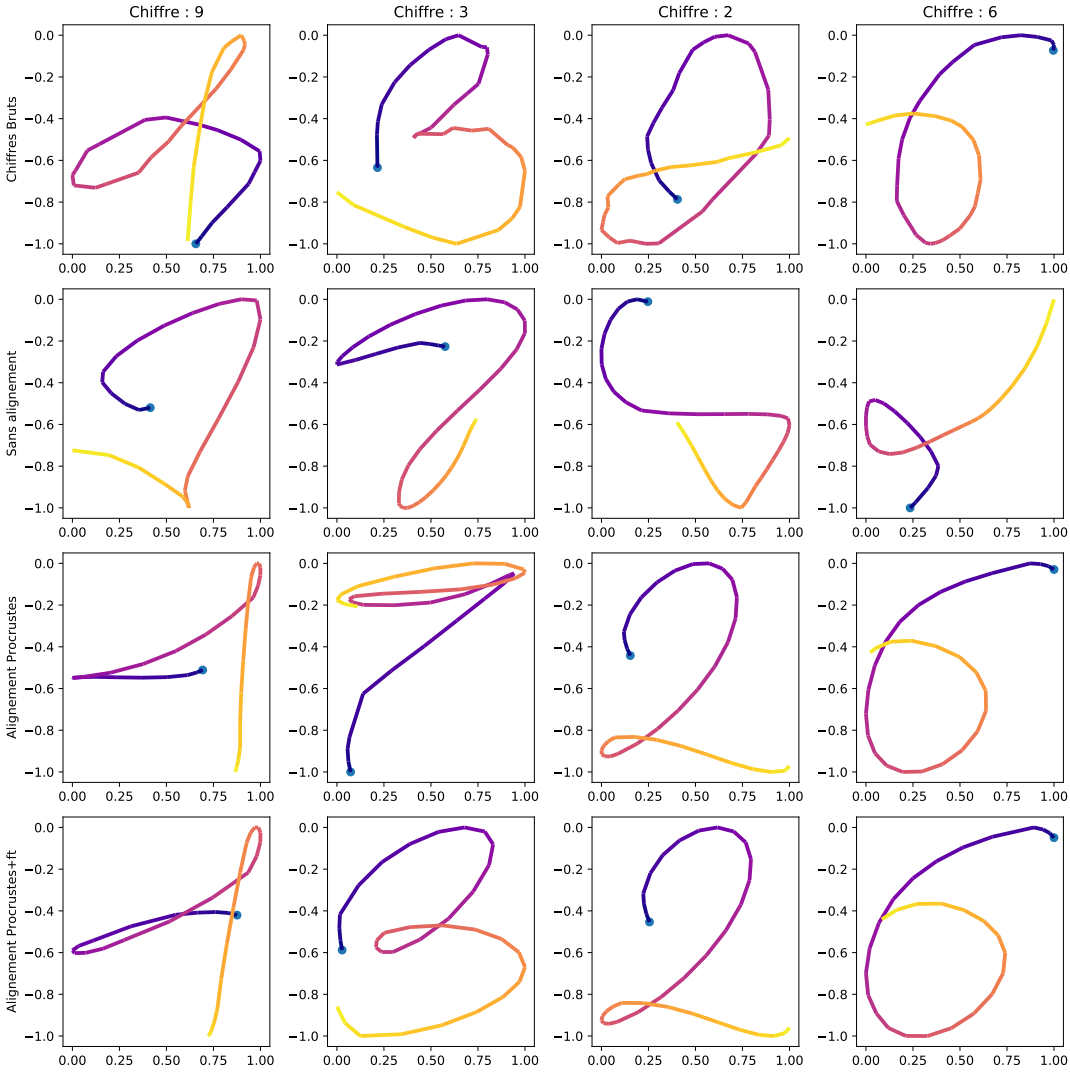

FIGURE IV.2 – Tracés de chiffres manuscrits produits par le décodeur utilisé avec des embeddings de $\mathcal{E}_{target}$, en suivant le protocole des expériences III.2.2.2,IV.1.2.1 et IV.1.2.2, avec les tracés originaux en comparaison sur la première ligne. Le point bleu marque le début du tracé et le dégradé du violet au jaune permet de percevoir l'ordre du tracé.

IV.1.2.2 Reconstruction de chiffres avec un alignement affiné.

Pour améliorer les résultats de l'expérience précédente, nous allons affiner la fonction d'alignement trouvée.

On se place donc encore du point de vue d'un attaquant ayant accès à un jeu d'embeddings non étiquetés $\mathcal{E}_{target}$, produits à partir de chiffres manuscrits. On cherche à frauder le système d'authentification en reconstruisant les tracés des chiffres à l'origine de ces embeddings.

IV.1.2.2 - A Le protocole Dans cette expérience, on réutilise les mêmes encodeurs et le même décodeur que dans l'expérience IV.1.2.1.

On commence par faire un clustering sur les embeddings de $\mathcal{E}_{target}$, puis on utilise la méthode de l'expérience II.2.1.3 pour trouver à quel chiffre correspond quel cluster. On compile une matrice de rotation W en utilisant l'analyse de Procrustes pour minimiser la distance entre les centres des clusters de chiffres.

W étant une matrice de rotation à coefficients réels, il est possible de l'affiner avec une descente de gradient stochastique et d'optimiser ses poids afin de minimiser une fonction de coût $\mathcal{L}$ donnée. Ici, on utilisera la *Log vraisemblance* entre les embeddings $\mathcal{E}_{target}$ et $\mathcal{E}_{attack}$ (voir section I.3.3) comme fonction de coût. La figure IV.3 résume le protocole de l'expérience.

IV.1.2.2 - B Les résultats Pour savoir si les données reconstruites pourraient être utilisées pour frauder l'encodeur attaqué, on va rejouer les données reconstruites $\widehat{\mathcal{D}}^{test}_{target}$ et mesurer le SFAR (voir section I.3) pour différents seuils. Les résultats sont présentés dans la table IV.1, ligne 4. On peut voir une augmentation du SFAR pour tous les seuils. Ces résultats montrent qu'il est possible d'inférer une information sur les scripteurs à partir d'un ensemble d'embeddings, sans accès ni à la valeur de leurs chiffres, ni à l'encodeur qui les a produits. La figure IV.2 montre un exemple de la reconstruction de chiffres manuscrits de $\mathcal{D}^{test}_{target}$ en utilisant le décodeur entraîné sur $\mathcal{D}^{train}_{attack}$ et l'alignement affiné.

Cependant, le taux de réussite de l'attaque reste proche du taux de faux positifs admis à chaque fois, les résultats de l'attaque pourraient donc être davantage une exploitation des faux positifs autorisés par le système qu'une fraude réelle.

Les travaux présentés dans les expériences IV.1.2.1 et IV.1.2.2 ont fait l'objet d'une publication [41] dans la conférence "*International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*", lors de sa 46e édition, en 2021.

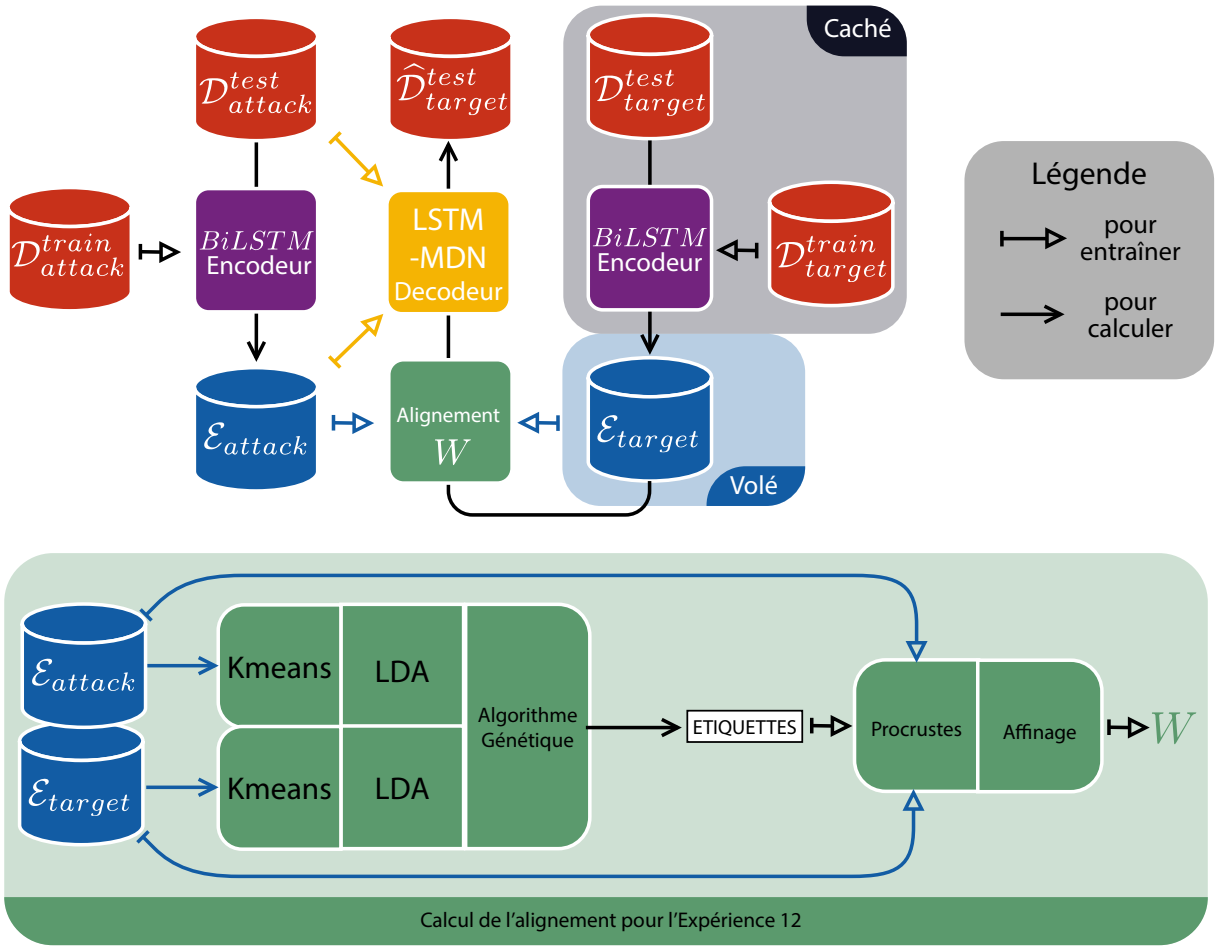

FIGURE IV.3 – Schéma représentant le protocole l'expérience IV.1.2.2.

IV.1.2.3 Wasserstein Procrustes

Pour améliorer encore notre alignement, nous avons décidé d'utiliser une nouvelle méthode. La méthode de Wasserstein Procrustes [100] est une méthode d'alignement visant à trouver la rotation qui minimise la distance entre deux jeux d'embeddings. Initialement, celle-ci a été développée pour aligner des embeddings de mots de langues différentes pour faire de la traduction automatique.

Soit $\mathcal{X} \in \mathbb{R}^{N \times D}$ et $\mathcal{Y} \in \mathbb{R}^{N \times D}$ deux jeux composés de N embeddings de dimension D.

IV.1.2.3 - A L'analyse de Procrustes [77] est une méthode permettant, si on connaît l'association entre les points de $\mathcal{X}$ et de $\mathcal{Y}$, de trouver la matrice W à coefficients réels telle que :

$$W^* = \min_{W \in \mathbb{R}^{D \times D}} ||\mathcal{X}W - \mathcal{Y}||_2^2 \tag{IV.1}$$

On peut trouver une solution dans l'espace des matrices orthogonales $\mathcal{O}_N$. Soit $\mathcal{X}^T\mathcal{Y} = USV^T$ la décomposition en valeurs singulières de $\mathcal{X}^T\mathcal{Y}$, alors l'analyse de Procrustes pose $W^* = UV^T$.

IV.1.2.3 - B La distance de Wasserstein [101] est une mesure du transport optimal nécessaire pour faire bouger un ensemble de point aux positions d'un second groupe. Si le n^{eme} élément de $\mathcal{X}$ correspond au n^{eme} élément de $\mathcal{Y}$, $\forall n \in [\![1, N]\!]$, alors la distance entre $\mathcal{X}$ et $\mathcal{Y}$ est : $||\mathcal{X} - \mathcal{Y}||_2^2$. En revanche, si $\mathcal{X}$ et $\mathcal{Y}$ ne sont pas ordonnés, il faut alors faire correspondre chaque vecteur de l'un avec un vecteur de l'autre. Pour cela, on utilise une matrice permutation. Soit $\mathcal{P}_N$ étant l'ensemble des matrices de permutation, les matrices qui assurent une association 1 à 1 : $\mathcal{P}_N = \{P \in \{0,1\}^{N \times N}, P\mathbf{1}_N = \mathbf{1}_N, P^T\mathbf{1}_N = \mathbf{1}_N\}$. Dans notre problème, on cherche donc la matrice de permutation $P^* \in \mathcal{P}_N$ qui minimisera la distance de Wasserstein entre 2 groupes de points :

$$P^* = \min_{P \in \mathcal{P}_N} ||\mathcal{X} - P\mathcal{Y}||_2^2 \tag{IV.2}$$

IV.1.2.3 - C Résoudre stochastiquement l'analyse de Procrustes en distance de Wasserstein Dans notre problème, on cherche à résoudre simultanément les équations IV.1 et IV.2 :

$$\min_{W \in \mathcal{O}_D} \min_{P \in \mathcal{P}_N} \|\mathcal{X}W - P\mathcal{Y}\|_2^2 \tag{IV.3}$$

On peut aussi écrire ce problème sous la forme :

$$\max_{W \in \mathcal{O}_D} \max_{P \in \mathcal{P}_N} tr(\mathcal{Y} W^T \mathcal{X}^T P) \tag{IV.4}$$

Pour ce faire, on va trouver la solution de chacune des équations pour différents sous ensembles de $\mathcal{X}$ et de $\mathcal{Y}$ sélectionnés aléatoirement, tour à tour, et faire varier petit à petit la rotation recherchée. Soit $W_t \in \mathcal{O}_N$ la matrice de rotation à l'étape t, $\mathcal{X} \in \mathbb{R}^{N \times D}$ et $\mathcal{Y} \in \mathbb{R}^{N \times D}$ deux ensembles de vecteurs, b un entier non nul, on applique donc l'algorithme suivant : Pour t allant de 1 à T :

1. Tirer aléatoirement $\mathcal{X}_t$ de $\mathcal{X}$ et $\mathcal{Y}_t$ de $\mathcal{Y}$, deux sous-ensembles de taille b.
2. Soit la matrice orthogonale W_t connue, calculer la matrice de permutation optimale P_t :
$$P_t = argmax_{P \in \mathcal{P}_N} tr(\mathcal{Y}_t W_t^T \mathcal{X}_t^T P)$$
3. Calculer le gradient G_t par rapport à W_t :
$$G_t = -2\mathcal{X}_t^T P_t \mathcal{Y}_t$$
4. Ajouter le gradient et projeter dans l'espace des matrices orthogonales :
$$W_{t+1} = \Pi_{\mathcal{O}_D}(W_t - \alpha G_t), \alpha \in \mathbb{R}^{+*}$$

   Pour toute matrice $M \in \mathbb{R}^{D \times D}$, la projection dans l'espace des matrices orthogonales est donnée par $\Pi_{\mathcal{O}_D}(M) = UV^T$, avec USV^T la décomposition en valeurs singulières de M.

Après T étapes, on obtient une matrice $W_T = W^*$ qui tend à vérifier l'équation IV.3, à condition d'avoir le taux d'apprentissage α et l'initialisation de W correctement choisis.

IV.1.2.3 - D L'initialisation Pour initialiser W, Grave, Joulin et Berthet [100] utilisent les mots les plus fréquents des deux langages à aligner, et supposent qu'une fois triés par fréquence d'utilisation, ils se correspondent. W est donc initialisé avec un Procrustes sur un sous-ensemble choisi à la main.

Dans notre cas, on ne possède que l'information de la valeur des chiffres associés aux embeddings. On initialise donc W en utilisant l'analyse de Procrustes [77] avec les moyennes des groupes d'embeddings de même chiffre.

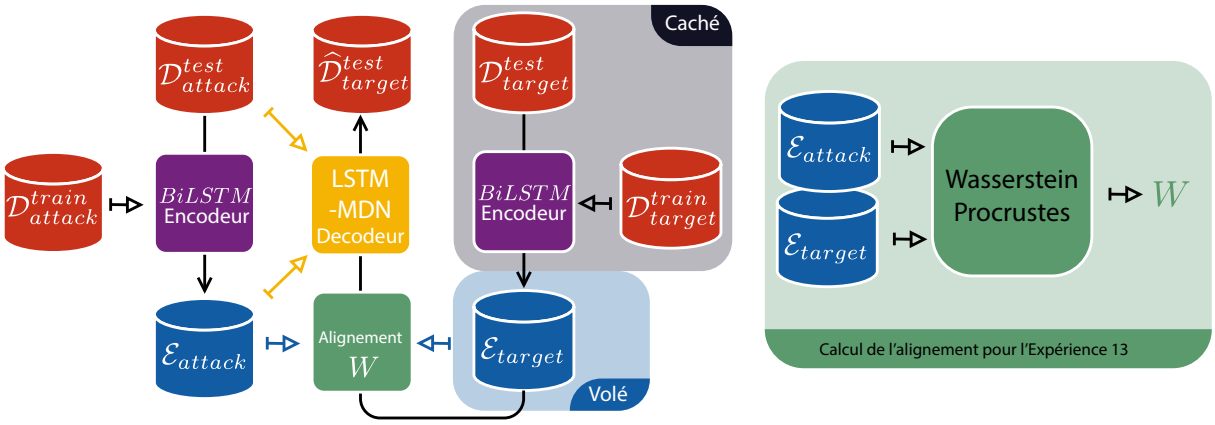

FIGURE IV.4 – Schéma représentant le protocole l'expérience IV.1.2.4.

IV.1.2.4 Reconstruction de chiffres avec Wasserstein Procrustes.

Pour améliorer encore les résultats de l'expérience précédente, nous avons décidé d'essayer un nouvel algorithme d'alignement : la méthode de Wasserstein Procrustes [100], présentée dans la section IV.1.2.3.

IV.1.2.4 - A Le Protocole Le protocole est exactement le même que pour l'expérience IV.1.2.2, à la différence que l'alignement est calculé avec l'algorithme de Wasserstein Procrustes à partir des jeux $\mathcal{E}_{target}$ et $\mathcal{E}_{attack}$. La figure IV.4 résume le protocole de l'expérience.

