

Rapport de stage

université
PARIS-SACLAY



Détections de contradictions dans un dataset multimodal de mensonges

Rapport de stage présenté et soutenu à Versailles,
le 05 Septembre 2024, par

Malek BEN HAMMED

Tuteur de stage Daniel CAMARA

Enseignante référente Ester MARIUCCI

Etablissement de formation UVSQ - Paris Saclay, 45 Av. des États Unis, 78000 Versailles

Entreprise d'accueil PJGN, 5 Bd de l'Hautil, 95300 Pontoise

Remerciements

Je tiens à remercier toutes les personnes qui m'ont soutenu tout au long de ce stage blablabla.

Malek BEN HAMMED

Table des matières

1 Présentation du Pôle	5
1.1 Pôle Judiciaire de la Gendarmerie Nationale.	5
1.1.1 Institut de Recherche Criminelle de la Gendarmerie Nationale.	5
1.1.2 Service Central de Renseignement Criminel.	5
1.2 Centre Forensique d'Intelligence Artificielle.	6
2 Découverte des projets et familiarisation	7
2.1 Objectifs des projets.	7
2.1.1 Détection de contradiction dans le texte (A).	7
2.1.2 Création du dataset multimodal de mensonge (B).	7
2.2 Ce qui est déjà en place. Pourquoi est-ce là?	8
2.2.1 Pour la détection de contradiction dans le texte.	8
2.2.2 Pour la création du dataset multimodal de mensonge.	8
3 Les débuts dans le projet A	9
3.1 Implémentation de nouvelles fonctions.	9
3.1.1 La fonction sur les similarités à n lettres près.	9
3.2 Tests sur un dataset.	10
3.2.1 Description du dataset.	10
3.2.2 Mise en place des modèles.	10
3.2.3 <i>Undersampling</i>	14
4 Projet pour la création du dataset	17
4.1 Sessions à vide pour comprendre le fonctionnement.	17
4.2 Déroulement de l'entretien.	18
4.3 Sessions de test avec des stagiaires et gendarmes du pôle.	21
5 Implication dans le projet B.	23
5.1 Défaut d'enregistrement audio.	23
5.2 Automatisation de la capture infrarouge.	24
5.2.1 Logiciel <i>TCView</i>	25

5.2.2 Extraction des données de température.	27
5.3 Solution de fusion pour l'audio et les vidéos.	29
5.4 Annotation de la transcription.	31
5.5 Bouton pour l'autofocus.	32
5.6 Score des candidats.	32
5.7 Amélioration de la qualité vidéo.	32
6 Premiers résultats	35

Chapitre 1

Présentation du Pôle

1.1 Pôle Judiciaire de la Gendarmerie Nationale.

Le *Pôle Judiciaire de la Gendarmerie Nationale (PJGN)* est un pôle d'expertise dédié à la criminalistique et à l'intelligence judiciaire, avec une compétence couvrant tout le territoire français. Il a été créé le 1er janvier 2011 à Rosny-sous-Bois avant de déménager à Pontoise en mai 2015. Le *PJGN* est composé d'environ 600 personnels (gendarmes et civils) et dispose d'une capacité de projection sur le terrain pour intervenir lors des incidents les plus graves. Le *PJGN* comprend principalement deux services majeurs : l'*Institut de Recherche Criminelle de la Gendarmerie Nationale (IRCGN)* et le *Service Central de Renseignement Criminel (SCRC)*.

1.1.1 Institut de Recherche Criminelle de la Gendarmerie Nationale.

L'*IRCGN* est une unité de référence en matière de criminalistique et d'expertise scientifique. Il est composé de plusieurs laboratoires spécialisés dans divers domaines tels que la biologie, la chimie, la balistique, la toxicologie, et bien d'autres. Il intervient dans la collecte, l'analyse et l'interprétation des preuves scientifiques, apportant une aide précieuse aux enquêtes criminelles. Grâce à ses équipements de pointe et à son expertise reconnue, il joue un rôle crucial dans la résolution de nombreuses affaires complexes.

1.1.2 Service Central de Renseignement Criminel.

Le *SCRC* est chargé de la collecte, de l'analyse et de la diffusion du renseignement criminel au sein de la *Gendarmerie Nationale*. Il se concentre sur l'analyse

stratégique et opérationnelle des données criminelles pour soutenir les investigations et améliorer la prévention de la criminalité. Le *SCRC* travaille en étroite collaboration avec d'autres services de renseignement et de police, tant au niveau national qu'international, pour partager les informations et coordonner les actions contre la criminalité organisée, le terrorisme, et d'autres menaces graves.

C'est dans le *SCRC* que le *Centre Forensique d'Intelligence Artificielle* se situe (en terme de localisation) (demander à Pierre/Daniel si c'est bien pour l'état major qu'ils travaillent)

1.2 Centre Forensique d'Intelligence Artificielle.

Le *Centre Forensique d'Intelligence Artificielle (CFIA)* du PJGN est une unité innovante dédiée à l'application de l'intelligence artificielle dans les enquêtes criminelles. Ce centre se focalise sur le développement et l'intégration de technologies avancées pour améliorer les capacités d'analyse et d'investigation. Il est composé de deux officiers commissionnés : le LCL *Daniel CAMARA* et le CNE *Pierre FICHEPOIL*; d'un personnel civil : *Camille MAGNOSI*.

Le *CFIA* collabore étroitement avec d'autres services de la Gendarmerie et des entités externes, tant au niveau national qu'international, pour partager des connaissances et des technologies. Il représente une avancée significative dans la manière dont les forces de l'ordre utilisent les technologies de pointe pour résoudre les crimes et améliorer la sécurité publique.

Il a été distingué à plusieurs reprises pour ses contributions innovantes dans le domaine de l'intelligence artificielle appliquée aux enquêtes criminelles notamment en tant que «Plus grand contributeur» à la plateforme d'Europol pour le développement d'outils comme *RosetAI*, un outil ayant une capacité à fournir une transcription de bonne qualité et une traduction multilingue et a aussi obtenu un «Prix d'Excellence Europol du projet innovant» pour la création d'une plateforme d'outils d'intelligence artificielle qui inclut des outils pour l'analyse d'empreintes digitales, la détection d'armes et de drogues, la transcription automatique de la parole en texte, l'extraction d'entités nommées et la traduction automatique dans plus de 100 langues.

Chapitre 2

Découverte des projets et familiarisation

Ce chapitre va présenter l'objectif de ce stage et va contextualiser les recherches sur la détection de contradictions.

2.1 Objectifs des projets.

2.1.1 Détection de contradiction dans le texte (A).

Ce projet vise à développer un outil automatisé de détection de mensonges dans le texte en utilisant des techniques de traitement du langage et d'apprentissage automatique. En nous basant sur des recherches en linguistique menées dans d'autres langues que le français, nous identifierons et implémenterons des caractéristiques linguistiques pertinentes. L'objectif est de créer un modèle capable de classifier automatiquement les textes comme véridiques ou mensongers.

L'implémentation en Python est justifiée par la richesse de ses bibliothèques de traitement du langage et d'apprentissage automatique. Ces outils permettront de prétraiter les données textuelles, d'extraire les caractéristiques linguistiques et de construire des modèles de classification.

2.1.2 Création du dataset multimodal de mensonge (B).

Ce projet vise à créer un dataset complet et diversifié contenant des récits mensongers et véridiques sous trois formats : vidéo, audio, et texte. Ce dataset sera une ressource précieuse pour la recherche en détection de mensonges, permettant d'analyser les caractéristiques distinctives des mensonges et des vérités à travers différents supports. L'objectif est de recueillir et d'annoter des récits provenant

de diverses sources, garantissant ainsi une variété de contextes et de styles de communication.

Le développement de ce projet reposera sur des outils de traitement multimodal pour extraire et synchroniser les informations des différents formats. Pour les audios et les vidéos, des techniques de reconnaissance vocale et d'analyse faciale seront utilisées pour extraire des transcriptions textuelles et des indices visuels/sonores. Le traitement de ces données se fera en Python. Ce dataset permettra aux chercheurs de développer et de tester des modèles de détection de mensonges plus robustes et précis.

2.2 Ce qui est déjà en place. Pourquoi est-ce là ?

2.2.1 Pour la détection de contradiction dans le texte.

Des travaux pour le projet détection de contradiction dans le texte ont été fait par un stagiaire qui a implémenté plusieurs fonctions pour extraire des caractéristiques de données textuelles. Ces caractéristiques concernent par exemple le genre d'un mot, le temps de conjugaison d'un mot... Mais aussi des fonctions qui vont comparer un mot à une liste de mots classifiés par leur humeur : par exemple les mots rire, joyeux et victoire font référence à l'humeur heureuse. On obtient donc un score global d'une phrase/texte en fonction des humeurs. Il y a aussi des caractéristiques concernant la reconnaissance d'entités : noms/prénoms, villes, entreprises, etc. Au total, il y a 88 caractéristiques différentes déjà présentes que l'on peut extraire de données textuelles. Cela a pour but de pouvoir entraîner un modèle sur ces différentes caractéristiques et de voir si oui ou non le texte est mensonger ou non.

2.2.2 Pour la création du dataset multimodal de mensonge.

Concernant le projet pour la création du dataset multimodal de mensonge, la globalité des fonctionnalités sont déjà en place. Cela va de la création de la session du candidat jusqu'à la fin de l'entretien en passant par le remplissage des différents formulaires et des captures audios et vidéos. Tout ceci a été mis en place par *Pierre* et des gendarmes de la *Division des affaires non élucidées (DiANE)*. Tout ceci a une finalité : pouvoir constituer une base de donnée multimodale de mensonge afin de l'étudier.

Chapitre 3

Les débuts dans le projet A

Le projet A a pour but d'analyser des données linguistiques. Pour cela, l'utilisation d'un outil nommé *Trankit* (déjà utilisé partiellement par l'ancien stagiaire) va permettre d'extraire énormément d'information à partir d'une phrase. *Trankit* est un outil open-source de traitement du langage naturel qui permet d'analyser et de comprendre le texte écrit. Il propose diverses fonctionnalités telles que la segmentation des phrases, la tokenisation, l'étiquetage des parties du discours, l'extraction des entités nommées, etc. *Trankit* est conçu pour être multilingue et est basé sur des modèles de deep learning pour fournir des analyses précises et robustes.

3.1 Implémentation de nouvelles fonctions.

Avec l'outil *Trankit*, on obtient une centaine de caractéristiques différentes que l'on peut extraire à partir de données textuelles, en plus de ce qui est déjà présent. Des fonction concernant notamment la présence de ponctuations, la moyenne de lettres par mots ou encore le nombre de phrases dans un texte sont aussi mises en places. Il s'agit de fonctions génériques qui sont simples à implémenter.

3.1.1 La fonction sur les similarités à n lettres près.

Cette fonction a pour objectif de lister les séquences similaires dans un texte. Pour cela, on doit définir à partir de quand une phrase est considérée similaire à une autre. On va utiliser la distance de *Levenshtein*, qui est une distance, au sens mathématique du terme, donnant une mesure de la différence entre deux chaînes de caractères. Grâce à cette mesure, on peut vérifier si deux mots sont proches à n lettres près : par exemple, le mot "simple" et "sample" sont distantes d'une lettre près. On va donc utiliser cette distance afin de définir une condition, au sein

de notre fonction sur les similarités, qui va considérer qu'un mot est similaire à un autre lorsque la distance de *Levenshtein* entre ces deux mots est inférieur ou égale à 2. Avec ce procédé, on peut alors trouver les séquences de mots similaires à d'autres. Par exemple pour la phrase "is this a simple test?" et la phrase "this is a simple text." sont similaires car pour "is" et "this" la distance de *Levenshtein* est de 2, et pour "test" et "text", la distance de *Levenshtein* est de 1.

Cette fonction est importante car une caractéristique présente dans les publications des chercheurs explique que les répétitions mots à mots peuvent être des indices de mensonge.

Avec l'ajout des caractéristiques précédentes et de la nouvelle utilisation de *Trankit*, on arrive à obtenir 260 caractéristiques différentes après un traitement de texte.

3.2 Tests sur un dataset.

Dans cette partie, nous allons appliquer toutes les techniques d'extraction de caractéristiques linguistique du texte à un dataset trouvé sur internet. La structure des données et les modèles utilisé dans toute cette partie seront instanciées avec le paramètre *randomstate* afin de garantir la reproductibilité des résultats et de permettre leur comparaison.

3.2.1 Description du dataset.

Le dataset contient environ 2 000 lignes d'articles où le titre, le site et le contenu de celui-ci y sont détaillés. Une colonne contient un label concernant la véracité de l'article : est-ce que son contenu est vrai ou est-ce une fausse information (fake news) ? Le taux d'article faux au sein du dataset est de 30% et de 70% pour le taux de vraie information, ce qui implique un déséquilibrage du dataset. On applique à ce moment le traitement linguistique mis en place précédemment afin d'obtenir toutes les caractéristiques possible que l'on va stocker dans un dataframe.

3.2.2 Mise en place des modèles.

Avant de passer à la mise en place des modèle, un traitement du dataframe s'impose. On va d'abord retirer les colonnes nulles et à faible variance. Elles représentent 149 colonnes, donc 149 caractéristiques en moins dans le dataframe. Il nous reste 111 caractéristiques que l'on va exploiter.

RandomForestClassifier.

Dans cette partie, on va utiliser le célèbre classifieur *RandomForestClassifier* que l'on va entraîner sur 70% du dataframe final, les 30% restants serviront à tester le modèle. Les paramètres du modèle sont pour l'instant ceux de base.

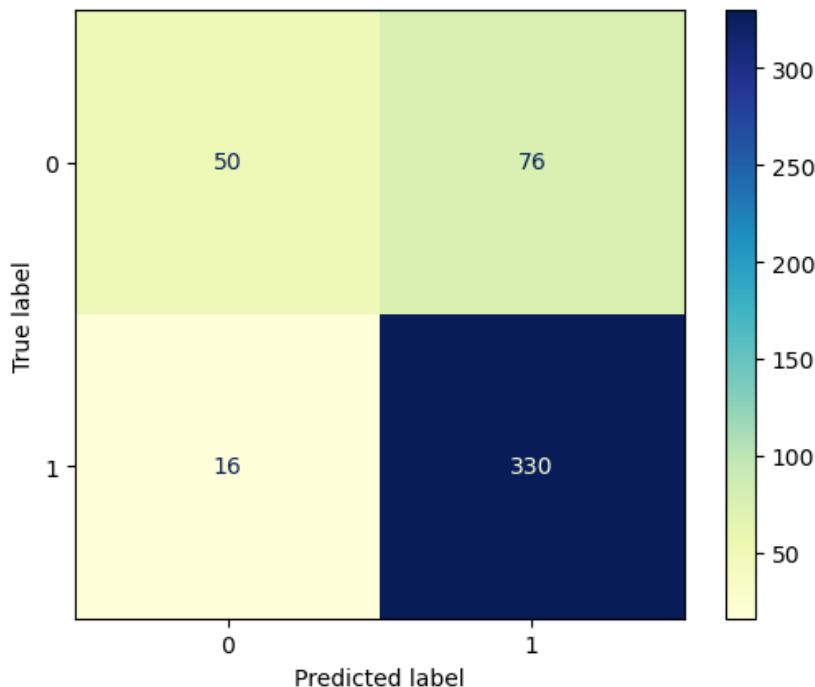


FIGURE 3.1 – Matrice de confusion.