IV.1.2.4 - B Les résultats Les résultats sont présentés dans la table IV.1, ligne 5. On peut voir une augmentation conséquente du taux d'attaques réussies (SFAR), ce qui nous permet d'affirmer que les embeddings de chiffres manuscrits contiennent effectivement suffisamment d'informations pour reconstruire les tracés originaux.

La figure IV.5 montre un exemple de la reconstruction de chiffres manuscrits de $\mathcal{D}^{test}_{target}$ en utilisant le décodeur entraîné sur $\mathcal{D}^{train}_{attack}$ et l'alignement de Wasserstein Procrustes.

IV.1.2.5 Reconstruction de chiffres avec un alignement oracle.

Pour mesurer les résultats optimaux qui pourraient être obtenus par un alignement sous forme de rotation, nous avons décidé de calculer les performances du système en supposant que nous ayons un accès en boite noire à l'encodeur attaqué.

Cet accès nous permet de faire passer des données connues dans l'encodeur ciblé, et donc :

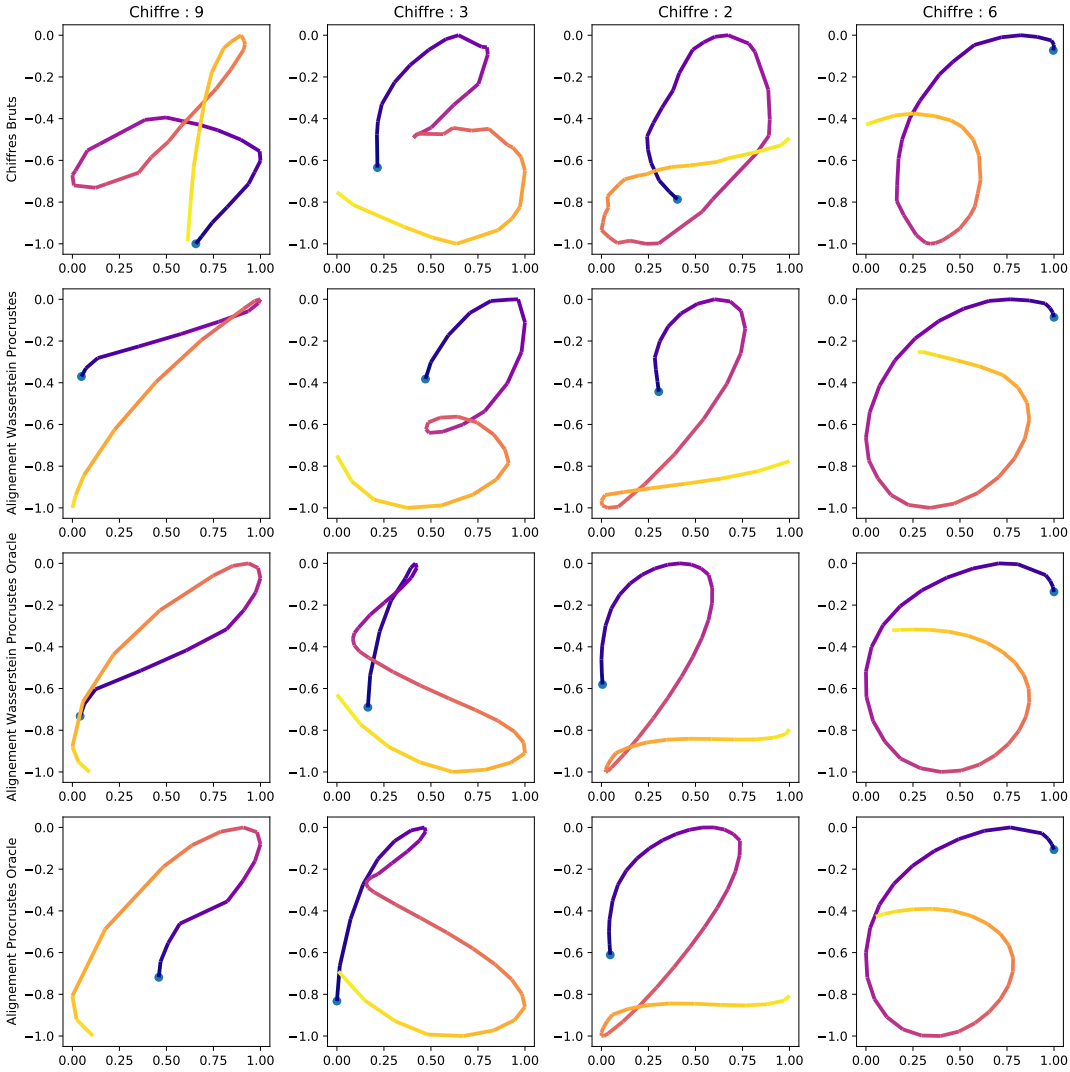

FIGURE IV.5 – Tracés de chiffres manuscrits produits par le décodeur utilisé avec des embeddings de $\mathcal{E}_{target}$, en suivant le protocole des expériences IV.1.2.4 et IV.1.2.5, avec les tracés originaux en comparaison sur la première ligne. Le point bleu marque le début du tracé et le dégradé du violet au jaune permet de percevoir l'ordre du tracé.

- De mesurer les limites de la méthode de Wasserstein Procrustes [100], déjà dans IV.1.2.4.
- De mesurer les limites d'un alignement par rotation, en utilisant un alignement supervisé : l'analyse de Procrustes [77].

IV.1.2.5 - A Le Protocole On ré-utilisera pour ce protocole les 4 jeux de données issus de $\mathcal{D}_{numbers}$, présenté section I.2.1 : $\mathcal{D}^{test}_{attack}$, $\mathcal{D}^{train}_{attack}$, $\mathcal{D}^{test}_{target}$ et $\mathcal{D}^{train}_{target}$, contenant chacun le quart des utilisateurs. Le protocole est similaire à celui des expériences précédentes :

- L'*encodeur d'attaque* est entraîné sur $\mathcal{D}^{train}_{attack}$.
- L'*encodeur cible* est entraîné sur $\mathcal{D}^{train}_{target}$.
- On calcule $\mathcal{E}_{attack}$ en faisant passer $\mathcal{D}^{test}_{attack}$ dans l'*encodeur d'attaque* entraîné.
- Le *décodeur* est entraîné sur $\mathcal{D}^{test}_{attack}$ et $\mathcal{E}_{attack}$.

La différence vient à partir de l'alignement. On suppose qu'on a accès aux données de $\mathcal{D}^{test}_{target}$. On va donc calculer :

- $\mathcal{E}^{*}_{attack}$ en faisant passer $\mathcal{D}^{test}_{attack}$ dans l'*encodeur cible* entraîné.
- $\mathcal{E}_{attack}$ en faisant passer $\mathcal{D}^{test}_{attack}$ dans l'*encodeur d'attaque* entraîné.

Le fait d'avoir des embeddings provenant du même jeu de données va faciliter l'alignement, et nous permettre également d'utiliser un alignement supervisé.

On utilise donc Wasserstein Procrustes sur $\mathcal{E}_{attack}$ et $\mathcal{E}^{*}_{attack}$ pour produire une première matrice de rotation : $\mathcal{R}_{WP}$. On utilise ensuite cette rotation pour projeter le jeu $\mathcal{E}^{*}_{attack}$ dans l'espace de $\mathcal{E}_{attack}$, ce qui nous donne $\mathcal{E}^{\mathcal{R}_{WP}}_{target}$, et on fait passer ces embeddings projetés dans le décodeur pour estimer les données $\widehat{\mathcal{D}}^{test-\mathcal{R}_{WP}}_{attack}$.

Sachant que la correspondance entre les embeddings de $\mathcal{E}_{attack}$ et $\mathcal{E}^{*}_{attack}$ est connue, on va pouvoir utiliser l'analyse de Procrustes pour calculer une deuxième matrice de rotation : $\mathcal{R}_P$, et l'utiliser la même manière pour produire le jeu de données $\widehat{\mathcal{D}}^{test-\mathcal{R}_P}_{attack}$. La figure IV.6 résume le protocole de l'expérience.

IV.1.2.5 - B Les résultats Les résultats sont présentés dans la table IV.1, lignes 6 et 7 (respectivement pour $\mathcal{R}_{WP}$ et $\mathcal{R}_P$. Si la précision sur la détection des chiffres, comme l'EER après rotation est peu impactée, on remarque une nette amélioration dans le $SFAR$ (80.36% et 81.40%), et une faible différence de performances entre l'algorithme supervisé et non supervisé. On remarque cependant que les performances optimales après alignement restent inférieures à celles obtenues avec un accès en boite noire à l'encodeur.

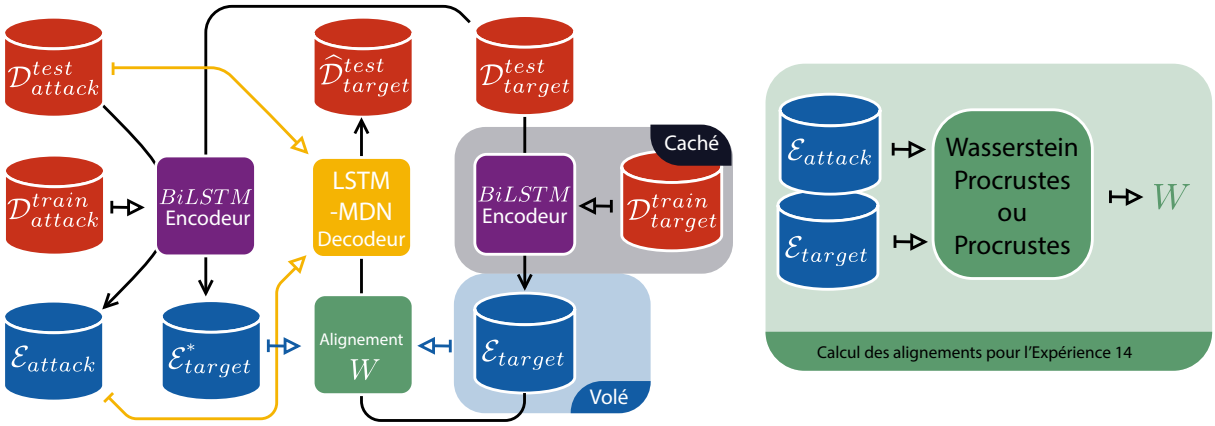

FIGURE IV.6 – Schéma représentant le protocole l'expérience IV.1.2.5.

Si nous souhaitions aller plus loin, cela nécessiterait donc d'utiliser un alignement plus complexe, comme un *alignement flow* (voir section III.1.3).

La figure IV.5 montre un exemple des chiffres reconstruits de $\widehat{\mathcal{D}}^{test-\mathcal{R}_{WP}}_{attack}$ et $\widehat{\mathcal{D}}^{test-\mathcal{R}_{P}}_{attack}$.

IV.2 Les alignements pour attaquer un système de vérification du locuteur

Dans le chapitre III, nous avons trouvé des moyens de reconstruire la parole à partir de x-vectors, en utilisant un système de conversion de voix : AutoVC. Cependant, dans les deux cas, le système de reconstruction est entraîné pour un jeu d'embeddings donné, pour lesquels nous connaissons les données brutes. Lorsqu'utilisé sur un jeu d'embeddings différent, comme dans le cas d'une attaque, la reconstruction ne fonctionne plus (voir expérience III.3.2.5).

Notre hypothèse est la même que pour la section précédente : les embeddings produits par chaque extracteur sont dans un espace qui leur est propre. Pour utiliser les embeddings produits par un extracteur différent, il faudrait les projeter dans le bon espace. Pour ce faire, nous avons choisi de trouver l'alignement qui minimise la distance entre les embeddings des deux espaces, et de l'utiliser pour projeter les embeddings entre espaces.

IV.2.1 La reconstruction de parole avec des alignements

Comme nous l'avons vu dans la section IV.1, l'alignement le plus efficace entre ensembles d'embeddings que nous avons trouvé est Wasserstein Procrustes, c'est donc directement celui-là que nous allons utiliser.

IV.2.1.1 Reconstruction de parole avec Wasserstein Procrustes.

L'objectif de cette expérience est de montrer qu'il est possible de reconstruire de la parole à partir d'embeddings sans avoir accès à l'encodeur, en utilisant un deuxième encodeur et un alignement de Wasserstein Procrustes. On se place du point de vue d'un attaquant désirant frauder un système d'authentification du locuteur, en utilisant une base d'embeddings d'enrôlement du système. On suppose que l'attaquant peut avoir connaissance de l'architecture de l'encodeur utilisé, qu'il a pu se fournir une base de données contenant des extraits de parole de locuteurs différents de ceux de la base volée. L'objectif pour cet attaquant est de produire des extraits de parole qui soient détectés par le système d'authentification comme appartenant aux mêmes locuteurs que ceux à l'origine des embeddings, sans aucune contrainte sur les mots utilisés.

IV.2.1.1 - A Les données Soit $\mathcal{D}^{train}_{encoder}$ le jeu de données composé de segments de paroles de 2 secondes extraits de voxceleb2 [35] et $\mathcal{D}^{test}_{encoder}$ les données de VoxCeleb1 [34]. Soit $\mathcal{D}^{train}_{decoder}$ le jeu de données composé des 100 premiers locuteurs de VCTK [36] et $\mathcal{D}^{test}_{decoder}$ les données des locuteurs restants. Pour plus de détails concernant ces données, voir section I.2.

IV.2.1.1 - B Le protocole On considère deux extracteurs de caractéristiques locuteur $encoder^{fast}$ et $encoder^{half}$, respectivement un Fast-ResNet34 [51] et un Half-ResNet34 [51]. Ils sont entraînés avec $\mathcal{D}^{train}_{encoder}$ jusqu'à atteindre des EER respectifs de 2.31% et 2.45% sur $\mathcal{D}^{test}_{encoder}$.

Pour la reconstruction d'extraits de parole, nous avons utilisé l'architecture AutoVC optimisée (telle que présentée à l'expérience III.3.2.2), en utilisant notre nouvelle fonction de coût pour la reconstruction (présentée dans la section III.3.2.3). On appellera $\mathcal{E}_{decoder}$ les embeddings produits par l'encodeur $encoder^{fast}$ à partir des données $\mathcal{D}^{train}_{decoder}$. Le décodeur est entraîné sur les données de $\mathcal{D}^{train}_{decoder}$ et leurs embeddings associés, en utilisant l'extracteur entraîné $\mathcal{D}^{train}_{encoder}$, jusqu'à stabilisation (500 époques).

Soit $\mathcal{E}_{fast}$ et $\mathcal{E}_{half}$ les embeddings respectivement produits par les encodeurs $encoder^{fast}$ et $encoder^{half}$ à partir des données $\mathcal{D}^{test}_{decoder}$. On utilise l'algorithme de Wasserstein Procrustes [100] pour trouver la matrice de rotation optimale W qui rapprochera au mieux les ensembles d'embeddings $\mathcal{E}_{fast}$ et $\mathcal{E}_{half}$.

On suppose que l'attaquant a volé le jeu d'embeddings $\mathcal{E}_{fast}$ et cherche à reconstruire des extraits de parole prononcés de la même manière que les locuteurs qui le composent, sans accès à l'encodeur $encoder^{fast}$.

On va donc utiliser la matrice W calculée pour projeter les embeddings de $\mathcal{E}_{half}$ dans l'espace des embeddings de $\mathcal{E}_{fast}$ pour pouvoir utiliser plus efficacement notre système de conversion de voix. La figure IV.7 schématise le protocole de cette expérience.

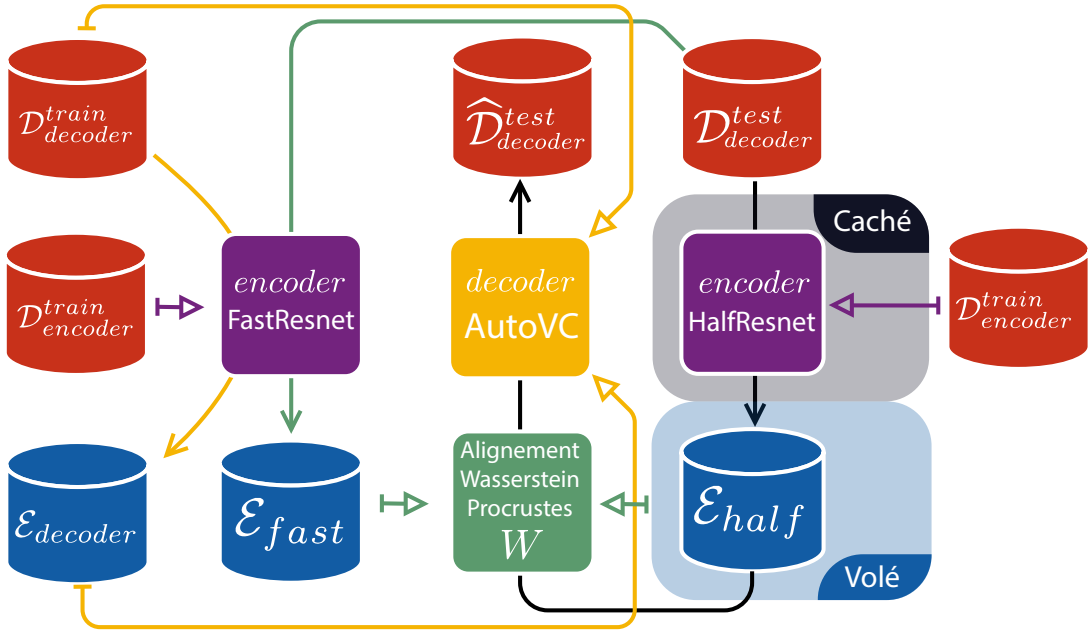

FIGURE IV.7 – Schéma résumant l'expérience IV.2.1.1.

IV.2.1.1 - C Les résultats Pour mesurer les performances de l'attaque, on utilise comme jeu de test $\mathcal{D}^{test}_{decoder}$. Comme précisé ci-dessus, après être passé par l'encodeur $encoder^{fast}$, on obtient le jeu d'embeddings $\mathcal{E}_{fast}$. Ensuite on projette le jeu $\mathcal{E}_{fast}$ dans l'espace des embeddings $\mathcal{E}_{half}$ en utilisant la rotation W, ce qui nous donne le jeu $\mathcal{E}^{W}_{fast}$. Puis on va reconstruire le jeu $\mathcal{E}^{W}_{fast}$ avec le décodeur entraîné pour obtenir un jeu de données de parole $\widehat{\mathcal{D}}^{test}_{decoder}$. Les résultats sont présentés dans la table IV.2, incluant une comparaison avec l'expérience III.3.2.5, pour les EER_{cible} et EER_{source} et les SFAR.

TABLE IV.2 – Tableau présentant les résultats de l'expérience IV.2.1.1, en terme d'EER_{cible}, EER_{source} et de SFAR, mesurés sur $\mathcal{D}^{test}_{decoder}$. La valeur maximale d'un EER est de 50%, on cherche donc à avoir EER_{source} aussi proche de 50 que possible. Les résultats de l'expérience III.3.2.5 sont aussi présentés à titre de comparaison.