On obtient à partir de la matrice de confusion que la précision des prédition est de 0.8 (80%) et que le *F1-score* avoisine 0.88 (88%). Cependant, on observe que le modèle a du mal à prédire la classe 0 mais arrive bien à prédire la classe 1. Cela est fortement du au fait que le dataset est déséquilibré et il est donc très facile d'avoir une bonne précision même si le modèle se trompe énormément quand il s'agit de prédire la classe 0.

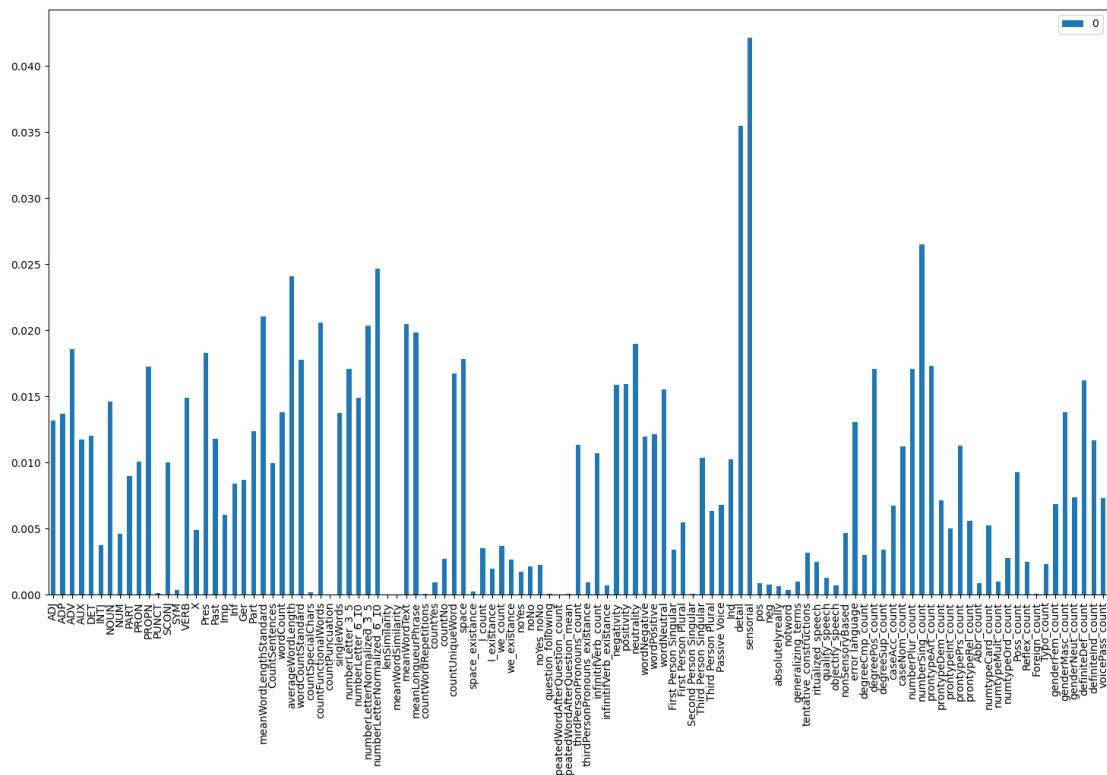


FIGURE 3.2 – Feature importance.

L'image ci-dessus représente l'importance de chaque caractéristique utilisée pour entraîner le modèle. Ces importances sont calculées en fonction de l'amélioration de la qualité des prédictions résultant de la division des arbres basée sur cette caractéristique, cumulée sur tous les arbres de la forêt. On voit que le modèle accorde beaucoup d'importance sur les caractéristiques "detail" et "sensorial". Il faut aussi savoir que la somme de chaque importance des caractéristiques donne 1.

A partir de là, une étape qui arrive naturellement consiste en la suppression des colonnes où le modèle n'accorde pas beaucoup d'importance. Pour cela, une simple boucle sur un seuil (de 0 à 1 avec un pas de 0.01) nous permet de sélectionner les caractéristiques les plus importantes qui, lorsqu'on les sommes, ne dépassent pas ce seuil. On arrive à obtenir un seuil optimal qui est de 0.83 où le modèle entraîné affiche une précision de 0.81 et un *F1-score* de 0.88. Le résultat est très légèrement amélioré.

Une optimisation des résultat par recherche d'hyperparamètres du classifieur en utilisant la méthode *GridSearchCV* de la librairie *scikit-learn* a été faite en jouant sur les paramètres suivants : *n_estimators*, *max_depth*, *min_samples_split*, *min_samples_leaf*, *max_features*, *criterion* et *bootstrap*. Malheureusement, les résultats de la recherche n'ont pas permis d'améliorer le modèle (on perd en précision).

Une élimination récursive des caractéristiques avec validation croisée (*RFECV* en anglais) permet de sélectionner uniquement les caractéristiques optimales du dataframe.

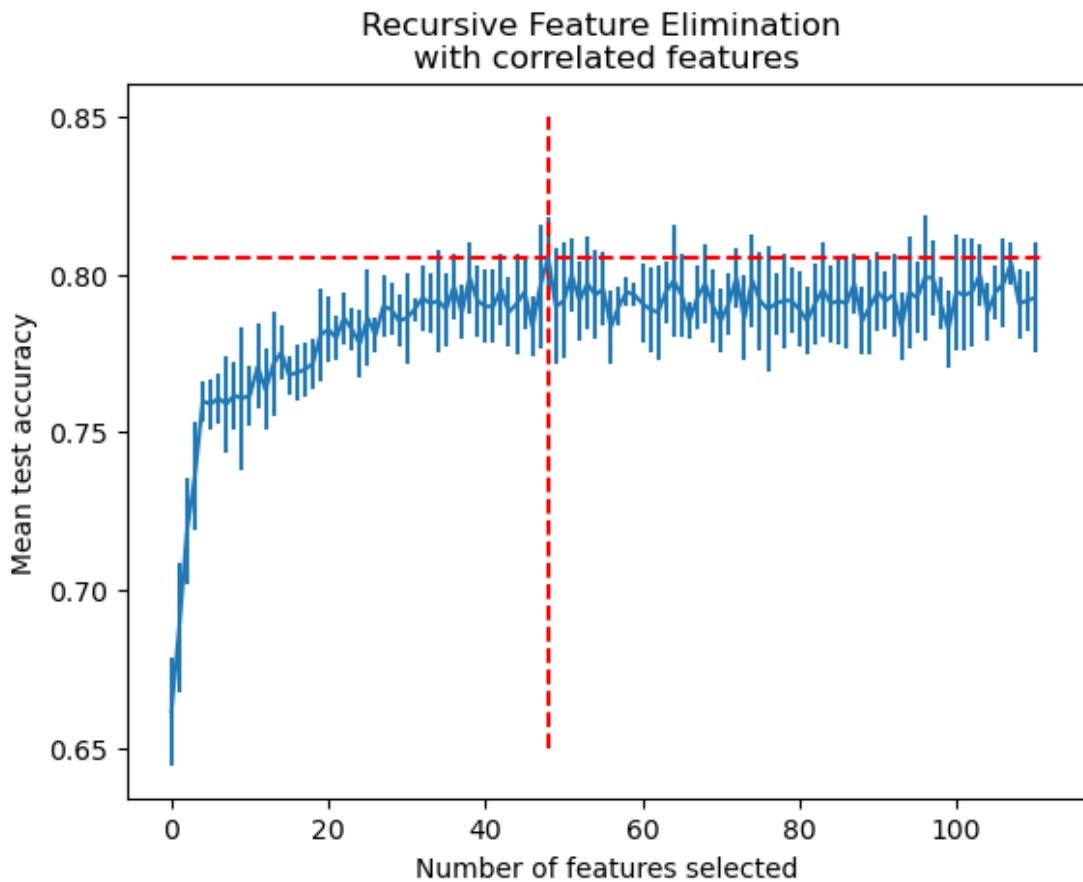


FIGURE 3.3 – *RFECV* appliqué au dataframe.

On peut voir dans la figure ci-dessus que le nombre de caractéristiques idéal est de 49 avec une précision de la prédiction de 0.79 et une précision moyenne du test de 0.80 avec une variation standard de 0.01. Les résultats ne sont toujours pas

meilleurs.

Une donnée importante est la précision équilibrée (*balanced accuracy* en anglais) qui permet d'évaluer les modèles de classification lorsque les classes sont déséquilibrées, afin de garantir que la performance est mesurée équitablement pour toutes les classes. Ici, pour le modèle de base, ce score est de 0.68.

PyTorch Neural Network.

Ici, on va entraîner un réseau de neurones grâce à la librairie *PyTorch* avec les données du dataframe. Notre réseau de neurones va comporter 111 entrées, 3 couches cachées de 64, 32 et 16 neurones, ainsi qu'une sortie.

On va utiliser l'optimiseur *Adam*, avec en paramètre un taux d'apprentissage de 0.001, qui est un composant qui ajuste les paramètres du modèle afin de minimiser la fonction de perte pendant l'entraînement.

D'ailleurs, le taux d'apprentissage va pouvoir être modifié dynamiquement par un *scheduler*, c'est un outil qui permet d'ajuster ce paramètre pendant l'entraînement du modèle.

Et enfin, la fonction de perte sera la *BCELoss* pour *Binary Cross Entropy Loss*, car en effet, on a seulement 2 classes à prédire. Elle va permettre de quantifier l'écart entre les prédictions du modèle et les véritables valeurs cibles.

RESULTATS A RECUPERER.

3.2.3 *Undersampling.*

Comme évoqué précédemment, le dataset est déséquilibré. On va alors utiliser la méthode de l'*undersampling* qui consiste à supprimer des lignes appartenant à la classe la plus présente dans le dataset, en l'occurrence la classe 1, jusqu'à obtenir une répartition équitable. Il existe un autre méthode, l'*oversampler*, qui consiste en la création de doublons appartenant à la classe la moins présente. Ce n'est pas ce que l'on va faire car cela croît le surapprentissage du modèle.

RandomForest.

Après avoir appliqué l'*undersampler* au dataframe, on obtient 908 lignes. On entraîne donc un nouveau classifieur comme précédemment et on obtient alors la matrice de confusion qui suit, avec une précision des prédition de 0.73 et un *F1-score* de 0.74.

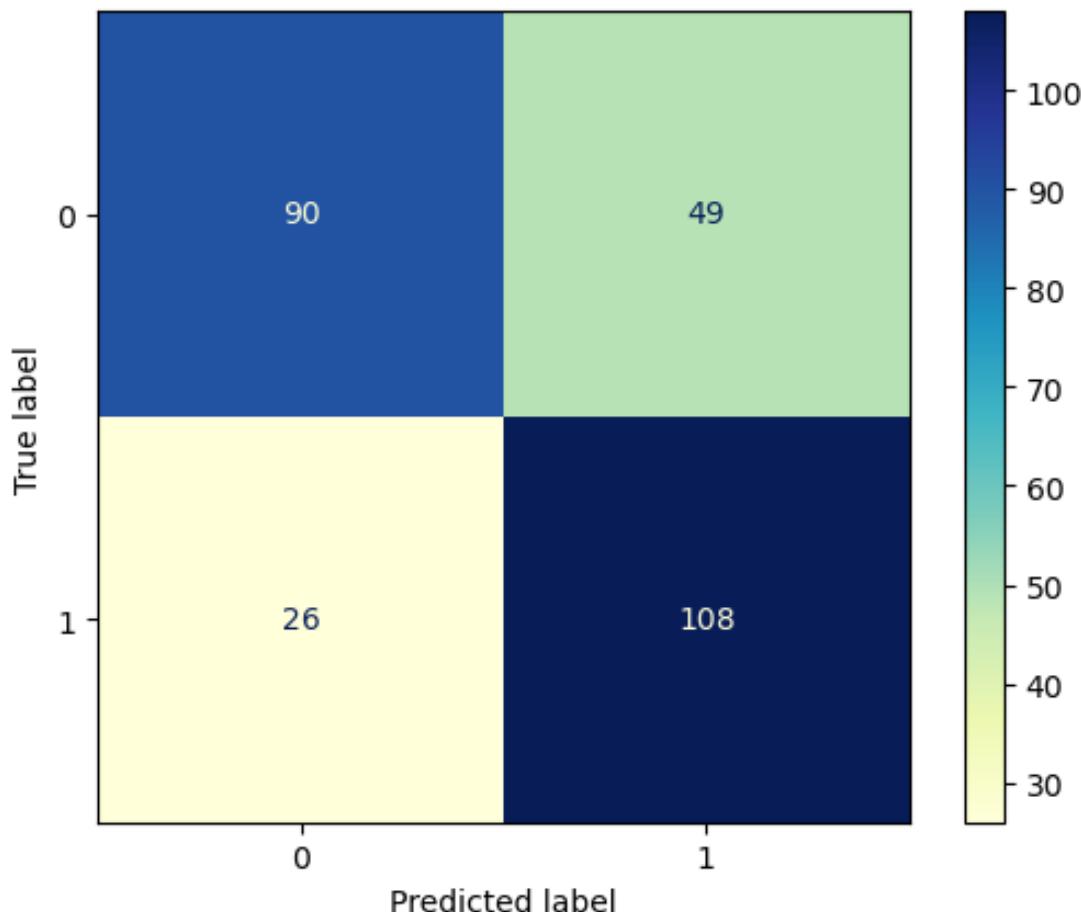


FIGURE 3.4 – Matrice de confusion.

On y voit que la classe 0 est beaucoup mieux prédite que dans les modèles précédents.

Une analyse en composante principale (*ACP*) permet de garder 97% de l'information en passant de 111 caractéristiques à 58 mais on perd 6 points de précision, passant de 0.73 à 0.67.

Les recherche d'hyperparamètres par la méthode *GridSearchCV* et la sélection de caractéristiques importante par *RFECV* ne permettent pas d'améliorer le modèle. On note que *RFECV* donne 0.70 de précision avec 91 caractéristiques sélectionnées sur un total de 111.

En comparaison avec l'entraînement du modèle sur le dataframe entier (sans l'*undersampling*), le score *balanced accuracy* est de 0.73, ce qui est meilleur que le modèle précédent.

PyTorch Neural Network.

On va utiliser la même structure que le modèle précédent. Après un entraînement sur 200 époques, on obtient une précision de prédiction sur les données de validation de 0.65.

AJOUTER LES SCHEMAS!!!!

Chapitre 4

Projet pour la création du dataset

Le projet pour la création du dataset va permettre d'obtenir des données audios, vidéos (infrarouge compris) et texte pour la détection de contradictions.

4.1 Sessions à vide pour comprendre le fonctionnement.

Le projet comprend déjà plusieurs outils établis par *Pierre*. Voici quelques figures qui résument bien ce qui est déjà en place.



FIGURE 4.1 – Page d'accueil.