Expérience	Encodeur	Alignement	$EER_{cible} \downarrow$	$EER_{source} \uparrow$	$SFAR_{EER} \uparrow$	$SFAR_1 \uparrow$
III.3.2.5	$fast$	Aucun	0.17%	50.00%	100.0%	99.74%
	$half$	Aucun	8.65%	44.14%	66.95%	3.15%
IV.2.1.1	$half$	WP	12.96%	46.53%	94.40%	90.81%
IV.2.1.2	$half$	Procrustes	8.33%	47.84%	98.00%	96.72%

On remarque que les résultats, bien que montrant une amélioration par rapport au cas sans alignement, ne sont pas suffisants pour montrer que l'attaque est une réussite.

A l'issue de cette expérience, deux possibilités s'offrent à nous :

1. Soit notre algorithme pour trouver l'alignement n'est pas assez bon.
2. Soit la rotation est une fonction d'alignement trop simple pour la situation.

Pour départager ces hypothèses, nous allons faire une expérience avec la rotation optimale, déterminée par une analyse de Procrustes, pour permettre de trouver une borne supérieure de l'alignement par rotation.

IV.2.1.2 Reconstruction de parole avec un alignement optimal.

L'objectif de cette expérience est de mesurer le résultat d'une attaque d'un système d'authentification en utilisant un alignement optimal, c'est-à-dire en utilisant la rotation la plus optimale possible entre deux groupes d'embeddings. Pour ce faire, on se placera du point de vue d'un attaquant informé, ayant un accès en boite noire à l'encodeur qu'il cherche à attaquer, mais uniquement pour calculer son alignement.

IV.2.1.2 - A Le protocole Dans cette expérience, on utilisera exactement le même protocole que dans l'expérience précédente (IV.2.1.1) pour la répartition des données, l'entraînement des deux encodeurs et du décodeur.

La différence est dans le calcul de l'alignement : Soit $\mathcal{E}_{fast}$ et $\mathcal{E}_{half}$ les jeux d'embeddings produits respectivement à partir des encodeurs $encoder^{fast}$ et $encoder^{half}$ à partir des données $\mathcal{D}^{test}_{decoder}$.

On va utiliser l'analyse de Procrustes [77] sur ces 2 jeux d'embeddings, en considérant comme un couple à rapprocher deux embeddings produits par deux encodeurs à partir d'un même extrait de parole.

La figure IV.8 schématise le protocole de cette expérience.

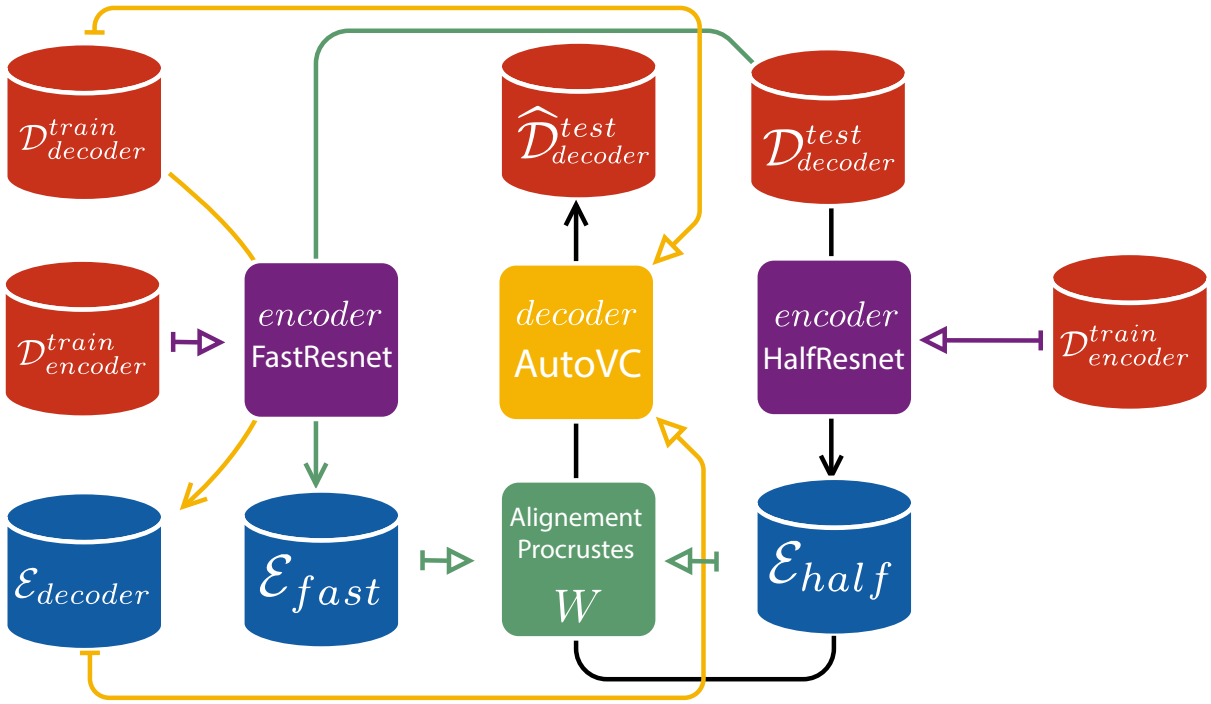

FIGURE IV.8 – Schéma résumant l'expérience IV.2.1.2.

IV.2.1.2 - B Les résultats Pour mesurer les performances optimales dans le cadre de cette attaque, on utilise comme jeu de test $\mathcal{D}^{test}_{decoder}$. Les résultats sont présentés dans la table IV.2, pour les EER_{cible} et EER_{source} et les SFAR.

On peut voir dans la table IV.2 que l'utilisation d'un alignement supervisé améliore encore les résultats de fraude du système : on observe jusqu'à 98% de SFAR à l'EER, contre 94.40% avec un alignement non supervisé, et 66.95% sans alignement. Cependant, ces résultats optimaux ne sont pas encore au niveau de ceux permis par un accès direct au décodeur attaqué.

IV.3 Conclusion du chapitre

Dans ce chapitre, nous avons présenté et exécuté une attaque de reconstruction de *templates* sous plusieurs variantes, sur des embeddings de tracés de chiffres manuscrits et de parole, en supposant un accès très limité aux encodeurs attaqués.

Dans le chapitre II, nous avons cherché à attaquer un système d'authentification biométrique en utilisant les embeddings d'enrôlement qu'il produisait. Que cela soit dans le cadre de systèmes d'authentification à partir de chiffres manuscrits ou d'extraits de parole, nous avons constaté que, sans informations sur les utilisateurs à leur origine, il est difficile de savoir à quel utilisateur appartient quel embedding.

En revanche, il est possible de reconstruire les embeddings afin de retrouver les données personnelles à leur origine, en exécutant ce qu'on appelle une *attaques de reconstruction de templates*. Ces attaques utilisaient des décodeurs, présentés dans le chapitre III, dont l'entraînement suppose un accès en boite noire à un encodeur. Tant que l'encodeur était le même que celui qui avait produit les embeddings, il n'y avais pas de problème, mais quand nous avons utilisé un encodeur différent (pour supposer une attaque sans accès à l'encodeur du système), alors le décodeur ne fonctionnait plus.

Notre hypothèse principale a été que les espaces de sortie de différents encodeurs n'étaient pas les mêmes, ce qui fait qu'un décodeur ne fonctionnait que sur l'espace d'embeddings sur lequel il avait été entraîné. Pour faire correspondre les espaces de sortie d'encodeurs différents, dans ce chapitre nous avons utilisé des **alignements**, supervisés ou non, visant à minimiser les distances entre groupes d'embeddings.

IV.3.1 L'alignement des embeddings de scripteur

Pour les embeddings produits pour la vérification du scripteur, nous avons montré, en utilisant l'algorithme de Wasserstein Procrustes [100], qu'il était possible de reconstruire les tracés de chiffres manuscrits à leur origine même sans avoir accès à l'encodeur qui les avait tracés, à condition de connaître l'architecture de ce dernier.

En revanche, la reconstruction n'était pas suffisamment précise pour que les tracés reconstruits puissent être considérés par le système de vérification comme appartenant à leur scripteur dans tous les cas. Nous pouvons donc conclure sur le cas du traitement des chiffres manuscrits en soulignant la vulnérabilité des embeddings, et l'importance de les protéger au même titre que les tracés de chiffres : comme des données personnelles.

IV.3.2 L'alignement des embeddings de locuteur

Pour les embeddings produits pour la vérification du locuteur, nous avons montré qu'il était possible de reconstruire de la parole prononcée de la même manière que le locuteur initial sans forcément contenir la même information linguistique, en utilisant des systèmes de conversion de voix. Cette reconstruction peut fonctionner sur d'autres encodeurs que celui auquel nous avons accès, mais les résultats sont fortement impactés sans l'utilisation d'un alignement pour projeter les embeddings à reconstruire dans le bon espace.

IV.3.3 D'autres utilisations des alignements entre embeddings

Mais l'utilisation d'un alignement entre groupes d'embeddings n'est pas limitée aux cas pour lesquels nous les utilisons dans cette thèse. Nous avons déjà vu qu'ils pouvaient être utilisés pour faire de la traduction automatique, comme dans Grave, Joulin et Berthet [100]. Dans le chapitre suivant, nous allons voir comment les utiliser afin de tester les limites d'un système de pseudo-anonymisation de la parole.

Chapitre V

L'anonymisation de la parole

Nous avons vu dans le chapitre précédent plusieurs techniques d'alignement entre groupes d'embeddings, supervisés ou non, utilisés dans le cadre d'attaques de reconstruction de *templates*.

Dans ce chapitre, nous allons utiliser des algorithmes d'alignement entre espaces d'embeddings pour vérifier la perméabilité d'un système d'anonymisation de la parole.

V.1 L'anonymisation de la parole

Nous avons vu dans le chapitre II que la parole est une donnée personnelle : elle contient des informations privées sur un utilisateur et peut être utilisée pour l'identifier, au même tire que son numéro de téléphone, une image de son visage ou son numéro de sécurité sociale (à des degrés divers de précision). À ce titre, suivant le Règlement Général de Protection des Données européen [1], elle doit être protégée. Nous avons choisi d'englober les informations personnelles contenues dans la voix dans le terme **information discriminante du locuteur** (voir section II.1.1), et les informations sur les mots utilisés dans le terme **informations linguistiques**. Il existe des cas d'utilisation où les seules informations pertinentes sont les informations linguistiques d'un extrait de parole, comme dans le cas de la transcription [57], [60], [102] ou de la traduction automatique [103]. Dans ces cas, pour des raisons de sécurité et de protection des données personnelles, il devient nécessaire d'utiliser un système qui puisse supprimer les informations discriminantes sans pour autant porter atteinte à l'intégrité des informations linguistiques. On appelle ces systèmes des *systèmes d'anonymisation de la parole*.

V.1.1 Les systèmes d'anonymisation de la parole

Un système d'anonymisation de la parole est un système qui transforme un extrait de parole pour en effacer les informations discriminantes du locuteur en minimisant les

pertes d'informations linguistiques.

V.1.1.0 - A Critères caractéristiques Le standard international ISO/IEC 24745, sur la protection des données biométriques [104] défini qu'un système d'anonymisation fonctionnel doit respecter deux critères :

1. La **Non-Associabilité** (*Un-linkability* en anglais) : on ne doit pas pouvoir lier une donnée anonymisée à sa contrepartie non anonymisée.
2. L'**Irréversibilité** (*Un-Reversibility* en anglais) : on ne doit pas pouvoir revenir à la contrepartie claire d'une donnée anonymisée uniquement à partir de cette dernière.

V.1.1.0 - B Les systèmes de pseudo-anonymisation sont des systèmes qui anonymisent des extraits de parole en modifiant l'information discriminante, tout en gardant une cohérence entre les extraits d'un même locuteur. Ainsi, un système de pseudo-anonymisation peut toujours être utilisé pour des tâches de reconnaissance du locuteur (voir section I.4.2) à partir des extraits anonymisées, mais il ne sera pas possible de remonter à l'identité du locuteur initial, uniquement de comparer un extrait de parole anonymisé avec le *template* anonymisé d'un locuteur.

Les systèmes de pseudo-anonymisation ont généré un intérêt croissant dans la communauté de la parole [67], pour pouvoir générer des jeux de données de parole respectueux de la vie privée. Cet intérêt s'est manifesté notamment par le lancement du Voice Privacy Challenge [105], [106], un challenge dont l'objectif est de produire le meilleur système de pseudo-anonymisation de la parole.

V.1.2 Le Voice privacy challenge

Le Voice Privacy Challenge, challenge bisannuel [105], [106] lancé en 2020, vise à dynamiser et standardiser la recherche dans l'anonymisation de la voix, en proposant un système de pseudo anonymisation garantissant à la fois des bonnes performances en vérification du locuteur et en transcription automatique de la parole.

V.1.2.0 - A La baseline du VPC2020 Ce Challenge propose une *baseline*, c'est-à-dire un kit de démarrage pour participer au challenge, composé d'un système de pseudo anonymisation et d'un système de vérification du locuteur déjà entraînés, ainsi qu'un protocole d'évaluation et les données nécessaires pour entraîner à nouveau le système de notre choix et évaluer l'ensemble.

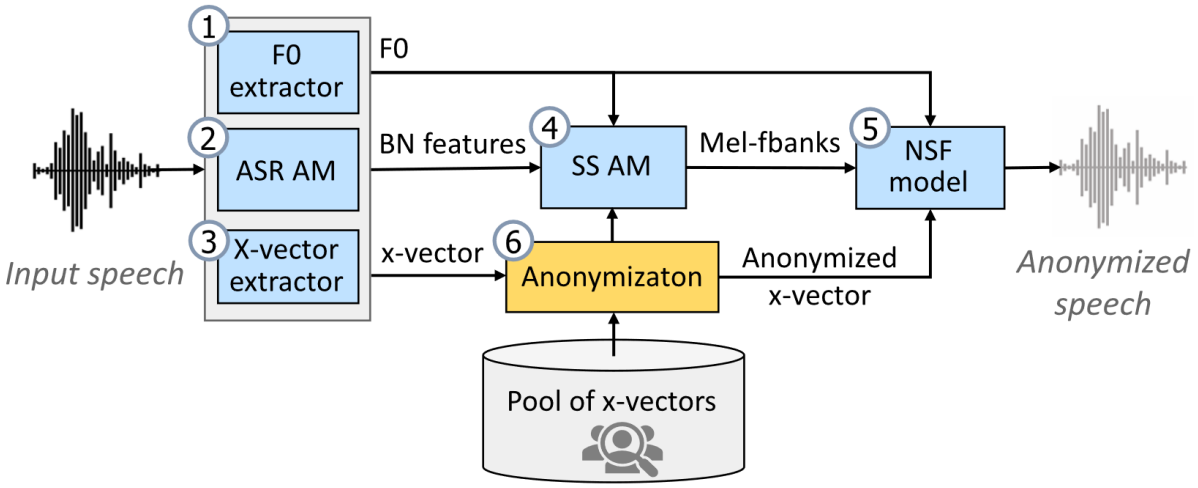

FIGURE V.1 – Schéma représentant l'architecture de la baseline 1 du Voice Privacy Challenge 2020 [106].

V.1.2.0 - B L'architecture est présentée dans la figure V.1. L'extrait de parole est d'abord traité par 3 systèmes différents, pour extraire :

1. Sa fréquence fondamentale (*F0*)
2. Ses informations linguistiques (*Bottle Neck Features*, ou BN)
3. Ses informations discriminantes, sous forme d'embedding (*x-vector*)

Le x-vector est ensuite modifié en utilisant une réserve de x-vectors, afin de le pseudo-anonymiser, puis la version modifiée est utilisée par un système de synthèse de voix (*SS AM*), avec les informations linguistiques et la F0 afin de produire des Mel-filterbanks (voir section I.4.1.1). Ces Mel-filterbanks sont finalement utilisé pour générer de la voix, à l'aide encore une fois du x-vector anonymisé et de la F0, en utilisant un Modèle de filtre neuronal (*NSF*).

Pour plus de détails sur les systèmes utilisés, se référer à la description complète : [106].

V.1.2.0 - C Les données Les jeux de données utilisés lors de ce challenge sont : VoxCeleb1 [34], VoxCeleb2 [35], VCTK [36], LibriSpeech [107] et LibriTTS [108].

VoxCeleb1 et 2 et VCTK ont déjà été présentés dans la section I.2.2. La répartition des données dans les différents jeux est fournie dans la table V.1

LibriSpeech [107] est un jeu de données produit pour l'entraînement de systèmes de reconnaissance automatique de la parole (ASR). Ce jeu contient 982h de parole annotée

(le texte prononcé est fourni), échantillonnée à 16kHz, lue par 2456 locuteurs à partir d'extraits de texte. Ces extraits sont répartis en sous-ensembles, parmi lesquels seuls LibriSpeech *test-clean*, *dev-clean*, *train-clean-100* et *train-other-500* seront utilisés ici.

LibriTTS [108] est un jeu dérivé de LibriSpeech, produit pour l'entraînement de systèmes dits *Text-To-Speech*, produisant de la parole à partir de texte. Il reprend les extraits et textes de Librispeech, mais en les optimisant pour la production de parole : échantillonnage à 24kHz, séparation des fichiers par phrases complètes, les textes non-normalisés sont inclus (avec la ponctuation et les majuscules) et le bruit de fond est éliminé. Au total, ce sont 585h de parole avec les textes associés, pour 2456 locuteurs qui sont inclus dans ce jeu. Seuls les sous-ensembles LibriTTS *train-clean-100* et *train-other-500* sont utilisés pour la baseline.

Toutes ces données sont réparties en 8 ensembles dont la répartition est montrée dans la table V.1.

TABLE V.1 – Tableau présentant les données utilisées pour former les différents jeux de données utilisés lors du *Voice Privacy Challenge 2020*. L'utilisation des différents jeux est détaillée dans les paragraphes V.1.2.0 - D et V.1.2.0 - E.