La figure 5.1 montre la page d'accueil du serveur. Elle a trois fonction : une pour créer une session, une pour restaurer une session et une pour accéder à la "page admin" qui va notamment permettre de lancer les entretiens (captures audios/vidéos). Lors de la création de la session, le candidat va devoir donner son consentement pour toutes les étapes qui vont suivre tout en ayant le droit de se retirer à tout moment sans justification. Il est aussi important de préciser que les données recueillis sont totalement anonymisées et que le candidat peut demander le retrait de sa session du dataset mais aussi que le candidat repartira avec au moins 30€ de carte cadeau. Après le consentement du candidat, il est invité à

répondre à un questionnaire mis en place par des gendarmes de la *DiANE*, puis a vingt minutes pour préparer deux sujets dont voici l'intitulé :

- Décrivez une expérience de voyage/vacances mémorable que vous avez vécue.
- Décrivez une situation embarrassante que vous avez vécu et pour laquelle vous avez ressenti de la honte ou de la peur.

Ces questions sont les mêmes pour tout les candidats et ils ont pour consigne de mentir dans l'une et de dire la vérité dans l'autre.

		Autofocus	Options
Identifiant	Status du dossier	Action	
renard-enfin-alerte	A accepté les termes de l'expérimentation	N/A	
lézard-dûment-rapide	A générée ses questions aléatoires	Lancer enregistrement interview (10s de délai)	
bétaill-tôt-joli	Rémpillage du formulaire de fin d'interview	Ouvrir le formulaire de fin d'interview	
breme-mal-mesuré	Terminé avec un score de ###	N/A	

Questions à poser :

1. Décrivez une expérience de voyage/vacances mémorable que vous avez vécue.
2. Décrivez une situation embarrassante que vous avez vécu et pour laquelle vous avez ressenti de la honte ou de la peur.

FIGURE 4.2 – Page admin.

Ci-dessus, une capture de la page admin où quatre sessions ont été créées avec leurs statut du dossier et leurs actions possible. C'est ici que les entretiens pourront être lancés, que les caméras et microphones pourront être sélectionnés et que l'autofocus pourra être fait. Seul l'enquêteur a accès à cette page (elle n'est accessible qu'après avoir entré un mot de passe). On remarquera qu'un rappel de l'intitulé des sujets posés aux candidats est présent en bas de page sans que l'enquêteur ne sache où le candidat doit dire la vérité ou mentir.

4.2 Déroulement de l'entretien.

Après que le candidat a fini de préparer ses sujets, il se dirige vers l'enquêteur afin de passer à l'étape de l'entretien. L'enquêteur fait l'autofocus, appuis sur le bouton pour commencer l'enregistrement, et là commence un entretien qui va durer une trentaine de minutes, où le candidat va d'abord commencer son récit portant sur un des sujets évoqué précédemment et l'enquêteur pourra ensuite demander des précisions, poser des questions... Au bout d'environ 15 minutes, l'enquêteur et le candidat passent au deuxième sujet. Le temps passé par sujets devrait en moyenne être de 15 minutes mais l'enquêteur est libre de faire durer un peu plus/moins l'entretien par sujet.

Au bout des 30 minutes, le candidat retourne sur sa session et commence à remplir un questionnaire de fin d'entretien. Il en est de même pour l'enquêteur afin d'évaluer le candidat. Ce passage au questionnaire de l'enquêteur est important car c'est à ce moment là qu'il fera une supposition concernant les sujets : le candidat a-t-il dit la vérité ou non (ou pas sûr) ? En fonction du résultat, si le candidat a réussi à tromper l'enquêteur, il gagne 30€ de carte cadeau bonus, sinon il se contentera de repartir avec les 30€ initialement prévu. Cela permet aux candidats de jouer le jeu et de ne pas se contenter de raconter des histoires fausses sans aucun effort.

Voici ci-dessous quelques images de l'installation du matériel/mobilier pendant les entretiens.

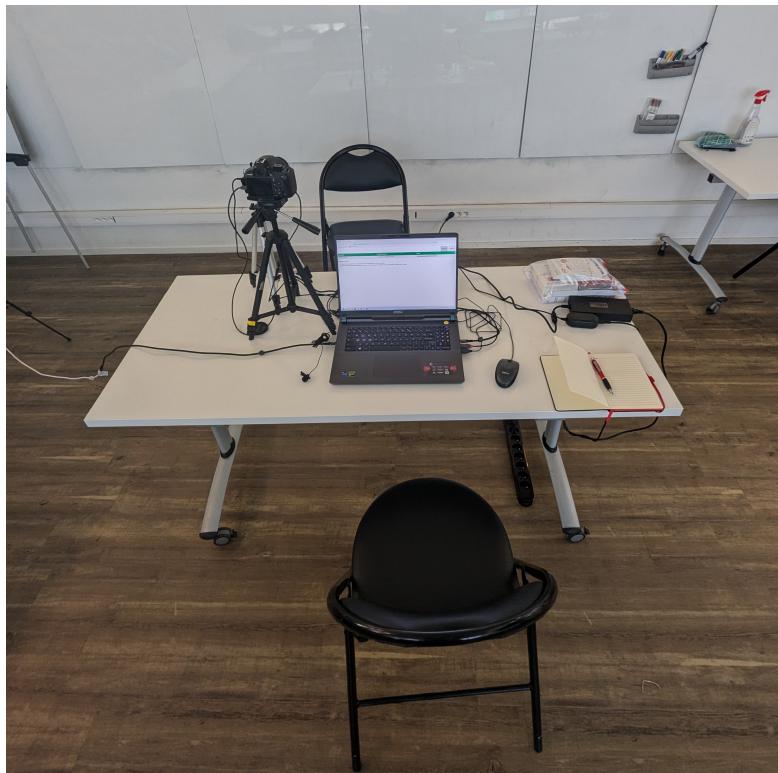


FIGURE 4.3 – Vue enquêteur.

En face de l'enquêteur se trouvera le candidat.

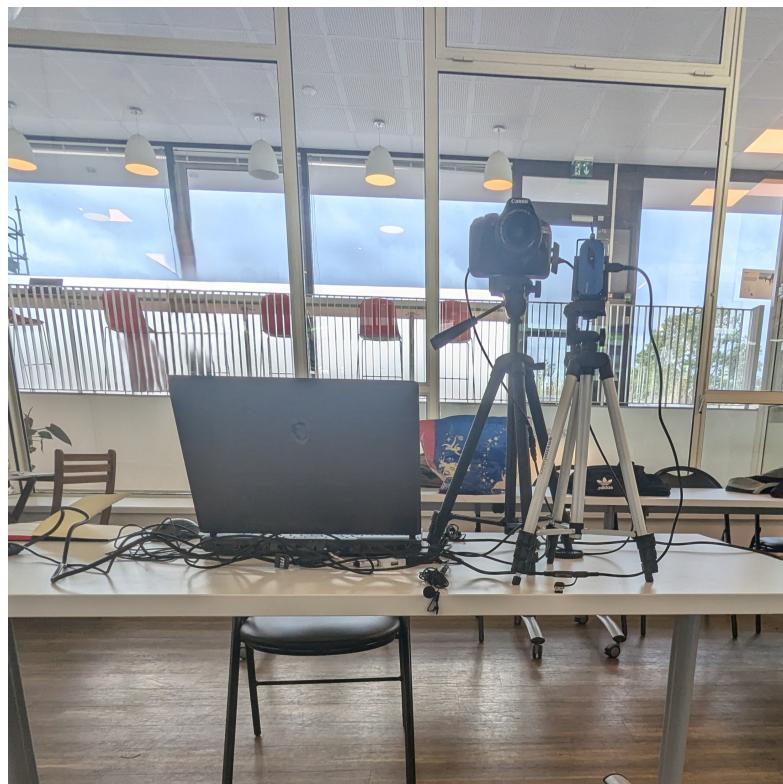


FIGURE 4.4 – Vue candidat.

Devant le candidat se trouvent deux caméra : l'appareil photo et la caméra infrarouge qui cadrent exclusivement la tête. L'appareil photo se concentrera sur les différents mouvement du visage comme la dilatation de la pupille ou les mouvement des lèvres et la caméra infrarouge sur les différences de températures sur le visage et ses extremités (nez, oreilles...).

4.3. SESSIONS DE TEST AVEC DES STAGIAIRES ET GENDARMES DU PÔLE.21



FIGURE 4.5 – Caméra située à droite du candidat.

Une caméra est située à droite du candidat et filme en plan large la globalité du corps afin de percevoir tout les mouvements du candidat pendant l'entretien.

4.3 Sessions de test avec des stagiaires et gendarmes du pôle.

Les sessions de tests avec quelques stagiaires du pôle avaient pour but de mettre en condition quelques enquêteurs et candidats afin qu'ils nous fassent des retours sur leur expérience, mais aussi de mettre en évidence plusieurs bugs. Ces sessions ont permis d'améliorer l'outil mais aussi la qualité des données recueillis pendant les sessions d'entretien. Il y a eu au total 12 sessions de tests dont 3 avec des gendarmes.

A la fin de chaque entretiens, un fichier contenant les réponses du formulaire ainsi que la transcription de phrases annotées vrai/faux par le candidat est sauvegardé en plus des trois fichiers vidéo (dont une infrarouge avec les données de

température) et des fichiers audios (enquêteur et candidat).

METTRE DES IMAGES des caméras

Chapitre 5

Implication dans le projet B.

5.1 Défaut d'enregistrement audio.

Les sessions de test ont permis de découvrir plusieurs défauts dans le projet tant hardware que software, notamment lors de la capture audio. En effet, un défaut majeur était présent dans le processus de d'acquisition audio. Les captures audio effectuées par les microphones branchés à des cartes son USB ne sont ni synchronisées entre elles, ni avec les vidéos.

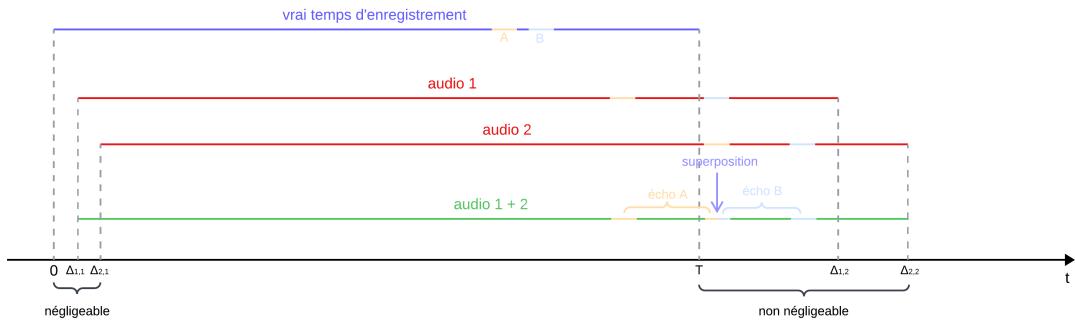


FIGURE 5.1 – Illustration du défaut pour deux audio.

Sur l'image ci-dessus, on peut observer que la capture audio des microphones est beaucoup plus longue que le temps réel de capture. De plus, la lettre "A" (de même pour la lettre "B"), prononcée à un instant précis en temps réel, apparaît à des moments différents dans l'audio 1 et l'audio 2 ce qui implique un écho lors de la fusion de ces deux audios et donc une transcription erronée. Il est important

de préciser que ce problème n'a "lieu" qu'à partir d'un temps d'enregistrement de 1-2 minutes. En dessous de ce temps, l'écho n'est pas perceptible.

La solution la plus efficace a été de créer un fichier qui sauvegarde l'horodatage au début de la capture audio pour chaque microphones, indiqué sur le schéma par $\Delta_{1,1}$ et $\Delta_{2,1}$, puis un autre fichier qui sauvegarde l'horodatage à la fin de la capture audio pour chaque microphones, indiqué sur le schéma par $\Delta_{1,2}$ et $\Delta_{2,2}$. Grâce à ces horodatages, le temps d'enregistrement exacte pour chaque microphone est connu par simple soustraction et une étape de traitement audio, avant l'étape de transcription du fichier audio, a été ajoutée. Cette étape consiste à récupérer la durée de l'audio le plus court, accélérer les autres audios de sorte à ce que leur durée coïncident avec celle de l'audio le plus court et enfin fusionner l'ensemble des audios.

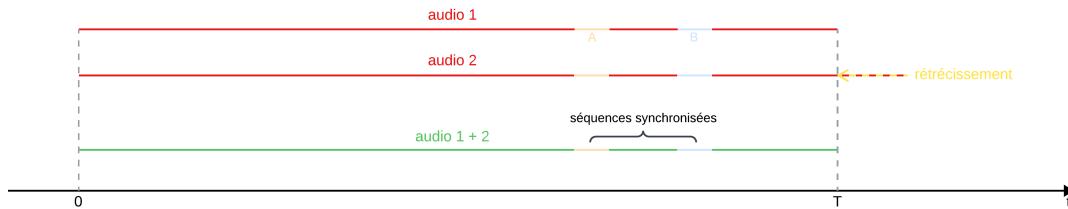


FIGURE 5.2 – Etape de fusion pour deux audios.

Lors de l'étape de rétrécissement des audios, l'horodatage $\Delta_{1,1}$ et $\Delta_{2,1}$ ne sont pas pris en compte car négligeable pour seulement deux capture audios ($\Delta_{1,1}$ et $\Delta_{2,1}$ sont de l'ordre du centième). Cependant, il y a un risque de décalage non négligeable si on souhaite capturer une dizaine/centaine d'audios.

5.2 Automatisation de la capture infrarouge.

Dans cette section, nous allons discuter sur pourquoi il fallait automatiser la capture infrarouge et des solutions trouvés pour réaliser l'automatisation.

Lors des entretiens, seul l'enquêteur était présent dans la pièce, avec le candidat, et donc devait lancer lui même l'enregistrement et le stopper. Or pour la caméra infrarouge, il n'était pas possible de lancer la capture vidéo par le script Python car la caméra infrarouge n'était pas détectée comme webcam par la librairie OpenCV

(ni par Windows d'ailleurs). Avec le risque d'oubli de lancement manuel de la capture infrarouge, une automatisation se devait d'être implémenter.

5.2.1 Logiciel *TCView*.

Le logiciel *TCView* permet de visualiser la sortie vidéo de la caméra infrarouge, de pouvoir capturer des photos/vidéos etc... Ici, c'est la capture vidéo qui nous intéresse. Une façon d'automatiser la capture vidéo serait donc de faire des cliques automatiques sur le bouton pour lancer et couper l'enregistrement. Et c'est ce que nous allons faire.

Pour cela, lors du lancement du serveur, une commande est exécutée afin de vérifier si le logiciel est lancé ou non, afin de ne pas avoir de problème de concurrence pour la capture vidéo. Il est ensuite ouvert ou non en fonction du résultat précédent et des cliques sont réalisées afin de sélectionner la caméra dans la liste des périphériques détectés. Puis la fenêtre se met en arrière plan afin d'améliorer l'expérience utilisateur en étant le plus discret possible. Lors du démarrage d'une interview, une variable globale *IS_RECORDING* change de statut passant de *False* à *True* et à partir de cet instant, un clique automatique va se faire pour lancer l'enregistrement. De même pour la fin d'une interview, *IS_RECORDING* passe de *True* à *False* et un clique automatique est réalisé. Pour la réalisation du clique, la souris est déplacée vers la localisation du bouton pour lancer/couper l'enregistrement et une simulation de clique est exécuté. La localisation des boutons reste toujours la même en fonction de la position du pixel en haut à gauche de la fenêtre du logiciel. On peut donc facilement les récupérer en suivant un motif pour chaque bouton (choix de la caméra et start/stop).