	Ensembles	Jeux utilisés	Usage	Locuteurs	Fichiers
Entraînement	$\mathcal{D}_{train}^{x-vector}$	VoxCeleb1&2	Entraînement	7 363	1 281 762
	$\mathcal{D}_{train}^{ASR\ AM}$	LibriSpeech train-clean-100	Entraînement	251	28 539
		LibriSpeech train-other-500	Entraînement	1 166	148 688
	$\mathcal{D}_{train}^{Dec}$	LibriTTS train-clean-100	Entraînement	247	33 236
	$\mathcal{D}_{train}^{Pool}$	LibriTTS train-other-500	Pool de x-vectors	1 160	205 044
Développement	$\mathcal{D}_{dev}^{ASR}$	LibriSpeech dev-clean	Enrôlement	29	343
			Essais	40	1 978
	$\mathcal{D}_{dev}^{ASV}$	VCTK dev	Enrôlement	30	600
			Essais (Commun)	30	695
			Essais (Différents)	30	10 677
Evaluation	$\mathcal{D}_{test}^{ASR}$	LibriSpeech test-clean	Enrôlement	29	438
			Essais	40	1 496
	$\mathcal{D}_{test}^{ASV}$	VCTK test	Enrôlement	30	600
			Essais (Commun)	30	70
			Essais (Différents)	30	10 748

V.1.2.0 - D Entraînement Les jeux $\mathcal{D}^{*}_{train}$ sont utilisés pour entraîner les différents systèmes qui composent la baseline :

- $\mathcal{D}^{x-vector}_{train}$ est utilisé pour entraîner le système x-vector.
- $\mathcal{D}^{ASR\ AM}_{train}$ est utilisé pour entraîner le système d'extraction des BN Features (*ASR AM*).
- $\mathcal{D}^{Dec}_{train}$ est utilisé pour entraîner les systèmes décodant l'information :
  - Le système de synthèse de voix (*SS AM*), qui produit des Mel-filterbanks à partir des BN, F0 et x-vector.
  - Le Modèle de filtre neuronal (*NSF*), qui produit de la parole à partir de Mel-filterbanks, F0 et x-vector.
- $\mathcal{D}^{Pool}_{train}$ est utilisé pour constituer un ensemble de x-vectors (La *Pool*) qui servira à l'algorithme d'anonymisation.

Les différents systèmes qui composent la baseline sont présentés dans la figure V.1. Les différents systèmes sont évalués pendant et après entraînement par des jeux de données différents, présentés dans le paragraphe suivant.

V.1.2.0 - E Évaluation Les jeux $\mathcal{D}^{*}_{dev}$ et $\mathcal{D}^{*}_{test}$ seront eux utilisés pour simuler l'enrôlement d'utilisateurs puis leurs tentatives d'authentification (essais). Les jeux $\mathcal{D}^{ASR}_{dev}$ et $\mathcal{D}^{ASR}_{test}$ seront respectivement utilisés pour évaluer les capacités du système à reconstruire de la parole *audible* pendant et après l'entraînement. La parole est considérée *audible* si elle est bien transcrite par un système de transcription automatique indépendant. Le système de transcription automatique utilisé ici est un système de *Kaldi Speech Recognition Toolkit* [109], entraîné sur LibriSpeech train-clean-360. Les jeux $\mathcal{D}^{ASV}_{dev}$ et $\mathcal{D}^{ASV}_{test}$ seront respectivement utilisés pour évaluer les capacités du système à reconstruire de la parole *pseudo-anonymisée*.

V.1.2.0 - F Les différents scénarii du challenge Pour évaluer les performances du système d'anonymisation du challenge, plusieurs scénarios sont proposés :

1. *La Baseline* : les x-vectors clairs sont comparés à d'autres x-vectors clairs pour mesurer l'efficacité du système en vérification du locuteur.
2. **L'attaquant naïf** : les x-vectors anonymisés sont comparés à des x-vectors clairs pour mesurer l'efficacité du système d'anonymisation.

3. **L'attaquant peu informé** : on suppose que l'attaquant a accès au système d'anonymisation, et peut l'utiliser pour anonymiser ses données. Les x-vectors anonymisés sont donc comparés à d'autres x-vectors anonymisés.
4. **L'attaquant semi-informé** : on suppose en plus que l'attaquant utilise ses données anonymisées pour ré-entraîner son encodeur, et on compare deux groupes de x-vectors anonymisés extraits par ce nouvel encodeur.

Ces différents scénarii permettent d'évaluer le fonctionnement du système de manière plus complète. Les résultats de la baseline du challenge face à ces 4 scénarii sont présentés lignes 1 à 4 du tableau V.2 en terme d'EER (voir section I.3).

V.2 L'inversion d'un système d'anonymisation de la parole

Dans la lignée des attaques réalisées au cours de cette thèse, nous avons cherché à tester les limites d'un système d'anonymisation, sur les deux critères qui le définissent : sa *non-associabilité* et son *irréversibilité*.

Nous avons vu dans le chapitre IV plusieurs techniques pour aligner des ensembles d'embeddings. Ici, nous allons utiliser ces techniques pour aligner un ensemble d'embeddings anonymisés et non anonymisés, pour savoir s'il est possible de retrouver l'équivalent non-anonymisés d'embeddings anonymisés.

Dans les expériences suivantes, nous considérons le système baseline du Voice Privacy Challenge 2020 [106], présenté section V.1.2,

V.2.1 Les mesures de succès d'une attaque

Dans cette section, nous allons présenter les métriques utilisées pour mesurer la *associabilité* et la *réversibilité* d'un système d'anonymisation pendant une attaque.

V.2.1.1 La Top 1 accuracy comme mesure de la réversibilité

La Top k accuracy est une métrique présentée section I.3.1, ici on va détailler son utilisation comme mesure de la réversibilité pour $k = 1$.

V.2.1.1 - A Mesure de la réversibilité d'embeddings de locuteurs Soit un ensemble d'embeddings $\mathcal{E}$ de locuteurs $\mathcal{U}$, un système d'anonymisation A et un système de dés-anonymisation D, on a alors le jeu d'embeddings anonymisés puis dés-anonymisés défini comme : $\widehat{\mathcal{E}} = D(A(\mathcal{E}))$. On peut alors comparer chaque embedding de $\widehat{\mathcal{E}}$ contre chaque embedding de $\mathcal{E}$, pour trouver les couples d'embeddings les plus proches. Soit $x_u \in \mathcal{E}$ un embedding produit respectivement par les locuteurs $u \in \mathcal{U}$, on pose g la fonction qui donne l'embedding de $\widehat{\mathcal{E}}$ le plus proche : $g : x_u \mapsto y_v$. On défini alors la *Top 1 accuracy* comme :

$$Top1Accuracy(A, D, \mathcal{E}) = 100 \times \frac{Card(\{x_u \in \mathcal{E} \mid g(x_u) = D(A(x_u))\})}{Card(\mathcal{E})} \tag{V.1}$$

Cette métrique sert à vérifier si les embeddings anonymisés puis dés-anonymisés sont assez proches de leur contrepartie n'ayant jamais été anonymisée.

V.2.1.1 - B Top 1 Speaker Accuracy Une dérivée est la Top 1 accuracy pour les locuteurs, où on cherche la proportion d'embeddings dont la contrepartie dés-anonymisée la plus proche possède le même locuteur, appelée *Top 1 Speaker Accuracy*. Soit $y_v \in \widehat{\mathcal{E}}$ l'embedding renvoyé par la fonction g à partir de x_u, et $v \in \mathcal{U}$ son locuteur associé, on pose $h : x_u \mapsto u$ la fonction qui renvoie le locuteur associé a un embedding, et on définit la *Top 1 Speaker Accuracy* comme :

$$Top1SpeakerAccuracy(A, D, \mathcal{E}) = 100 \times \frac{Card(\{x_u \in \mathcal{E} \mid h(g(x_u)) = h(D(A(x_u)))\})}{Card(\mathcal{E})} \tag{V.2}$$

Cette métrique a été présentée pour la première fois dans [110] et sera surtout utilisée dans les expériences V.2.2, V.2.3 et V.2.4.

V.2.1.2 L'EER comme mesure de l'associabilité

Comme présenté dans la section I.3.2, l'EER est une mesure de l'efficacité d'un système d'authentification. Cette mesure est représentative de la répartition d'un groupe d'embeddings dans l'espace, par rapport aux utilisateurs qui leurs sont associés : Si les embeddings appartenant à un même utilisateur sont similaires entre eux, et que les embeddings d'utilisateurs différents sont moins semblables, alors le système est considéré plus sécurisé et l'EER sera bas.

Dans le cadre d'un système de pseudo-anonymisation de la parole, on peut utiliser

l'EER pour comparer des embeddings originaux et des embeddings anonymisés. Pour chaque couple composé d'un embedding de chaque groupe, on peut utiliser une fonction de similarité pour calculer un score. Si ce score est plus haut pour les embeddings de même locuteur que pour les autres, alors non seulement on peut retrouver facilement les locuteurs associés à chaque embedding anonymisé, mais l'EER calculé sera aussi très bas. Au contraire, si le score est indépendant des locuteurs les ayant produits, alors l'EER sera haut, et les embeddings anonymisés seront totalement dissociés de leur contrepartie. Nous pouvons donc utiliser l'EER comme une mesure de l'associabilité d'un système d'anonymisation.

V.2.2 Alignement supervisé d'embeddings avec Procrustes

Dans cette expérience, nous voulons attaquer le système d'anonymisation proposé par le Voice Privacy Challenge 2020 en utilisant un alignement supervisé entre un ensemble de x-vectors anonymisés et non-anonymisés. L'objectif est de trouver une transformation réversible entre les x-vectors clairs et anonymisés, pour permettre de retrouver des informations sur les locuteurs d'extraits de parole anonymisés. Ici la transformation réversible sera une rotation, calculée en utilisant un alignement supervisé : l'*Analyse de Procrustes* [77].

On se place du point de vue d'un attaquant ayant un accès en boite noire à un système de pseudo-anonymisation pré-entraîné : la baseline du VPC2020, ainsi qu'à son encodeur de x-vectors. Cet attaquant s'est procuré une base de données d'extraits de voix anonymisés, et souhaite connaître l'identité des différents locuteurs composant cette base. On suppose également que l'attaquant aura trouvé une base de données contenant des extraits de voix **non-anonymisées** appartenant aux mêmes utilisateurs, sans aucun extrait en commun.

Ces hypothèses sont celles de *l'attaquant peu informé* (*lazy-informed attacker*), tel que présenté dans les différents scénarii proposés par le VPC2020. Pour aller plus loin, on pourrait aussi supposer que l'attaquant utilise un extracteur de x-vectors qui soit entraîné en utilisant également des données anonymisées. Ce scénario est appelé celui de *l'attaquant semi-informé* (*semi-informed attacker*) dans le VPC2020. Nous présenterons les résultats correspondants aux deux hypothèses dans le tableau V.2.

V.2.2.0 - A Les Données Parmi les données présentées dans le tableau V.1 :

- $\mathcal{D}_{dev}^{ASV}$ sera utilisé sous l'alias $\mathcal{D}_{cible}$ pour produire les données anonymisées ciblées par l'attaquant : $\mathcal{D}_{cible}^{anon}$.
- $\mathcal{D}_{test}^{ASV}$ sera utilisé sous l'alias $\mathcal{D}_{attaque}$ pour son attaque.

Leur utilisation est détaillée dans le protocole ci-dessous

V.2.2.0 - B Le Protocole Ayant accès à l'encodeur et au système VPC, on va calculer les x-vectors liés aux données claires et pseudo-anonymisées. Soit $\mathcal{E}_{cible}^{anon} \in \mathbb{R}^{n_1 \times d}$ le jeu de n_1 x-vectors anonymisés produits à partir de $\mathcal{D}_{cible}^{anon}$. Soit $\mathcal{D}_{attaque}^{anon}$ la contrepartie anonymisée du jeu $\mathcal{D}_{attaque}$, et $\mathcal{E}_{attaque}^{anon} \in \mathbb{R}^{n_2 \times d}$ et $\mathcal{E}_{attaque} \in \mathbb{R}^{n_2 \times d}$ leurs ensembles de x-vectors respectifs. Utilisant la baseline du VPC2020, le nombre de dimensions des x-vectors est fixé à $d = 512$.

On connaît les x-vectors avant et après anonymisation pour le jeu d'attaque, ainsi que la correspondance un à un entre les deux ensembles, on peut donc aligner les deux ensembles avec une analyse de Procrustes [77], ce qui nous donne la matrice de rotation $\mathcal{W}_P \in \mathbb{R}^{d \times d}$. Une fois cette rotation calculée, on va pouvoir utiliser son inverse ($\mathcal{W}_P^T$, car une rotation est inversible, et son inverse est sa transposée) pour projeter l'ensemble $\mathcal{E}_{cible}^{anon}$ dans l'espace des x-vectors non anonymisés, et obtenir $\widehat{\mathcal{E}}_{cible} \in \mathbb{R}^{n_1 \times d}$.

Le protocole est schématisé dans la figure V.2.

V.2.2.0 - C Des variations possibles Pour cette expérience, nous avons proposé 3 variations possibles du protocole initial, visant à améliorer les résultats :

1. L'utilisation d'encodeurs de x-vectors **genrés**, c'est-à-dire qu'on utilise un encodeur différent pour les hommes et les femmes, et chacun est entraîné uniquement sur des données du genre choisi, ce qui permet d'améliorer les performances de l'extracteur sur chaque genre.
2. L'utilisation d'une technique de réduction dimensionnelle : l'**ACP** avant d'appliquer l'alignement, afin d'ordonner et de réduire le nombre de dimensions à 70 (par rapport aux 512 initiales), et ainsi faciliter l'alignement. Le nombre de 70 a été choisi car c'est le nombre de dimensions à partir du quel la variance expliquée pour chaque nouvelle dimension devient négligeable.
3. L'utilisation d'extracteurs de x-vectors **ré-entraînés** sur des données anonymisées, afin d'améliorer leurs performances sur les données anonymisées.

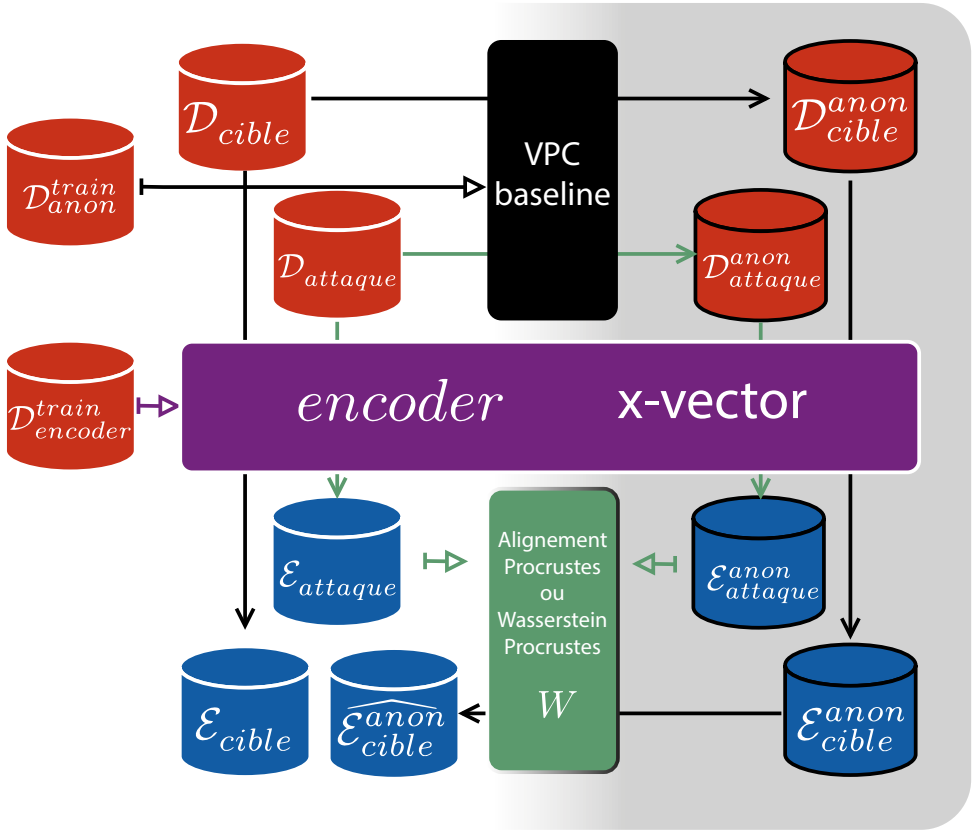

FIGURE V.2 – Schéma représentant le protocole des expériences V.2.2 et V.2.3.

V.2.2.0 - D Les Résultats Une fois $\widehat{\mathcal{E}}_{cible}$ obtenu, on va pouvoir le comparer aux jeux de x-vectors clairs pour tester son *irréversibilité* et sa *non-associabilité*.

Pour mesurer la *non-associabilité*, on va utiliser l'*EER* (voir section I.3) en confrontant $\widehat{\mathcal{E}}_{cible}$ et $\mathcal{E}_{attaque}$, avec les couples fournis par le protocole d'évaluation du VPC2020, afin de rester cohérent avec les résultats antérieurs. Pour mesurer l'*irréversibilité*, on va regarder si les x-vectors dés-anonymisés sont assez proches de leurs équivalents clairs, en utilisant la Top 1 accuracy sur les locuteurs (voir section I.3) entre $\widehat{\mathcal{E}}_{cible}$ et $\mathcal{E}_{cible}$.

Les résultats sont présentés dans les lignes 5 à 12 du tableau V.2, pour les différentes variations proposées à cette expérience. On remarque que les meilleurs résultats sont obtenus lorsque toutes les variations proposées sont utilisées. On remarque également qu'on obtient de meilleurs résultats en terme d'EER que la baseline, que ce soit pour pour le scénario *semi-informé* (14.6%/13.1% pour les Femmes/Hommes au lieu de 17.1%/14.1%) ou pour le scénario *peu informé* (25.4%/24.4% pour les Femmes/Hommes, au lieu de 29.4%/29.1%). Ces résultats restent assez éloignés des EER qu'on aurait obtenus pour des données non anonymisées (10.3%/2.9%), mais s'en rapprochent tout de même. Enfin, dans le meilleurs des cas, on arrive à retrouver les locuteurs à l'origine de **60%** des x-vectors.

On montre donc qu'il est possible d'attaquer les propriétés d'un système d'anonymisation en utilisant un algorithme d'alignement supervisé sur ses x-vectors. Cependant, cette méthode suppose de connaître les liens entre x-vectors anonymisés et non anonymisés

V.2.3 Alignement non supervisé d'embeddings avec Wasserstein Procrustes

Dans cette expérience, on se place du point de vue d'un attaquant voulant attaquer la baseline du Voice Privacy Challenge 2020, dans les mêmes conditions que l'expérience précédente : L'attaquant a un accès en boite noire à un système de pseudo-anonymisation pré-entraîné : la baseline du VPC2020, ainsi qu'à son encodeur de x-vectors. Il s'est procuré une base de données d'extraits de voix anonymisés (dite "base de données d'attaque"), et souhaite connaître l'identité des différents locuteurs composant cette base. On suppose également que l'attaquant aura trouvé une base de données contenant des extraits de voix **non-anonymisés** appartenant aux mêmes utilisateurs, sans aucun extrait en commun. **Cependant**, et c'est la seule différence dans nos hypothèses de départ avec l'expérience précédente, on suppose qu'après anonymisation de la base de données d'attaque, l'atta-

quant ne connait pas la correspondance entre les extraits clairs et anonymisés.

V.2.3.0 - A Le Protocole On réutilisera les mêmes données $\mathcal{D}_{cible}$ et $\mathcal{D}_{attaque}$ que dans l'expérience V.2.2. De la même manière, on calcule $\mathcal{E}^{anon}_{attaque}$ et $\mathcal{E}_{attaque}$ à l'aide de la baseline du VPC2020. Ensuite, ne connaissant pas le lien entre les embeddings des deux jeux, on utilisera l'algorithme d'alignement **non supervisé** de Wasserstein Procrustes [100] (voir section IV.1.2.3) pour calculer la matrice de rotation $\mathcal{W}_{WP} \in \mathbb{R}^{d\times d}$. Une fois cette rotation calculée, on va pouvoir utiliser son inverse pour projeter l'ensemble $\mathcal{E}^{anon}_{cible}$ dans l'espace des x-vectors non anonymisés, et obtenir $\widehat{\mathcal{E}}_{cible} \in \mathbb{R}^{n_1\times d}$.

Le Protocole étant le même que celui de l'expérience V.2.2, mis à part l'alignement, la figure V.2 reste valable pour le décrire.

V.2.3.0 - B Les Résultats Une fois $\widehat{\mathcal{E}}_{cible}$ calculé, on va pouvoir calculer les performances de l'attaque en terme d'*EER* et de *Top 1 accuracy*, en le comparant avec les jeux $\mathcal{E}_{attaque}$ et $\mathcal{E}_{cible}$, respectivement pour calculer la *non-associabilité* et l'*irréversibilité* du système après l'attaque.

Les résultats sont présentés pour les 3 variations proposées (avec un encodeur *genré*, *ré-entraîné* et/ou en utilisant une *ACP*), dans les lignes 13 à 20 du tableau V.2. On peut observer que de manière non supervisée, les résultats obtenus sont très proches de ceux présentés à l'expérience V.2.2 : Pour l'EER, on a 14.0%/13.2% (pour les Femmes/Hommes) au lieu de 14.6%/13.1%, et pour la Top 1 accuracy on a 57.4%/62.1% au lieu de 59.8%/60.0%.

On en déduit que même avec un alignement non supervisé, on arrive à inverser un système d'anonymisation avec une rotation. Pour aller plus loin, nous avons cherché la limite théorique que nous pouvions atteindre pour l'attaque d'un système d'anonymisation par rotation.

V.2.4 Alignements optimaux d'embeddings

Dans cette expérience, nous cherchons le résultat théorique optimal, pour les métriques choisies, qu'un attaquant puisse obtenir en utilisant une rotation pour attaquer le système d'anonymisation du Voice Privacy Challenge 2020.

Pour atteindre ce meilleur résultat théorique, on va se place du point de vue d'un attaquant "*oracle*", c'est-à-dire : qui aurait accès à toutes les informations qu'il pourrait souhaiter. On va donc directement calculer notre alignement sur le jeu de données ciblé, sans utiliser de jeu d'attaque. Dans les expériences précédentes, nous entraînions notre

alignement sur un jeu d'attaque avant de le tester sur le jeu cible, ici on va mesurer les performances de l'attaque directement sur le jeu cible.

V.2.4.0 - A Les Données Parmi les données présentées dans le tableau V.1, $\mathcal{D}_{dev}^{ASV}$ sera utilisé sous l'alias $\mathcal{D}_{cible}$ pour produire les données anonymisées ciblées par l'attaquant : $\mathcal{D}_{cible}^{anon}$. Leur utilisation est détaillée dans le protocole ci-dessous

V.2.4.0 - B Le Protocole Ayant accès à l'encodeur et au système VPC, on va calculer les x-vectors liés aux données claires et pseudo-anonymisées. Soit $\mathcal{E}_{cible}^{anon} \in \mathbb{R}^{n_1 \times d}$ le jeu de n_1 x-vectors anonymisés produits à partir de $\mathcal{D}_{cible}^{anon}$. Utilisant la baseline du VPC2020, le nombre de dimensions des x-vectors est fixé à $d = 512$.

On connaît les x-vectors avant et après anonymisation pour le jeu d'attaque, ainsi que la correspondance une à une entre les deux ensembles, on peut donc aligner les deux ensembles avec une analyse de Procrustes [77], ce qui nous donne la matrice de rotation $\mathcal{W}_{OP} \in \mathbb{R}^{d \times d}$ (O pour Oracle, car c'est la matrice calculée directement à partir du jeu cible). De la même manière, on peut aussi utiliser l'algorithme de Wasserstein Procrustes [100] pour calculer de manière non supervisée une matrice de rotation $\mathcal{W}_{OWP} \in \mathbb{R}^{d \times d}$. Une fois ces rotations calculées, on va pouvoir utiliser leurs inverses pour projeter l'ensemble $\mathcal{E}_{cible}^{anon}$ dans l'espace des x-vectors non anonymisés, et obtenir $\widehat{\mathcal{E}}_{cible} \in \mathbb{R}^{n_1 \times d}$.

Dans cette expérience, on a voulu mesurer les performances optimales du système, nous n'avons donc utilisé que les variations qui nous avaient donné les meilleurs résultats précédemment :

- L'extracteur de x-vectors a été *ré-entraîné* sur des données anonymisées.
- L'extraction est *genrée*.
- Une *ACP* a été appliquée aux x-vectors avant de les aligner.

Le protocole est schématisé dans la figure V.3.

V.2.4.0 - C Les Résultats Une fois $\widehat{\mathcal{E}}_{cible}$ calculé, on va pouvoir calculer les performances de l'attaque en terme d'*EER* et de *Top 1 accuracy*, en le comparant avec le $\mathcal{E}_{cible}$, pour calculer la *non-associabilité* et l'*irréversibilité* du système après l'attaque.

Les résultats sont présentés pour un alignement supervisé et non supervisé respectivement dans les lignes 21 et 22 du tableau V.2.

On peut observer que quelle que soit la méthode d'alignement, on peut obtenir une très bonne *réversibilité* : tous les résultats sont entre 98.0% et 99.0% pour la *Top 1 accuracy*.

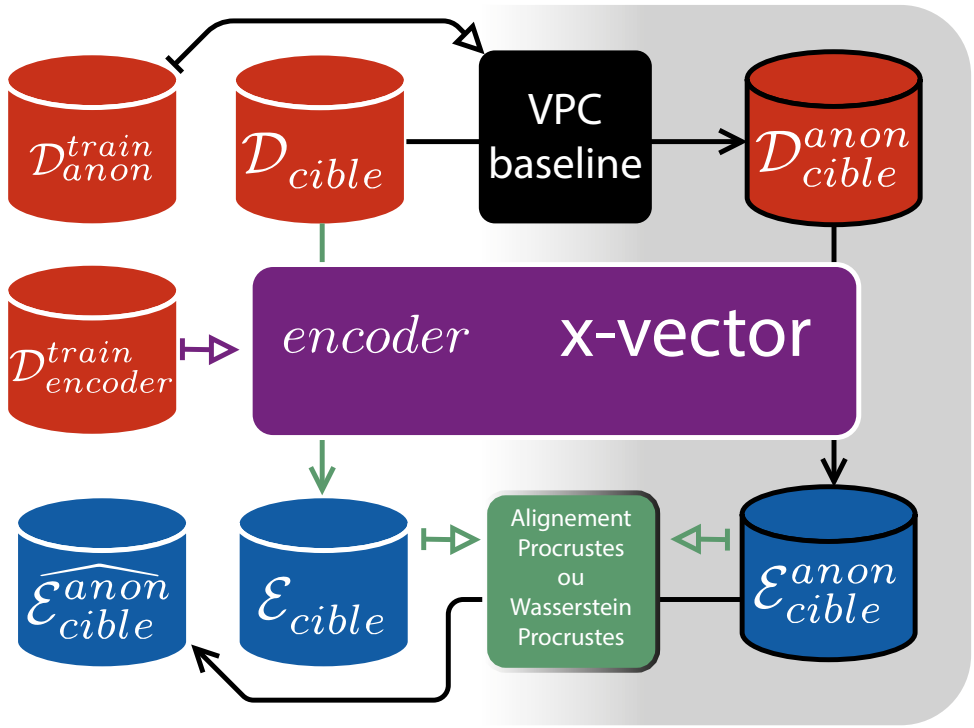

FIGURE V.3 – Schéma représentant le protocole de l'expérience V.2.4.

Pour l'EER, on descend à 12.1%/8.7%, ce qui se rapproche beaucoup de l'*associabilité* obtenue sur les données claires : 10.3%/2.9%.

On montre donc ici qu'il est possible, uniquement avec une rotation, d'inverser un système d'anonymisation. Le problème qui suivra sera donc de minimiser la différence des performances entre l'ensemble d'entraînement et de test.

Les travaux présentés dans cette section ont tous été réalisé en collaboration avec un autre doctorant : Pierre CHAMPION. Ces travaux ont fait l'objet d'une publication [110] dans la conférence "*IEEE Automatic Speech Recognition and Understanding*", en 2021.

TABLE V.2 – Tableau présentant les résultats des expériences V.2.2, V.2.2 et V.2.4. Les lignes 1-4 correspondent aux résultats donnés par la baseline du challenge, sur des données non-anonymisées et pour les 3 scénarii proposés. Les lignes 5-12, 13-20 et 21-22 correspondent aux différentes expériences, avec leurs variations (utilisation d'**ACP**, d'extracteurs **genrés** et/ou **ré-entraînés**. Les meilleures performances pour sont indiquées en **gras**.

		Genré	ACP	x-vector extracteur	$EER \downarrow$ F	$EER \downarrow$ H	$Top1 Acc. \uparrow$ F	$Top1 Acc. \uparrow$ H
1	Données claires			original	10.3%	2.9%		
2	Attaquant naïf			original	49.0%	42.6%		
3	Att. peu informé			original	**29.4**%	**29.1**%		
4	Att. semi-informé			ré-entraîné	**17.1**%	**14.1**%		
5	Exp. V.2.2 - W_P			original	41.9%	30.6%	25.5%	36.6%
6		✓		original	40.1%	31.0%	26.6%	45.8%
7			✓	original	32.6%	32.7%	27.1%	40.8%
8		✓	✓	original	**25.4**%	**24.4**%	**30.5**%	**50.0**%
9				ré-entraîné	29.0%	23.6%	58.7%	59.2%
10		✓		ré-entraîné	27.0%	21.1%	54.8%	56.6%
11			✓	ré-entraîné	21.5%	23.1%	51.9%	57.0%
12		✓	✓	ré-entraîné	**14.6**%	**13.1**%	**59.8**%	**60.0**%
13	Exp. V.2.3 - W_{WP}			original	43.6%	33.4%	25.2%	22.1%
14		✓		original	40.7%	35.9%	26.6%	20.9%
15			✓	original	36.3%	35.4%	**24.1**%	38.6%
16		✓	✓	original	**26.4**%	**25.2**%	**24.1**%	**39.4**%
17				ré-entraîné	31.4%	24.2%	57.2%	60.2%
18		✓		ré-entraîné	28.6%	24.0%	48.6%	47.1%
19			✓	ré-entraîné	21.6%	23.6%	48.5%	**62.1**%
20		✓	✓	ré-entraîné	**14.0**%	**13.2**%	**57.4**%	61.4%
21	Exp. V.2.4 - W_{OP}	✓	✓	ré-entraîné	12.1%	8.7%	98.8%	98.0%
22	Exp. V.2.4 - W_{OWP}	✓	✓	ré-entraîné	13.1%	10.0%	99.0%	98.4%

V.3 Conclusion du chapitre

Dans ce chapitre, nous avons examiné l'utilisation d'alignements pour inverser un système de pseudo-anonymisation de la parole.

Nous avons montré qu'un système de pseudo-anonymisation comme celui de la baseline du Voice Privacy Challenge pouvait être approché par une transformation simple et inversible, comme une rotation, entre les espaces des x-vectors correspondants à des segments de parole clairs et anonymisés.

Les résultats obtenus montrent qu'il y a toujours une différence notable entre les résultats avec une rotation calculée à partir d'autres données ou avec un algorithme oracle, ayant accès à toutes les données. Et même avec une connaissance parfaite de toutes les données, une rotation n'est pas suffisante pour parfaitement inverser un système d'anonymisation. Il sera donc nécessaire à l'avenir de passer par des systèmes inversibles plus complexes, comme les flows, expliqués section III.1.3.

Chapitre VI

Conclusions et Perspectives

Nous utilisons quotidiennement une gamme grandissante de services numériques via une variété d'appareils numériques. Ces services - de communications, bancaires, professionnels, de divertissement - sont pour la plupart assurés par un fournisseur distant, qui doit s'assurer que l'utilisateur est bien celui qu'il prétend être. Malheureusement, il est impossible de s'assurer totalement de l'identité d'un utilisateur distant, aucun moyen d'authentification n'est sûr à 100%. Comme le fournisseur de service recherche la plus grande sécurité possible, il a alors plusieurs possibilités :

- Améliorer la sécurité de chaque moyen d'authentification (demander des mots de passe plus complexes, améliorer les performances du système de vérification biométrique)
- Ajouter des moyens d'authentification (mémoriels, matériels et biométriques)

Chaque moyen d'authentification est basé sur des données propres à l'utilisateur : ses connaissances (*authentification mémorielle*), ses possessions (*authentification matérielle*) ou son corps (*authentification biométrique*).

Dans le cadre de l'authentification biométrique, ces données sont des données personnelles, protégées par le RGPD [1]. En effet, non seulement vos données biométriques (visage, empreintes digitales, parole, écriture, démarche...) permettent de vous identifier, mais elles sont également très difficiles ou impossibles à changer en cas de vol, au contraire d'un mot de passe ou d'une clé. La protection de ces données est non seulement une contrainte légale et éthique pour le fournisseur de service, mais également un besoin pour l'utilisateur de ces services : personne ne veut se faire voler ses données biométriques.

La protection de ces données pose plusieurs questions :

1. Est-il possible de voler ces données ?
2. Est-il possible d'utiliser des données volées pour frauder un système d'authentification ?
3. Comment se défendre face à ces menaces ?

Pour savoir si il est possible de voler des données, nous nous sommes placés du point de vue d'un attaquant cherchant à les voler. En se basant sur la manière dont les systèmes d'authentification biométrique à base d'embeddings fonctionnent, nous avons considéré que les embeddings d'enrôlement pouvaient représenter une faille et décidé de nous placer dans le cas où un attaquant aurait volé un jeu de ces embeddings et chercherait à reconstruire ces données pour ensuite frauder un système d'authentification.

La défense face à ces attaques n'est pas traitée dans cette thèse, mais il existe un challenge visant à mettre en place cette défense pour des systèmes de vérification du locuteur : SASV 2022 [32]. Nous avons participé au challenge mais sans que nos résultats ne soient suffisamment significatifs pour permettre une publication. Sur le jeu d'évaluation du challenge, et en utilisant le système de vérification du locuteur fourni, nous avons réussi à descendre à un EER de 0.56% pour la détection d'attaques de fraude variées, et un EER de 1.47% pour la vérification conjointe du locuteur et de la fraude. Ces performances nous ont placés 13e/23, sans innovation majeure.

Nous avons donc décidé de nous concentrer sur la reconstruction de données à partir d'embeddings, spécifiquement pour deux biométries comportementales : les tracés de chiffres manuscrits et la parole. Les systèmes ciblés étaient des systèmes d'authentification (du scripteur et du locuteur), mais également un système de pseudo-anonymisation de la parole également basé sur des embeddings de parole.

Les défis posés par l'interprétation de ces embeddings sont multiples :

- La reconstruction de données à partir d'embeddings n'avait été faite que sur des images de visages ou d'empreintes digitales, il a donc fallu trouver un moyen de reconstruire une donnée temporelle, ne contenant pas que l'information discriminante de l'utilisateur mais également les information indépendantes :
  - L'information linguistique pour la parole.
  - L'information du chiffre pour les tracés de chiffres.
- La reconstruction des embeddings a toujours été conditionnée à un accès à l'encodeur qui les avait produit, nous avons donc exploré les solutions permettant d'assurer cette reconstruction avec un accès à un autre encodeur - un encodeur leurre.
- Les embeddings produits par les deux encodeurs étant différents, nous avons dû trouver une solution pour passer d'un espace d'embeddings à l'autre, en utilisant des techniques supervisées ou non supervisées en fonction des hypothèses considérées pour différentes attaques.

Nous allons développer dans la section suivante les contributions produites au cours de cette thèse et les résultats associés, puis nous aborderons les limites de ces résultats, avant de conclure sur les futurs travaux et les perspectives ouvertes par cette thèse.

VI.1 Contributions et résultats

VI.1.1 De l'attaque d'un système de vérification du scripteur

Pour répondre à ces problématiques, nous avons d'abord considéré un système de vérification du scripteur. Ce système extrait à partir de tracés de chiffres manuscrits (séquences de points en 2 dimensions) des embeddings discriminants l'identité du scripteur et le chiffre tracé, à l'aide d'un réseau de neurones **Bi-LSTM**. En fonctionnement normal, ce système collecte un ensemble d'embeddings d'enrôlement de la part des utilisateurs du système, puis lors d'une demande d'authentification, il compare un nouvel embedding (produit à partir de nouveaux chiffres produits pour la tentative d'authentification) aux embeddings précédemment enregistrés et prend la décision d'accepter l'utilisateur si l'embedding d'authentification est assez similaire aux embeddings d'enrôlement précédemment enregistrés. La similarité étant mesurée par un score de similarité calculé sur une paire d'embeddings, ici un produit scalaire entre deux embeddings de norme 1.

En se plaçant du point de vue d'un attaquant s'étant procuré le jeu des embeddings d'enrôlement de ce système, nous avons souhaité savoir à quel chiffre était associé chacun des embeddings. Pour ce faire, un clustering en 10 groupes sur les embeddings volés a permis de les grouper par chiffre (à 98.96% de précision), puis un alignement avec un jeu d'embeddings produit par un autre encodeur à partir de données d'utilisateurs disjoints a permis de retrouver la valeur des chiffres associés à chaque cluster (dans 100% des cas). Ces résultats sont présentés en détail dans le chapitre II.

Ensuite, nous avons entraîné un décodeur à l'architecture LSTM-MDN à reconstruire des chiffres à partir d'embeddings, en n'ayant accès qu'à un encodeur leurre. Les performances du décodeur sur les embeddings produits par l'encodeur leurre ont permis une fraude du système de vérification jusqu'à 97.15% des cas. Pour le faire fonctionner avec l'encodeur attaqué, nous avons d'abord aligné les espaces de sortie des deux encodeurs avec un alignement non supervisé, ce qui nous a permis de frauder le système attaqué jusqu'à 54.64% des cas. Après avoir essayé plusieurs alignements, celui qui nous a permis d'obtenir les meilleurs résultats était celui de Wasserstein-Procrustes, tiré de la littéra-

ture en traduction automatique, il a été initialement conçu pour calculer un alignement non supervisé pour associer deux ensembles d'embeddings de mots, dans deux langues différentes. Ces résultats sont présentés en détail dans le chapitre III.