Une sécurité à lieu lors de chaque clique :

- Au lancement de l'interview, une commande va vérifier si un fichier vidéo à bien été créé dans le dossier l'enregistrement, ce qui va confirmer le lancement de l'enregistrement. Si aucun fichier n'est créé, un nouveau clique automatique est réalisé.
- Pour la fin de l'interview, une autre commande va vérifier si le fichier est en cours d'utilisation par le processus de capture *TCView*. Si oui, le clique automatique est réalisé, sinon aucun clique n'est fait car cela relancera un nouvel enregistrement (alors qu'on parle bien de stopper l'enregistrement).

Dans les deux cas, au maximum 5 cliques automatiques sont réalisés afin de ne pas se retrouver dans une boucle incontrôlable de cliques automatiques et des alertes sont créées dans la page *admin* du serveur afin d'avertir l'enquêteur de lancer/stopper la capture infrarouge manuellement en cas d'échecs des cliques. Le

fichier vidéo de la capture infrarouge est ensuite sauvegardé dans la session en cours et en cas d'échec de sauvegarde, une alerte est aussi affichée.

Les cliques automatiques ont un grand risque de ne pas aboutir malgré les 5 essais pour lancer et couper l'enregistrement car il suffit d'un petit mouvement de souris par l'utilisateur pour faire sortir le curseur des extrémités du bouton start/stop (le bouton étant petit). Des recherches de solutions pour le blocage total de la souris pendant les cliques automatiques ou pour la création de souris virtuelle ont été faites mais n'ont aboutis à rien.

En réalité, une autre recherche à permis de découvrir que les drivers de la caméra infrarouge pouvaient être modifiés et donc que la caméra infrarouge pouvait donc directement être contrôlée comme une webcam, mais aussi d'accéder aux données de températures brutes. Mention spéciale à *LeoDJ*, utilisateur *GithHub*, qui a proposé cette solution à un autre utilisateur n'arrivant pas à exécuter un de ses codes pour contrôler une caméra infrarouge et qui n'était pas détectée par *OpenCV*. Un grand merci à lui.



FIGURE 5.3 – Image infrarouge d'un ordinateur portable

L'image ci-dessus est une capture infrarouge d'un ordinateur portable en cours d'utilisation. La partie supérieure montre l'image en noir et blanc assez classique : plus la couleur se rapproche du noir, plus il fait froid et inversement, plus elle est blanche, plus il fait chaud ; la partie inférieure montre une image verte plutôt bizarre qui contient les données de températures et nous allons voir juste après comment lire ces données, encore grâce à *LeoDJ*.

5.2.2 Extraction des données de température.

Dans l'image contenant des dégradés de vert, chaque pixel contient un vecteur de longueur 2 : (x, y) . En position x , on retrouve les niveaux de couleur et en position y , la couche des couleurs auquel le pixel appartient. Le niveau de couleur est un nombre entre 0 et 255 correspondant au niveau de vert dans l'image comme l'image ci-dessous.



FIGURE 5.4 – Barre de couleur verte.

La couche de couleurs permet de distinguer les plages de températures. Par exemple, la couche 0 correspond à des températures entre 0 Kelvin (K) et 20 K et la couche 5, entre 100 K et 120 K (plages non réelles, seulement pour illustrer). Chacune de ces couches contient tout les niveaux de couleurs.

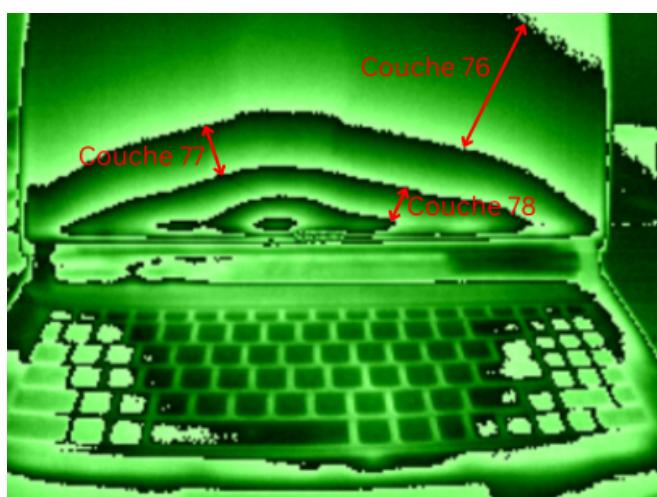


FIGURE 5.5 – Illustration des couches.

Grâce à la rétro-ingénierie de *LeoDJ* qui a étudié plusieurs modèles de caméra infrarouge, dont celle qu'on utilise, nous avons pu comprendre comment récupérer les données de température. Il explique dans un post sur *EEVBlog* que beaucoup de caméra infrarouge utilisent la même façon de coder les données de températures et il donne une formule pour décoder ces données. Pour rappel, chaque pixel contient un vecteur (x, y) correspondant aux données de températures brutes. A partir de ces données, voici la formule que *LeoDJ* a proposé :

$$T = \frac{x + y \times 256}{64} - 273,15$$

où T est la température en Celsius.

Grâce à cela, on peut récupérer les données de températures exactes directement pendant la capture infrarouge. On pourrait alors se dire que cela est inutile car on a déjà la vidéo infrarouge finale, mais cela est plus compliqué que ça. Avec seulement la vidéo infrarouge, on a seulement accès à la carte de couleurs *Jet* et exclusivement des comparaisons entre deux pixels peuvent être faites. En aucun cas, les données de températures peuvent être récupérées seulement à partir de cette image. Une autre solution serait d'obtenir la température maximum et minimum de l'image et d'ensuite faire une échelle par rapport à la carte de couleurs. En réalité, cette solution n'est pas envisageable car la complexité de calcul est monstrueuse. Pour récupérer les données de températures, on utilise une image 256×192 , on calcule la température pour chaque pixel : c'est un $O(1)$; on a donc $256 \times 192 = 49152$ calculs de températures par images. Avec un flux de 25 images par secondes, on obtient environ 1 250 000 calculs par secondes. Une telle demande de calculs n'est pas souhaitable étant donné les tâches parallèles en cours.

On va donc quadrier l'image pour pour récupérer les données de températures dans ces points afin d'avoir tout de même quelques données de température plutôt que rien. Pour cela, un quadrillage de 51×48 pixels est mis en place pour une image de 256×192 . On perd le format original 4 : 3 en obtenant un format 17 : 16 mais on a le quadrillage maximal en données de température conservant 25 images par secondes pendant la capture.

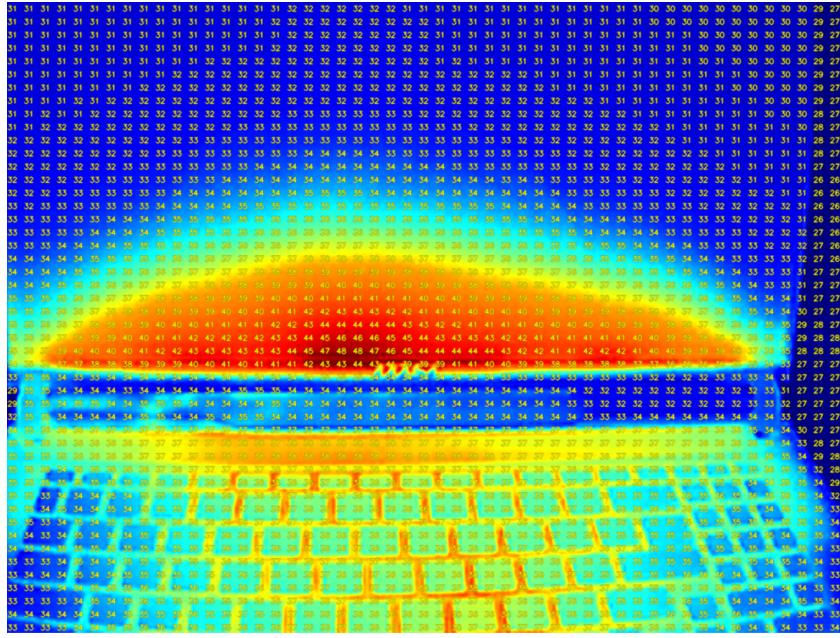


FIGURE 5.6 – Quadrillage de l'image infrarouge avec températures.