Nous avons donc montré qu'il était possible d'extraire une information sur les utilisateurs à partir d'embeddings de scripteurs de manière non supervisée, et sans accès à leur encodeur.

VI.1.2 De l'attaque d'un système de vérification du locuteur

Après avoir mesuré ces performances sur les chiffres manuscrits, nous avons décidé de faire la comparaison avec un système d'authentification différent : la vérification du locuteur. Ces systèmes sont :

- Plus précis (moins de 1% d'EER pour la vérification du locuteur contre 6.5% pour l'état de l'art sur la vérification du scripteur).
- Plus utilisés dans la littérature scientifique et dans l'industrie.
- Fonctionnent aussi avec une donnée plus complexe (l'information à analyser peut se trouver dans le domaine temporel et fréquentiel).
- Contiennent des informations indépendantes du locuteur plus variées (les informations linguistiques contenues dans un extrait de parole sont beaucoup plus variées qu'un chiffre entre 0 et 9).

Le premier problème rencontré est venu de la complexité de l'information linguistique : impossible de retrouver le contenu linguistique d'un embedding en utilisant notre méthode de clustering, car il aurait fallu posséder une liste de tous les contenus possibles, et être assuré que les deux jeux d'embeddings contiennent une quantité suffisante de contenus en commun. Par exemple, si tous les extraits ne contenaient qu'un mot, appartenant à une liste connue de N mots, et que nous étions sûrs que chaque jeu d'embedding possède assez d'exemples pour chacun de ces mots, alors nous aurions pu retrouver à quel mot correspondaient quels embeddings. Ce genre de scénario est très spécifique et donc difficile à réaliser en pratique. Nous sommes donc directement passés à la reconstruction des extraits de paroles à partir de leurs embeddings.

Le deuxième problème est venu du contenu des embeddings : ils sont produits par un système dont l'objectif est d'extraire uniquement l'information discriminante du locuteur, ne laissant que peu de place pour l'information linguistique. Nous avons donc utilisé un système de conversion de voix. Ce système utilise l'embedding d'un locuteur cible et

un extrait de parole source pour construire un extrait de parole contenant l'information linguistique de la source mais prononcée à la manière du locuteur cible. En optimisant le système considéré - AutoVC - pour nos besoins, et en l'entraînant avec un accès en boite noire à l'encodeur attaqué, nous avons pu frauder le système de vérification du locuteur à 99.74% pour une limite laissant passer 1% des attaquants. Ces résultats sont présentés en détail dans le chapitre III.

Le troisième problème a été de trouver un alignement pour faire fonctionner ce système sans avoir besoin d'accéder à l'encodeur. En utilisant l'alignement de Wasserstein Procrustes déjà utilisé pour les chiffres, nous avons pu frauder le système de vérification des locuteurs jusqu'à 90.81% des fois avec la même limite que précédemment. Ces résultats sont présentés en détail dans le chapitre IV.

Nous avons donc montré qu'il était possible d'extraire une information sur les utilisateurs à partir d'embeddings de locuteurs de manière non supervisée, et sans accès ni à leur encodeur, ni à son architecture.

VI.1.3 De l'attaque d'un système d'anonymisation de la parole

Forts de notre expérience avec les alignements d'embeddings locuteur, nous avons décidé de mesurer les limites d'un système d'anonymisation de la parole à travers ses embeddings. Un système d'anonymisation de la parole est défini par les caractéristiques de *non-associabilité* et d'*irréversibilité* des données anonymisées. Nous avons choisi d'étudier celui proposé comme baseline pour le Voice Privacy Challenge 2020.

Pour ce faire, nous avons utilisé les hypothèses des attaques proposées lors du challenge : On se place du point de vue d'un attaquant désirant retrouver à quel locuteur appartiennent les extraits de parole d'un jeu de données anonymisées (casser le principe de non-associabilité), et à reconstruire la parole telle qu'elle était avant anonymisation (casser le principe d'irréversibilité). Pour ce faire, on suppose que l'attaquant a un accès :

- Au jeu de données anonymisées dit "*cible*".
- À un jeu de données dit "*d'attaque*", non anonymisé, par les mêmes locuteurs.
- Au système d'anonymisation en boite noire.
- À un encodeur d'embeddings locuteurs entraîné, lui aussi en boite noire.

Nous avons donc anonymisé les données du jeu d'attaque, calculé les embeddings clairs et anonymisés à partir des jeux d'attaque originaux et anonymisés, puis calculé

des fonctions d'alignements de manière supervisée et non supervisée entre les deux jeux d'embeddings.

Pour chaque fonction d'alignement entre l'espace anonymisé et non-anonymisé, nous avons projeté les embeddings associés au jeu cible vers l'espace non-anonymisé, puis les avons comparés avec leur équivalents non-anonymisés (pour savoir si ils avaient bien été reconstruits), et avec les embeddings du même locuteur du jeu d'attaque (pour savoir si il était possible de retrouver le locuteur).

Pour l'associabilité, nous avons mesuré l'EER entre les embeddings du jeu d'attaque et de cible, en suivant les couples fournis pour l'évaluation du Voice Privacy Challenge 2020. Pour les alignements supervisé et non supervisé, nous avons respectivement trouvé un minimum de 13.1% et 13.2% d'EER, sachant que l'encodeur assurait un EER minimum de 2.9% sur des données claires. Nous avons donc pu ré-associer les embeddings anonymisés avec succès, même si nous n'avons pas atteint les mêmes performances qu'avec des données claires.

Pour la réversibilité, nous avons mesuré le pourcentage des embeddings projetés qui étaient plus proche d'un embedding du même locuteur que d'un autre embedding, ce que nous avons appelé *top 1 accuracy*. Pour les alignements supervisé et non supervisé, nous avons respectivement trouvé un maximum de 60.0% et 62.1% de top 1 accuracy. Nous avons donc pu inverser les embeddings anonymisés avec succès, même si nous n'avons pas atteint les 100%. Ces résultats sont présentés en détail dans le chapitre V.

VI.2 Limites

Dans cette section, j'expose les limites constatées sur chacune des tâches effectuées au cours de la thèse.

VI.2.1 Du clustering d'embeddings de chiffres

Notre clustering d'embedding de chiffres a atteint les 98.96% de précision, quand l'étiquetage des clusters a été efficace dans 100% des cas. Pour améliorer notre étiquetage des embeddings, il nous faudra donc améliorer notre clustering. Comme présenté dans la table II.3, même en essayant plusieurs algorithmes de clustering, nous n'avons pas fait mieux. En revanche, on remarque que le réseau qui a produit ces embeddings n'atteint que 97.32% de précision sur l'évaluation du chiffre. Nous sommes donc sûrement limités

ici par la performance du réseau. Après un examen des tracés de chiffres (comme ceux de la figure I.3), on se rend compte que la base de données ne contient pas que des tracés de bonne qualité, ce qui pourrait expliquer qu'on n'atteigne pas les 100% de précision. L'amélioration des résultats passera donc soit par plus de données de chiffres manuscrits, soit des jeux de données plus propres.

VI.2.2 De l'étiquetage d'embeddings de chiffres

Notre technique pour l'étiquetage d'embeddings de chiffre fonctionne très bien (100% de précision sur les clusters de chiffres), mais présente une limite très contraignante : elle ne fonctionne que pour un nombre d'étiquettes *limité* et *connu*. Ici nous avons pu nous en servir car nous savions qu'il n'y avait que 10 chiffres (de 0 à 9) et nous avions suffisamment d'exemplaires de chaque classe (plusieurs milliers d'embeddings pour 10 classes). En revanche, Si l'on souhaite l'appliquer à d'autres données, on se retrouve rapidement limité, par exemple :

- pour savoir quel utilisateur est concerné : car on devrait savoir en avance quels sont les utilisateurs contenus dans le jeu de données.
- pour savoir le mot prononcé dans un extrait de voix : car on devrait gérer un nombre de mots possibles beaucoup trop importants.

Ce sont ces limites qui nous ont poussé à envisager des alignements statistiques plutôt que de poursuivre dans l'analyse des clusters.

VI.2.3 De la reconstruction de données à partir d'embeddings

VI.2.3.1 Pour les embeddings de scripteur

La reconstruction des chiffres à l'aide d'un LSTM-MDN a fait baisser l'accuracy sur la détection du chiffre de 97% à 85%, et monter son EER de 12.72% à 17.71%. On a donc une dégradation des performances, qui peut être due à l'imperfection du décodeur, ou de la taille des embeddings de scripteur, qui n'ont pu contenir qu'une information limitée. Pour aller plus loin , il nous faudrait un décodeur plus précis, ou un système utilisant des embeddings plus grands. Dans le cadre d'une attaque, la dimension des embeddings est fixée. En revanche, il est toujours possible de produire un meilleur décodeur pour générer de l'écriture manuscrite.

VI.2.3.2 Pour les embeddings de locuteur

La reconstruction de la parole par conversion de voix, à l'aide d'AutoVC nous a permis d'atteindre un EER très bas (0.25%) sur un jeu de test venant de la même partition que l'ensemble d'entraînement. Les utilisateurs étaient totalement disjoints, mais tous les extraits étaient enregistrés dans les mêmes conditions. En revanche, en testant son fonctionnement sur un jeu dit "hors domaine", enregistré dans des conditions différentes, on a remarqué une grande baisse des performances : Dans le tableau III.4, on remarque que l'EER monte de 0.25% à 1.10% et que le $sFAR$ descend de 99.74% à 60.07%. Cette grosse différence de performances montre les limites de la reconstruction de parole sur différents domaines.

Par ailleurs, les expériences au cours de cette thèse ont toujours été faites sur des mel-spectrogrammes et pas sur de l'audio, nous manquons également d'un système pouvant reconstruire de la parole naturelle à partir de mel-spectrogrammes. Ces systèmes existent, nous avons fait le choix de ne pas les implémenter dans un premier temps afin de nous focaliser sur les apports en reconstruction du spectrogramme.

VI.2.4 De l'alignement par rotation

VI.2.4.1 Pour un système d'authentification

Les résultats pour l'alignement entre différents encodeurs n'ont pas permis d'assurer une reconstruction parfaite des données, ou en tout cas pas de retrouver les performances du décodeur sur l'encodeur ayant servi à l'entraîner.

Pour le système de vérification du scripteur, si le décodeur permettait de reconstruire des données assez réalistes pour frauder le système dans 97.15% des cas, l'utilisation avec des embeddings d'un encodeur différent et une rotation comme alignement ne nous a permis d'atteindre que 54.64% de fraude. Nous avons voulu mesurer la limite théorique atteignable avec un alignement sous forme de rotation, en utilisant un algorithme d'alignement supervisé, et cette limite a été mesurée à 81.40%.

Pour le système de vérification du locuteur, le décodeur seul a pu frauder le système jusqu'à 99.74% des fois, mais une fois essayé sur un encodeur différent avec un alignement **non supervisé** par rotation, cette performance est descendue à 90.81%. Nous avons donc également mesuré la limite maximale de fraude avec un alignement par rotation en utilisant une rotation produite par un alignement **supervisé**, ce qui nous a permis de monter jusqu'à 96.72% de fraude.

On a donc montré que, même en améliorant notre technique d'alignement, la rotation sera un facteur limitant dans l'alignement des systèmes d'authentification, que ça soit pour la vérification du locuteur ou du scripteur. Nous sommes également limités par nos hypothèses, car nos attaques sur le système de vérification du scripteur nécessitaient la connaissance de l'architecture du réseau utilisé, et nos attaques sur le système de vérification du locuteur nécessitaient la connaissance des données utilisées pour l'entraînement.

VI.2.4.2 Pour un système d'anonymisation de la parole

Pour mesurer les limites dues à l'alignement de notre système d'anonymisation, nous avons calculé les performances optimales, dans le cas d'un alignement supervisé où l'attaquant aurait accès aux données attaquées pour son calcul. La limite théorique mesurée pour un alignement sous forme de rotation a été mesurée à :

- 8.7% d'EER pour l'*associabilité* (contre 2.9% d'EER évalué sur les données claires).
- 99.0% de top 1 accuracy pour la *réversibilité* (contre 100% théorique pour les données claires).

Si on se rapproche fortement des données claires, on remarque tout de même un écart, qui justifiera lui aussi de ne pas se conforter à l'avenir aux alignements par rotation pour ces attaques.

VI.3 Perspectives

Nous avons présenté précédemment les résultats et limites expérimentales liées à cette thèse, nous allons maintenant détailler les perspectives ouvertes par cette thèse et son impact sur plusieurs domaines.

VI.3.1 Perspectives Académiques

Les travaux présentés dans cette thèse ont permis de mettre en évidence les limites des systèmes d'authentification à base de biométries comportementales et des systèmes de pseudo-anonymisation de la parole. Ces limites ont été mises en évidences en se plaçant du point de vue d'un attaquant, sous diverses hypothèses. À la fin de cette thèse, ce sont ces hypothèses qui constituent maintenant la limite académique de nos travaux : pour aller plus loin, il faudra trouver un moyen de s'en affranchir.

Nous avons montré la présence d'informations dans les embeddings de systèmes d'authentification du scripteur. Ces systèmes n'étant pas encore très précis et les bases de données de chiffres manuscrits encore petites et bruitées, la présence de l'information reconstructible est encore très limitée, nous avons donc espoir de voir les performances s'améliorer sur de futurs systèmes, utilisant des technologies d'apprentissage profond plus récentes, ou des données plus propres.

Nous avons également montré qu'il était possible de reconstruire des extraits de parole à partir d'un embedding de locuteur, avec et sans accès à son encodeur. Les performances avec accès à l'encodeur sont très dépendantes du domaine des données utilisées pour entraîner le décodeur. Il reste donc du travail pour examiner le comportement d'un système de conversion de voix sur d'autres domaines, mais également du travail d'analyse pour examiner l'impact d'une synthèse totale de la parole, car nous nous sommes arrêtés à la reconstruction de spectrogrammes ici. Les performances sans accès au décodeur n'ont été acceptables qu'avec davantage de connaissances sur les données utilisées. Nos futurs travaux pourraient soit se concentrer sur la reconstruction sans accès aux données, soit sur l'utilisation d'attaques pour connaître les données utilisées pour entraîner un système (appelées *membership inference attacks*).

Nous avons finalement montré les possibilités d'inversion d'un système de pseudo-anonymisation de la parole à l'aide d'un alignement entre les jeux d'embeddings supervisés et non supervisés.

Dans toutes nos expériences utilisant des alignements supervisés ou non supervisés au cours de cette thèse, nous avons toujours utilisé des alignements sous forme de rotation, une fonction linéaire inversible. Pour aller plus loin et dépasser les limites observées lors de nos expériences "*oracles*", il nous faudra considérer des fonctions non-linéaires inversibles, aussi appelées "*normalizing flows*".

VI.3.2 Perspectives Industrielles et Sociétales

Si les perspectives académiques sont nombreuses, la question de l'impact de ces travaux hors de la communauté académique se pose. Notre utilisation croissante de services numériques via un éventail d'appareils connectés a permis une croissance des moyens d'authentification dans les dernières années, et notamment une généralisation de l'authentification biométrique. En parallèle, l'Europe et la France ont fait le choix de protéger les données personnelles de leurs citoyens, en instituant un protocole général de protection des données. Le concept est simple, même si l'application est très complexe : les *données*

personnelles (toute donnée ou ensemble de données permettant d'identifier un utilisateur) d'un utilisateur ne doivent pas être récoltées si ce n'est pas nécessaire, ne doivent pas être stockées sans l'accord de l'utilisateur, et doivent bénéficier d'une protection adéquate.

Les travaux présentés dans cette thèse se sont focalisés sur l'information discriminante comprise dans les embeddings produits à partir de biométries comportementales, ainsi que sur les techniques utilisées pour reconstruire les données biométriques à l'origine de ces embeddings. En réalisant des attaques de reconstruction de templates sur des embeddings produits à partir de données variées, nous avons démontré que **les embeddings d'un système d'authentification biométrique comportementale contiennent des informations personnelles, discriminantes d'un locuteur**. Ils devraient donc être légalement considérés et protégés comme des données personnelles.

Si le concept d'attaque de reconstruction de templates existait déjà avant cette thèse, nos apports peuvent se résumer en trois points principaux :

- Leur application sur des biométries comportementales comme la parole et les chiffres manuscrits .
- La reconstruction de parole à partir d'embeddings pour la fraude.
- Leur application sans accès à l'encodeur.

Pour atteindre ce dernier point, nous avons utilisé des alignements statistiques inversibles très simples : des rotations. Ces alignements sont conçus pour faire transiter un ensemble d'embeddings d'un espace vectoriel à un autre. Nous les avons déjà utilisés pour appliquer une attaque de reconstruction sur un encodeur non accessible, pour attaquer un système de pseudo-anonymisation, mais nous pourrions également les utiliser pour convertir les embeddings produits par un vieux système directement dans l'espace de sortie d'un système plus récent par exemple.

GLOSSAIRE

accuracy Métrique mesurant la performance d'une classification. Nombre d'éléments correctement classifiés sur le nombre d'éléments testés. Résultat compris en 0 et 1 ou 0% et 100%. 28, 55, 71, 73, 75, 96, 97

associabilité *linkability* - Caractéristique d'un système d'anonymisation ou de pseudo-anonymisation. C'est la possibilité de pouvoir lié une donnée anonymisée à sa version non anonymisée . 114, 118, 120, 123–126

attaque adversarielle Attaque visant à perturber un système d'authentification par une transformation minimale sur la donnée d'entrée. 25

attaques de reconstruction Attaque visant à reconstruire à partir d'embeddings ou de templates des données d'un utilisateur donné, afin de frauder un système d'authentification. 25, 69, 91, 111, 113

authentification Processus permettant à un système informatique de s'assurer de la légitimité de la demande d'accès faite par une entité afin d'autoriser son accès à des ressources du système. Le système d'authentification vérifie si une entité ayant déjà subie un enrôlement est bien celui qu'elle prétend être, au contraire du système d'identification qui trouve celui qu'elle prétends être indépendamment de sa déclaration. 15, 16, 18, 19, 21–23, 25, 28, 29, 34, 48, 143

BiLSTM *Bidirectionnal Long Short Term Memory* - Ce sont des réseaux LSTM qui lisent une séquence dans les deux sens. Pour plus d'informations, allez voir la partie I.4.3.3. 43, 53, 54, 57, 59, 71, 74, 82, 94

biométrie Mesure du vivant. Utilisé pour désigner des techniques de reconnaissance, d'authentification et d'identification. 16–18, 26

EER *Equal Error Rate* - Métrique mesurant la performance de vérification d'un système d'authentification. C'est le taux d'erreur pour lequel un système commet autant de faux rejets que de fausses acceptations. 28, 30, 31, 119, 120, 130

embedding Représentation en grande dimension, discriminante de l'identité d'un utilisateur, produit le plus souvent par un système à base de réseaux de neurones. 18–21, 23, 25, 29, 33, 34, 37, 48, 119, 120

encodeur ou **Extracteur de caractéristiques** Système conçu pour extraire à partir d'une donnée biométrique les caractéristiques discriminantes d'un utilisateur sous forme d'un embedding. 20, 24, 37, 48

enrôlement Processus permettant à un système informatique d'enregistrer le profil d'un utilisateur. Il est utilisé lors du premier contact d'un utilisateur avec le système, et est nécessaire à toute authentification. 18, 19, 21, 29, 56, 63

FAR *False Acceptation Rate* - C'est le taux de fausses acceptations pour un système d'authentification. 5, 29–31

fausse acceptation ou **Faux positif** Lorsqu'un système d'authentification accepte un utilisateur non légitime. 22, 29, 31

faux rejet ou **Faux négatif** Lorsqu'un système d'authentification rejette un utilisateur légitime. 22, 29

fraude attaque d'un système d'authentification visant à se faire passer pour un utilisateur donné en trompant le système.. 5, 15, 22, 23, 25, 31, 32, 67, 74, 83–86, 99

FRR *False Rejection Rate* - C'est le taux de faux rejets pour un système d'authentification. 5, 29–31

irréversibilité *Un-Reversability* - Caractéristique d'un système d'anonymisation ou de pseudo-anonymisation. C'est la possibilité de ne pas pouvoir reconstruire une donnée à partir de sa contrepartie anonymisée. Son inverse est la réversibilité . 114, 118, 123–125, 142

locuteur Sujet parlant, qui produit des énoncés, par opposition à celui qui les écoute et y répond. 17, 27, 28, 36, 39, 40, 49, 50, 52, 55, 64–66, 79–84, 87–89, 91, 93, 107, 108, 112–114, 116, 117, 119, 120, 123, 146

log vraisemblance *log likelihood* - Métrique mesurant la distance entre 2 ensembles de points, représentés comme deux distribution statistiques. 32–34, 59–61, 99

LSTM *Long Short Term Memory* - Ce sont des réseaux de neurones récurrents qui intègrent des vecteurs permettant de garder en mémoire des informations sur le long terme et le court terme. Pour plus d'informations, allez voir la partie I.4.3.2. 42–46, 57, 58, 67, 70–75, 77, 82, 91, 94, 96, 140, 143, 144, 147

MDN *Mixture Density Network* - Ce sont des réseaux de neurones qui utilisent des modèles de mélanges de gaussiennes (ou GMM - *Gaussian Mixture Model*) pour

prédire une densité de probabilité. Pour plus d'informations, allez voir la partie I.4.3.4 . 44–46, 73, 75, 77, 91, 94, 96, 143, 144, 147

perturbation attaque d'un système d'authentification visant à perturber sa décision, provoquer soit un faux refus, soit une fausse acceptation.. 22, 25

réversibilité *Reversability* - Caractéristique d'un système d'anonymisation ou de pseudo-anonymisation. C'est la possibilité de pouvoir reconstruire une donnée à partir de sa contrepartie anonymisée. Son inverse est l'irréversibilité . 118, 125, 141

scripteur Sujet écrivant, qui produit des symboles manuscrits, par opposition à celui qui les lit. Dans cette thèse il s'agit majoritairement de l'utilisateur ayant produit des chiffres manuscrits sur un écran tactile. 6, 26, 44, 46, 49, 50, 52–57, 59, 63, 64, 70, 71, 75, 91, 93, 96, 99, 111, 143, 146

sFAR *spoofing False Acceptation Rate* - C'est le taux de fausses acceptations à un seuil d'acceptation donné, pour un système d'authentification, lorsque celui-ci est face à une attaque de fraude. Par exemple, $sFAR_{1\%}$ est le taux d'attaques de fraude qui sont passées pour un système qui est calibré pour ne laisser passer que 1% des attaquants naïfs. 31, 32, 83, 84, 89, 90, 108–110, 147, 148

template ou **Gabarit biométrique** Représentation statistique d'un utilisateur calculé à partir d'un ensemble d'embeddings. 21, 25, 69

top 1 accuracy Métrique mesurant la performance en ré-identification entre 2 groupes d'éléments, un reconstruit et un brut. Nombre d'éléments reconstruits dont l'équivalent brut le plus proche appartient au même utilisateur, divisé par le nombre d'éléments testés. Résultat compris en 0 et 1 ou 0% et 100%. 119, 123–125

x-vector Désigne l'embedding d'un système de vérification du locuteur à base de réseaux de neurones. La dénomination viens du réseau éponyme [19]. 20, 64, 65, 87, 115, 117, 127

TABLE DES FIGURES

Liste des tableaux

Liste des contributions

Conférences d'audience internationale avec comité de relecture

- Thebaud, T., Le Lan, G., & Larcher, A. (2021, June). Handwritten digits reconstruction from unlabelled embeddings. *In ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 2540-2544). IEEE.*
- Thebaud, T., Le Lan, G., & Larcher, A. (2021, December). Spoofing Speaker Verification With Voice Style Transfer And Reconstruction Loss. *In 2021 IEEE International Workshop on Information Forensics and Security (WIFS) (pp. 1-7). IEEE.*

Workshops d'audience internationale avec comité de relecture

- Thebaud, T., Lan, G. L., & Larcher, A. (2020, October). Unsupervised labelling of stolen handwritten digit embeddings with density matching. *In International Conference on Applied Cryptography and Network Security (pp. 545-563). Springer, Cham.*
- Champion, P., Thebaud, T., Le Lan, G., Larcher, A., & Jouvet, D. (2021, December). On the invertibility of a voice privacy system using embedding alignement. *In ASRU 2021-IEEE Automatic Speech Recognition and Understanding Workshop.*

Conférences d'audience nationale avec comité de relecture

- Thebaud, T., Lan, G. L., & Larcher, A. (2021, April) Étiquetage non supervisé de représentations de chiffres manuscrits volées. *In Actes de la conférence CAID 2020 (p. 128). In CAID 2020-Second Conference on Artificial Intelligence for Defence.*

BIBLIOGRAPHIE

[1] European Parliament and Council, « Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC », *General Data Protection Regulation*, 2016.

[2] B. D. Medlin, J. A. Cazier et D. P. Foulk, « Analyzing the vulnerability of US hospitals to social engineering attacks : how many of your employees would share their password ? », *International Journal of Information Security and Privacy (IJISP)*, t. 2, *3*, p. 71-83, 2008.

[3] B. Hitaj, P. Gasti, G. Ateniese et F. Perez-Cruz, « Passgan : A deep learning approach for password guessing », in *International Conference on Applied Cryptography and Network Security*, Springer, 2019, p. 217-237.

[4] S. Mare, M. Baker et J. Gummeson, « A study of authentication in daily life », in *Twelfth symposium on usable privacy and security (SOUPS 2016)*, 2016, p. 189-206.

[5] A. C. Weaver, « Biometric authentication », *Computer*, t. 39, *2*, p. 96-97, 2006.

[6] A. K. Jain, K. Nandakumar et A. Nagar, « Biometric template security », *EURASIP Journal on advances in signal processing*, t. 2008, p. 1-17, 2008.

[7] J. Wayman, A. Jain, D. Maltoni et D. Maio, « An introduction to biometric authentication systems », in *Biometric Systems*, Springer, 2005, p. 1-20.

[8] W. Yang, S. Wang, J. Hu, G. Zheng et C. Valli, « Security and accuracy of fingerprint-based biometrics : A review », *Symmetry*, t. 11, *2*, p. 141, 2019.

[9] R. Cappelli, D. Maio, A. Lumini et D. Maltoni, « Fingerprint image reconstruction from standard templates », *IEEE transactions on pattern analysis and machine intelligence*, t. 29, *9*, p. 1489-1503, 2007.

[10] W. Jia, W. Xia, B. Zhang, Y. Zhao, L. Fei, W. Kang, D. Huang et G. Guo, « A survey on dorsal hand vein biometrics », *Pattern Recognition*, t. 120, p. 108 122, 2021.

[11] M. Ramalho, P. L. Correia, L. D. Soares et al., « Biometric identification through palm and dorsal hand vein patterns », in *2011 IEEE EUROCON-International Conference on Computer as a Tool*, IEEE, 2011, p. 1-4.

[12] A. M. Badawi, « Hand Vein Biometric Verification Prototype : A Testing Performance and Patterns Similarity. », *IPCV*, t. 14, p. 3-9, 2006.

[13] F. S. Samaria et A. C. Harter, « Parameterisation of a stochastic model for human face identification », in *Proceedings of 1994 IEEE workshop on applications of computer vision*, IEEE, 1994, p. 138-142.

[14] G. Mai, K. Cao, P. C. Yuen et A. K. Jain, « On the reconstruction of face images from deep face templates », *IEEE transactions on pattern analysis and machine intelligence*, t. 41, *5*, p. 1188-1202, 2018.

[15] S. Lawrence, C. L. Giles, A. C. Tsoi et A. D. Back, « Face recognition : A convolutional neural-network approach », *IEEE transactions on neural networks*, t. 8, *1*, p. 98-113, 1997.

[16] M. M. Kasar, D. Bhattacharyya et T. Kim, « Face recognition using neural network : a review », *International Journal of Security and Its Applications*, t. 10, *3*, p. 81-100, 2016.

[17] L. Muda, B. KM et I. Elamvazuthi, « Voice Recognition Algorithms using Mel Frequency Cepstral Coefficient (MFCC) and Dynamic Time Warping (DTW) Techniques », *Journal of Computing*, t. 2, *3*, p. 138-143, 2010.

[18] V. A. Mann, R. Diamond et S. Carey, « Development of voice recognition : Parallels with face recognition », *Journal of experimental child psychology*, t. 27, *1*, p. 153-165, 1979.

[19] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey et S. Khudanpur, « X-vectors : Robust dnn embeddings for speaker recognition », in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2018, p. 5329-5333.

[20] R. M. Hanifa, K. Isa et S. Mohamad, « A review on speaker recognition : Technology and challenges », *Computers & Electrical Engineering*, t. 90, p. 107 005, 2021.

[21] L. Lee et W. E. L. Grimson, « Gait analysis for recognition and classification », in *Proceedings of Fifth IEEE International Conference on Automatic Face Gesture Recognition*, IEEE, 2002, p. 155-162.

[22] R. Tolosana, R. Vera-Rodriguez et J. Fierrez, « BioTouchPass : Handwritten Passwords for Touchscreen Biometrics », *IEEE Transactions on Mobile Computing*, 2019.

[23] G. Le Lan et V. Frey, « Securing smartphone handwritten pin codes with recurrent neural networks », in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2019, p. 2612-2616.

[24] C. Gold, D. v. d. Boom et T. Zesch, « Personalizing Handwriting Recognition Systems with Limited User-Specific Samples », in *International Conference on Document Analysis and Recognition*, Springer, 2021, p. 413-428.

[25] A. K. Jain, K. Nandakumar et A. Ross, « 50 years of biometric research : Accomplishments, challenges, and opportunities », *Pattern recognition letters*, t. 79, p. 80-105, 2016.

[26] S. Huang, N. Papernot, I. Goodfellow, Y. Duan et P. Abbeel, « Adversarial attacks on neural network policies », *arXiv preprint arXiv :1702.02284*, 2017.

[27] M. Todisco, X. Wang, V. Vestman, M. Sahidullah, H. Delgado, A. Nautsch, J. Yamagishi, N. Evans, T. Kinnunen et K. A. Lee, « ASVspoof 2019 : Future Horizons in Spoofed and Fake Audio Detection », in *INTERSPEECH 2019-20th Annual Conference of the International Speech Communication Association*, 2019.

[28] A. Madry, A. Makelov, L. Schmidt, D. Tsipras et A. Vladu, « Towards Deep Learning Models Resistant to Adversarial Attacks », in *International Conference on Learning Representations*, 2018.

[29] A. Kurakin, I. Goodfellow, S. Bengio, Y. Dong, F. Liao, M. Liang, T. Pang, J. Zhu, X. Hu, C. Xie et al., « Adversarial attacks and defences competition », in *The NIPS'17 Competition : Building Intelligent Systems*, Springer, 2018, p. 195-231.

[30] T. Thebaud, G. Le Lan et A. Larcher, « Spoofing Speaker Verification With Voice Style Transfer And Reconstruction Loss », in *2021 IEEE International Workshop on Information Forensics and Security (WIFS)*, IEEE, 2021, p. 1-7.

[31] J. Galbally, A. Ross, M. Gomez-Barrero, J. Fierrez et J. Ortega-Garcia, « Iris image reconstruction from binary templates : An efficient probabilistic approach based on genetic algorithms », *Computer Vision and Image Understanding*, t. 117, *10*, p. 1512-1525, 2013.

[32] J.-w. Jung, H. Tak, H.-j. Shim, H.-S. Heo, B.-J. Lee, S.-W. Chung, H.-G. Kang, H.-J. Yu, N. Evans et T. Kinnunen, « SASV Challenge 2022 : A Spoofing Aware Speaker Verification Challenge Evaluation Plan », *arXiv preprint arXiv :2201.10283*, 2022.

[33] R. Tolosana, R. Vera-Rodriguez, J. Fierrez et J. Ortega-Garcia, « Incorporating touch biometrics to mobile one-time passwords : Exploration of digits », in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018, p. 471-478.

[34] A. Nagrani, J. S. Chung et A. Zisserman, « VoxCeleb : a large-scale speaker identification dataset », *Telephony*, t. 3, p. 33-039, 2017.

[35] J. S. Chung, A. Nagrani et A. Zisserman, « VoxCeleb2 : Deep Speaker Recognition », *Proc. Interspeech 2018*, p. 1086-1090, 2018.

[36] C. Veaux, J. Yamagishi, K. MacDonald et al., « Superseded-cstr vctk corpus : English multi-speaker corpus for cstr voice cloning toolkit », *University of Edinburgh. The Centre for Speech Technology Research (CSTR)*, 2016.

[37] J.-M. Cheng et H.-C. Wang, « A method of estimating the equal error rate for automatic speaker verification », in *2004 International Symposium on Chinese Spoken Language Processing*, IEEE, 2004, p. 285-288.

[38] N. Brümmer et E. De Villiers, « The bosaris toolkit : Theory, algorithms and code for surviving the new dcf », *arXiv preprint arXiv :1304.2865*, 2013.

[39] D. A. v. Leeuwen et N. Brümmer, « An introduction to application-independent evaluation of speaker recognition systems », in *Speaker classification I*, Springer, 2007, p. 330-353.

[40] S. O. Sadjadi, C. Greenberg, E. Singer, L. Mason et D. Reynolds, « The 2021 NIST Speaker Recognition Evaluation », *arXiv preprint arXiv :2204.10242*, 2022.

[41] T. Thebaud, G. Le Lan et A. Larcher, « Handwritten Digits Reconstruction from Unlabelled Embeddings », in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2021, p. 2540-2544.

[42] T. Kinnunen, K. A. Lee, H. Delgado, N. Evans, M. Todisco, M. Sahidullah, J. Yamagishi et D. A. Reynolds, « t-DCF : a detection cost function for the tandem assessment of spoofing countermeasures and automatic speaker verification », *arXiv preprint arXiv :1804.09618*, 2018.

[43] J. Allen, « Short term spectral analysis, synthesis, and modification by discrete Fourier transform », *IEEE Transactions on Acoustics, Speech, and Signal Processing*, t. 25, *3*, p. 235-238, 1977.

[44] V. Tiwari, « MFCC and its applications in speaker recognition », *International journal on emerging technologies*, t. 1, *1*, p. 19-22, 2010.

[45] W.-N. Hsu, B. Bolte, Y.-H. H. Tsai, K. Lakhotia, R. Salakhutdinov et A. Mohamed, « Hubert : Self-supervised speech representation learning by masked prediction of hidden units », *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, t. 29, p. 3451-3460, 2021.

[46] S. Chen, C. Wang, Z. Chen, Y. Wu, S. Liu, Z. Chen, J. Li, N. Kanda, T. Yoshioka, X. Xiao et al., « WavLM : Large-Scale Self-Supervised Pre-Training for Full Stack Speech Processing », *arXiv preprint arXiv :2110.13900*, 2021.

[47] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser et I. Polosukhin, « Attention is all you need », in *Advances in neural information processing systems*, 2017, p. 5998-6008.

[48] N. Dehak, P. J. Kenny, R. Dehak, P. Dumouchel et P. Ouellet, « Front-end factor analysis for speaker verification », *IEEE Transactions on Audio, Speech, and Language Processing*, t. 19, *4*, p. 788-798, 2010.

[49] D. Snyder, P. Ghahremani, D. Povey, D. Garcia-Romero, Y. Carmiel et S. Khudanpur, « Deep neural network-based speaker embeddings for end-to-end speaker verification », in *2016 IEEE Spoken Language Technology Workshop (SLT)*, IEEE, 2016, p. 165-170.

[50] Y. Zhao, T. Zhou, Z. Chen et J. Wu, « Improving deep CNN networks with long temporal context for text-independent speaker verification », in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2020, p. 6834-6838.

[51] J. S. Chung, J. Huh, S. Mun, M. Lee, H.-S. Heo, S. Choe, C. Ham, S. Jung, B.-J. Lee et I. Han, « In Defence of Metric Learning for Speaker Recognition », *Proc. Interspeech 2020*, p. 2977-2981, 2020.

[52] B. Desplanques, J. Thienpondt et K. Demuynck, « ECAPA-TDNN : Emphasized Channel Attention, Propagation and Aggregation in TDNN based speaker verification », in *Interspeech2020*, International Speech Communication Association (ISCA), 2020, p. 3830-3834.

[53] A. Larcher, K. A. Lee et S. Meignier, « An extensible speaker identification sidekit in python », in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2016, p. 5095-5099.

[54] S. Hochreiter, « The vanishing gradient problem during learning recurrent neural nets and problem solutions », *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, t. 6, *02*, p. 107-116, 1998.

[55] K. He, X. Zhang, S. Ren et J. Sun, « Deep residual learning for image recognition », in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, p. 770-778.

[56] A. Krizhevsky, I. Sutskever et G. E. Hinton, « Imagenet classification with deep convolutional neural networks », *Advances in neural information processing systems*, t. 25, 2012.

[57] A. Graves, A.-r. Mohamed et G. Hinton, « Speech recognition with deep recurrent neural networks », in *2013 IEEE international conference on acoustics, speech and signal processing*, Ieee, 2013, p. 6645-6649.

[58] L. R. Medsker et L. Jain, « Recurrent neural networks », *Design and Applications*, t. 5, p. 64-67, 2001.

[59] S. Hochreiter et J. Schmidhuber, « Long short-term memory », *Neural computation*, t. 9, *8*, p. 1735-1780, 1997.

[60] A. Graves et J. Schmidhuber, « Framewise phoneme classification with bidirectional LSTM and other neural network architectures », *Neural networks*, t. 18, *5-6*, p. 602-610, 2005.

[61] R. Tolosana, R. Vera-Rodriguez, J. Fierrez et J. Ortega-Garcia, « Exploring recurrent neural networks for on-line handwritten signature biometrics », *Ieee Access*, t. 6, p. 5128-5138, 2018.

[62] M. Schuster et K. K. Paliwal, « Bidirectional recurrent neural networks », *IEEE transactions on Signal Processing*, t. 45, *11*, p. 2673-2681, 1997.

[63] A. Graves, « Generating sequences with recurrent neural networks », *arXiv preprint arXiv :1308.0850*, 2013.

[64] X.-Y. Zhang, G.-S. Xie, C.-L. Liu et Y. Bengio, « End-to-end online writer identification with recurrent neural network », *IEEE transactions on human-machine systems*, t. 47, *2*, p. 285-292, 2016.

[65] J. F. Brewer, « Graphology », *Complementary Therapies in Nursing and Midwifery*, t. 5, *1*, p. 6-14, 1999.

[66] *Definition de la parole par le CNRTL*, `https://www.cnrtl.fr/definition/parole`, Accessed : 2022-08-16.

[67] N. Tomashenko, B. M. L. Srivastava, X. Wang, E. Vincent, A. Nautsch, J. Yamagishi, N. Evans, J. Patino, J.-F. Bonastre, P.-G. Noé et al., *The VoicePrivacy 2020 challenge evaluation plan*, 2020.

[68] D. P. Kingma et J. Ba, « Adam : A Method for Stochastic Optimization », in *ICLR (Poster)*, 2015.

[69] D. Raj, D. Snyder, D. Povey et S. Khudanpur, « Probing the information encoded in x-vectors », in *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, IEEE, 2019, p. 726-733.

[70] P. Guillot, *La cryptologie : l'art des codes secret : L'art des codes secrets*. EDP sciences, 2013.

[71] J. A. Hartigan, M. A. Wong et al., « A k-means clustering algorithm », *Applied statistics*, t. 28, *1*, p. 100-108, 1979.

[72] T. Thebaud, G. Le Lan et A. Larcher, « Unsupervised labelling of stolen handwritten digit embeddings with density matching », in *International Workshop on Security in Machine Learning and its Applications (SiMLA)*, 2020.

[73] U. Von Luxburg, « A tutorial on spectral clustering », *Statistics and computing*, t. 17, *4*, p. 395-416, 2007.

[74] F. Murtagh et P. Legendre, « Ward's hierarchical agglomerative clustering method : which algorithms implement Ward's criterion ? », *Journal of classification*, t. 31, *3*, p. 274-295, 2014.

[75] M. Ester, H.-P. Kriegel, J. Sander, X. Xu et al., « A density-based algorithm for discovering clusters in large spatial databases with noise. », in *kdd*, t. 96, 1996, p. 226-231.

[76] S. Wold, K. Esbensen et P. Geladi, « Principal component analysis », *Chemometrics and intelligent laboratory systems*, t. 2, *1-3*, p. 37-52, 1987.

[77] J. C. Gower, « Generalized procrustes analysis », *Psychometrika*, t. 40, *1*, p. 33-51, 1975.

[78] J. H. Holland, « Genetic algorithms », *Scientific american*, t. 267, *1*, p. 66-73, 1992.

[79] N. Ketkar, « Stochastic gradient descent », in *Deep learning with Python*, Springer, 2017, p. 113-132.

[80] R. H. Riffenburgh, « Linear discriminant analysis », thèse de doct., Virginia Polytechnic Institute, 1957.

[81] K. A. Lee, A. Larcher, G. Wang, P. Kenny, N. Brümmer, D. Van Leeuwen, H. Aronowitz, M. Kockmann, C. Vaquero, B. Ma et al., « The RedDots data collection for speaker recognition », in *Interspeech 2015*, 2015.

[82] D. P. Kingma et M. Welling, « Auto-encoding variational bayes », *arXiv preprint arXiv :1312.6114*, 2013.

[83] ——, « Stochastic gradient VB and the variational auto-encoder », in *Second International Conference on Learning Representations, ICLR*, t. 19, 2014, p. 121.

[84] D. Ha et D. Eck, « A Neural Representation of Sketch Drawings », in *International Conference on Learning Representations*, 2018.

[85] H. Kameoka, T. Kaneko, K. Tanaka et N. Hojo, « Stargan-vc : Non-parallel many-to-many voice conversion using star generative adversarial networks », in *2018 IEEE Spoken Language Technology Workshop (SLT)*, IEEE, 2018, p. 266-273.

[86] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville et Y. Bengio, « Generative adversarial nets », in *Advances in neural information processing systems*, 2014, p. 2672-2680.

[87] A. Radford, L. Metz et S. Chintala, « Unsupervised representation learning with deep convolutional generative adversarial networks », *arXiv preprint arXiv :1511.06434*, 2015.

[88] T. Karras, S. Laine et T. Aila, « A style-based generator architecture for generative adversarial networks », in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, p. 4401-4410.

[89] D. J. Rezende et S. Mohamed, « Variational Inference with Normalizing Flows », in *ICML*, 2015.

[90] R. Tolosana, P. Delgado-Santos, A. Perez-Uribe, R. Vera-Rodriguez, J. Fierrez et A. Morales, « DeepWriteSYN : On-line handwriting synthesis via deep short-term representations », 2021.

[91] S. H. Mohammadi et A. Kain, « An overview of voice conversion systems », *Speech Communication*, 2017, issn : 0167-6393.

[92] D. G. Childers, K. Wu, D. Hicks et B. Yegnanarayana, « Voice conversion », *Speech Communication*, t. 8, *2*, p. 147-158, 1989.

[93] S. Yuan, P. Cheng, R. Zhang, W. Hao, Z. Gan et L. Carin, « Improving Zero-Shot Voice Style Transfer via Disentangled Representation Learning », in *International Conference on Learning Representations*, 2020.

[94] S. Dhar, N. D. Jana et S. Das, « An Adaptive Learning based Generative Adversarial Network for One-To-One Voice Conversion », *IEEE Transactions on Artificial Intelligence*, 2022.

[95] L. Sun, K. Li, H. Wang, S. Kang et H. Meng, « Phonetic posteriorgrams for many-to-one voice conversion without parallel data training », in *2016 IEEE International Conference on Multimedia and Expo (ICME)*, IEEE, 2016, p. 1-6.

[96] T. Toda, Y. Ohtani et K. Shikano, « One-to-many and many-to-one voice conversion based on eigenvoices », in *2007 IEEE International Conference on Acoustics, Speech and Signal Processing-ICASSP'07*, IEEE, t. 4, 2007, p. IV-1249.

[97] K. Qian, Z. Jin, M. Hasegawa-Johnson et G. J. Mysore, « F0-consistent many-to-many non-parallel voice conversion via conditional autoencoder », in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2020, p. 6284-6288.

[98] Y. Rebryk et S. Beliaev, « ConVoice : Real-Time Zero-Shot Voice Style Transfer with Convolutional Network », *arXiv preprint arXiv :2005.07815*, 2020.

[99] K. Qian, Y. Zhang, S. Chang, X. Yang et M. Hasegawa-Johnson, « Autovc : Zero-shot voice style transfer with only autoencoder loss », in *International Conference on Machine Learning*, PMLR, 2019, p. 5210-5219.

[100] E. Grave, A. Joulin et Q. Berthet, « Unsupervised Alignment of Embeddings with Wasserstein Procrustes », in *The 22nd International Conference on Artificial Intelligence and Statistics*, PMLR, 2019, p. 1880-1890.

[101] S. Vallender, « Calculation of the Wasserstein distance between probability distributions on the line », *Theory of Probability & Its Applications*, t. 18, *4*, p. 784-786, 1974.

[102] D. Yu et L. Deng, *Automatic Speech Recognition.* Springer, 2016.

[103] F. Stahlberg, « Neural machine translation : A review », *Journal of Artificial Intelligence Research*, t. 69, p. 343-418, 2020.

[104] Document ISO/IEC 24745 :2011, « Information technology — Security techniques — Biometric information protection », *ISO/IEC JTCI SC27 Security Techniques*, 2011.

[105] N. Tomashenko, X. Wang, X. Miao, H. Nourtel, P. Champion, M. Todisco, E. Vincent, N. Evans, J. Yamagishi et J. F. Bonastre, « The VoicePrivacy 2022 Challenge Evaluation Plan », *arXiv preprint arXiv :2203.12468*, 2022.

[106] N. Tomashenko, B. M. L. Srivastava, X. Wang, E. Vincent, A. Nautsch, J. Yamagishi, N. Evans, J. Patino, J.-F. Bonastre, P.-G. Noé et al., « Introducing the VoicePrivacy initiative », in *INTERSPEECH 2020*, 2020.

[107] V. Panayotov, G. Chen, D. Povey et S. Khudanpur, « Librispeech : an asr corpus based on public domain audio books », in *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, IEEE, 2015, p. 5206-5210.

[108] H. Zen, V. Dang, R. Clark, Y. Zhang, R. J. Weiss, Y. Jia, Z. Chen et Y. Wu, « LibriTTS : A Corpus Derived from LibriSpeech for Text-to-Speech », *Proc. Interspeech 2019*, p. 1526-1530, 2019.

[109] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlíček, Y. Qian, P. Schwarz, J. Silovský, G. Stemmer et K. Vesel, « The Kaldi speech recognition toolkit », *IEEE Workshop on Automatic Speech Recognition and Understanding*, 2011.

[110] P. Champion, T. Thebaud, G. Le Lan, A. Larcher et D. Jouvet, « On the invertibility of a voice privacy system using embedding alignement », in *ASRU 2021-IEEE Automatic Speech Recognition and Understanding Workshop*, 2021.

[111] Y. Sun, C. Cheng, Y. Zhang, C. Zhang, L. Zheng, Z. Wang et Y. Wei, « Circle loss : A unified perspective of pair similarity optimization », in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, p. 6398-6407.

[112] K. Kobayashi, T. Hayashi, A. Tamamori et T. Toda, « Statistical Voice Conversion with WaveNet-Based Waveform Generation. », in *Interspeech*, 2017, p. 1138-1142.

[113] K. Chen, B. Chen, J. Lai et K. Yu, « High-quality Voice Conversion Using Spectrogram-Based WaveNet Vocoder. », in *Interspeech*, 2018, p. 1993-1997.

[114] F. Kreuk, Y. Adi, M. Cisse et J. Keshet, « Fooling end-to-end speaker verification with adversarial examples », in *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, IEEE, 2018, p. 1962-1966.

[115] H. Akaike, « A new look at the statistical identification model », *IEEE Transactions on Automatic Control*, t. 19, p. 716, 1974.

[116] S. Forrest, « Genetic algorithms : principles of natural selection applied to computation », *Science*, t. 261, *5123*, p. 872-878, 1993.

[117] S. A. Abdulrahman et B. Alhayani, « A comprehensive survey on the biometric systems based on physiological and behavioural characteristics », *Materials Today : Proceedings*, 2021.

[118] C. Zhou, X. Ma et G. N. Di Wang, « Density Matching for Bilingual Word Embedding », in *Proceedings of NAACL-HLT*, 2019, p. 1588-1598.

[119] Y. Hoshen et L. Wolf, « Non-Adversarial Unsupervised Word Translation », in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018, p. 469-478.

[120] G. Glavaš, R. Litschko, S. Ruder et I. Vulić, « How to (Properly) Evaluate Cross-Lingual Word Embeddings : On Strong Baselines, Comparative Analyses, and Some Misconceptions », in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019, p. 710-721.

[121] T. Mikolov, K. Chen, G. Corrado et J. Dean, « Efficient estimation of word representations in vector space », *arXiv preprint arXiv :1301.3781*, 2013.

[122] C. Szegedy, A. Toshev et D. Erhan, « Deep neural networks for object detection », *None*, 2013.

[123] S. Minaee, A. Abdolrashidi, H. Su, M. Bennamoun et D. Zhang, « Biometric recognition using deep learning : A survey », *arXiv preprint arXiv :1912.00271*, 2019.

[124] S. Balakrishnama et A. Ganapathiraju, « Linear discriminant analysis-a brief tutorial », in *Institute for Signal and information Processing*, t. 18, 1998, p. 1-8.

[125] P. Matějka, O. Glembek, F. Castaldo, M. J. Alam, O. Plchot, P. Kenny, L. Burget et J. Černocky, « Full-covariance UBM and heavy-tailed PLDA in i-vector speaker verification », in *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2011, p. 4828-4831.

[126] R. Biswas, M. Alam et H. Sack, « Is Aligning Embedding Spaces a Challenging Task ? A Study on Heterogeneous Embedding Alignment Methods », *arXiv preprint*, 2020.

[127] R. Rapp, « Identifying Word Translation in Non_Parallel Texts », in *ACL*, 1995.

[128] P. Fung, « Compiling bilingual lexicon entries from a non-parallel English-Chinese corpus », in *Third Workshop on Very Large Corpora*, 1995.

[129] M. Cuturi, « Sinkhorn distances : Lightspeed computation of optimal transport », *Advances in neural information processing systems*, t. 26, p. 2292-2300, 2013.

[130] T. Mikolov, Q. V. Le et I. Sutskever, « Exploiting similarities among languages for machine translation », *arXiv preprint arXiv :1309.4168*, 2013.

[131] L. Rüschendorf, « The Wasserstein distance and approximation theorems », *Probability Theory and Related Fields*, 1985.

[132] P. Bojanowski et A. Joulin, « Unsupervised learning by predicting noise », in *International Conference on Machine Learning*, 2017.

[133] I. Jolliffe, « Principal component analysis », *Encyclopedia of statistics in behavioral science*, 2005.

[134] M. Maouche, B. M. L. Srivastava, N. Vauquier, A. Bellet, M. Tommasi et E. Vincent, « A comparative study of speech anonymization metrics », in *INTERSPEECH 2020*, 2020.

[135] B. M. L. Srivastava, M. Maouche, M. Sahidullah, E. Vincent, A. Bellet, M. Tommasi, N. Tomashenko, X. Wang et J. Yamagishi, « Privacy and utility of x-vector based speaker anonymization », *IEEE/ACM Transactions on Audio, Speech and Language Processing*, 2021.

[136] V. Panayotov, G. Chen, D. Povey et S. Khudanpur, « Librispeech : an asr corpus based on public domain audio books », in *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, IEEE, 2015, p. 5206-5210.

[137] D. A. Reynolds, « Gaussian Mixture Models. », *Encyclopedia of biometrics*, t. 741, 2009.

[138] G. Lample, A. Conneau, M. Ranzato, L. Denoyer et H. Jégou, « Word translation without parallel data », in *International Conference on Learning Representations*, 2018.

[139] R. Giot, M. El-Abed et C. Rosenberger, « Fast computation of the performance evaluation of biometric systems : Application to multibiometrics », *Future Generation Computer Systems*, t. 29, *3*, p. 788-799, 2013.

[140] R. Shokri, M. Stronati, C. Song et V. Shmatikov, « Membership inference attacks against machine learning models », in *2017 IEEE Symposium on Security and Privacy (SP)*, IEEE, 2017, p. 3-18.

[141] T. F. Quatieri, T. Talkar et J. S. Palmer, « A framework for biomarkers of COVID-19 based on coordination of speech-production subsystems », *IEEE Open Journal of Engineering in Medicine and Biology*, t. 1, p. 203-206, 2020.

DOCTORAT BRETAGNE LOIRE / MATHSTIC

LIUM
Laboratoire Informatique
Université du Mans

Titre : Attaques par reconstruction de données biométriques comportementales à l'aide d'alignements statistiques d'embeddings

Mot clés : Vérification du Locuteur, Vérification du scripteur, Fraude, Authentification biométriques, Alignements statistiques supervisés et non supervisés, Reconstruction de données

Résumé : La protection des données personnelles est un devoir éthique et légal en Europe. Les systèmes d'authentification biométriques sont naturellement concernés : ils utilisent des vecteurs discriminants les utilisateurs extraits à partir de leurs données biométriques. D'un point de vue entre l'académique et l'industriel, cette thèse explore les vulnérabilités de ces vecteurs - *embeddings* - qui pourraient être utilisées pour voler des données personnelles et frauder un système d'authentification, appliquées à deux biométries comportementales : la parole et les chiffres manuscrits tracés à la main. Les attaques visant à reconstruire ou analyser des embeddings incluent généralement un accès à l'encodeur qui les a produit. Nous avons proposé d'utiliser un alignement statistique non supervisé pour nous permettre, sans accès à l'encodeur, d'analyser un ensemble d'embeddings de chiffres manuscrits volés et de retrouver les chiffres qui leurs étaient associés. En affinant cet alignement et en utilisant un décodeur adapté, nous avons pu reconstruire les tracés de chiffres assez précisément pour frauder le système d'authentification qui les avait produits. La reconstruction de parole est plus complexe, car elle contiens une information linguistique très faiblement conservée dans les embeddings, nous avons donc utilisé un système de conversion de voix pour reconstruire des extraits de paroles assez précis pour frauder un système de vérification du locuteur. Nous avons aussi proposé une attaque sur un système de pseudo anonymisation, en utilisant des alignements supervisés et non supervisés sur les embeddings anonymisés produits par celui-ci, pour explorer ses limites.

Title: Spoofing attacks using behavioural biometric data reconstruction and statistical alignments

Keywords: Speaker Verification, Scriptor Verification, behavioural biometric authentication system spoofing, supervised and non supervised alignments, Reconstruction

Abstract: Privacy protection is an ethical and legal duty in Europe. Biometric authentication systems are naturally implicated, as they use embeddings, extracted from biometric data. From a view between industry and academic, this thesis explore the embeddings vulnerabilities for data thieft and spoofing, applied to two behavioural biometries : Voice and handwritten digits. Template reconstruction attacks usually have an access to the encoder that produce the embeddings. We proposed to use an unsupervised statistical alignment to analyze a set of embeddings without access to their encoder, to find their associated digits. By fine-tuning this alignment and using the adapted decoder, we reconstructed the digit drawings to spoof a digit authentication system. Speech reconstruction is more complex, because it contains linguistic information weakly transmetted through the embeddings, so we had to use a voice conversion system to reconstruct voice extracts well enough to spoof a speaker authentication system. We also proposed an inversion attack on a pseudo-anonymisation system, using supervised and unsupervised alignements on the raw and anonymized embeddings, to explore its limits.