On récupère sur chaque image un vecteur de taille 2448 contenant les données de températures qu'on sauvegardera dans un fichier contenant les températures de toutes les images de la vidéo infrarouge.

5.3 Solution de fusion pour l'audio et les vidéos.

Pendant les interviews, le flux audio obtenu par la camera/webcam ne peut être capturé en même temps que le flux vidéo. Les microphones prennent le relais sur cette tâche. Or, comme on l'a vu précédemment, les captures audios ne sont pas synchronisées avec les captures vidéos et pour ne rien faciliter, les captures vidéos ont un délai qui diffèrent les unes des autres : de 0.2 à 0.4 secondes pour la caméra et environ 12 secondes pour la webcam ; d'ailleurs, après la modification d'une option lors de la capture vidéo, le délai de 12 secondes de la webcam est réduit au même ordre de grandeur que la caméra. On a donc un problème si on veut fusionner l'audio avec chaque vidéo.

Pour y remédier, on va créer un horodatage à chaque début de capture audio/vidéo pour obtenir le vrai temps de départ de capture, couper le flux audio/vidéo qui dépasse et fusionner le tout. Il ne faut pas oublier d'accélérer de nouveau l'audio car malgré le rétrécissement précédent, il n'est toujours pas calibré pour la vraie durée d'enregistrement. Pour cela, on créer un horodatage pour

le véritable début d'enregistrement et la véritable fin d'enregistrement.

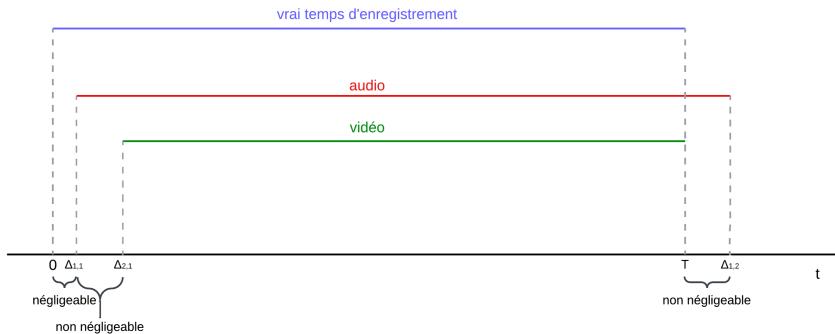


FIGURE 5.7 – Illustration du problème.

Dans le schéma ci-dessus, 0 et T sont les horodatage du vrai temps d'enregistrement. L'audio dépassant de $\Delta_{1,2}$ va être accéléré pour que T coïncide avec $\Delta_{1,2}$. et ensuite, on va couper le début de l'audio pour que lors de la fusion, l'audio et la vidéo soient exactement de la même longueur.

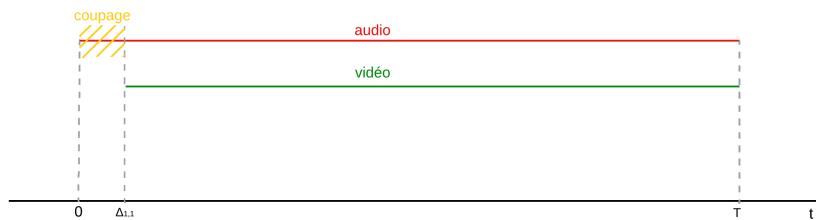


FIGURE 5.8 – Solution apportée.

On peut alors fusionner l'audio et la vidéo.

Cependant, malgré la solution ci-dessus, un décalage très minime mais perceptible apparaît toujours lors de la fusion entre la caméra *Canon* et l'audio. Cela est due au fait que l'horodatage de la capture vidéo de la caméra (concerne aussi la webcam) n'indique pas exactement quand est-ce que la capture vidéo a réellement commencé. L'obtention d'un motif a permis de corriger le problème : il suffisait de diviser par deux la valeur $\Delta_{1,1}$ (pour rappel, $\Delta_{1,1}$ correspond au temps de lancement de la caméra) et de faire la même procédure avec ce nouvel horodatage. Cette procédure a été testée et vérifiée sur plus de 10 sessions avec 20 vidéos de 30 minutes chacune environ. A chaque fois, l'audio et les vidéos étaient synchronisées.

Néanmoins, cette partie ne va pas être intégrée dans le projet car trop dépendante des caméras/webcams utilisées. Elle sera utilisée en post-traitement dans le cadre de la création du jeu de données.

5.4 Annotation de la transcription.

Pendant l'étape de la transcription audio, *pyannote*, un outil de distinction de locuteur était inclus pour que le candidat ait uniquement sa partie audio à consulter afin de ne pas encombrer l'affichage de la transcription. Problème, l'outil *pyannote* fait de mauvaises distinctions entre deux hommes et deux femmes qui parlent et rend donc son utilisation impossible.

Lors des sessions, deux microphones sont utilisés : un pour l'enquêteur et un pour le candidat. Inspiré d'une idée de *Pierre*, une solution apportée consiste en la segmentation des audios en fonction de leur volume en décibels. Grâce à cela, on peut classifier chaque mot de la transcription en fonction du locuteur pour ensuite reconstituer chaque phrase avec son locuteur, étant choisi en fonction de qui sera apparu le plus de fois lors de la classification mots à mots.

Voici ce que l'on obtiendrait :

- pred_id : 1 (pourc 1.00) - real_id : 1 - normalement c'est censé avoir commencé
- pred_id : 1 (pourc 0.67) - real_id : 1 - elle est un peu mal orientée
- pred_id : 1 (pourc 0.60) - real_id : 1 - mais c'est pas grave
- pred_id : 0 (pourc 0.00) - real_id : 1 - 3, 2, 1
- pred_id : 1 (pourc 1.00) - real_id : 1 - bon monsieur
- pred_id : 1 (pourc 1.00) - real_id : 1 - on vous reçoit aujourd'hui pour
- pred_id : 1 (pourc 0.83) - real_id : 1 - que vous nous racontez votre vie
- pred_id : 0 (pourc 0.00) - real_id : 0 - oui
- pred_id : 0 (pourc 0.00) - real_id : 0 - bonjour
- pred_id : 0 (pourc 0.00) - real_id : 0 - ça c'est un micro
- ...

Deux locuteurs parlent : il s'agit en fait de *Pierre* (*locuteur 1*) et moi (*locuteur 0*). Ici, "pred_id" correspond à la prédiction du locuteur faite pour une phrase, "pourc" correspond au pourcentage de mots classé en tant que *locuteur 1* : par exemple pour la deuxième phrase, 67% des mots sont classé *locuteur 1*. Ce pourcentage peut être vu comme une probabilité. On comprend rapidement que "pred_id" résulte directement de "pourc". Lorsque "pourc" est supérieur (resp. inférieur) à 0.5, *locuteur 1* (resp. *locuteur 0*) est associé à "pred_id". "real_id" fait référence

au vrai locuteur que j'ai annoté : on peut voir ici que pour la quatrième phrase, *locuteur 1* est associé à "real_id" tandis que *locuteur 0* est associé à "pred_id". Il y a donc des erreurs lors de la diarisation du locuteur. Grâce à l'annotation de petits échantillons audios, le taux d'erreur est estimé à 7%. Cela est du à la superposition des voix (lorsque les deux locuteurs se coupent la parole, par exemple) ou bien lorsque le microphone frotte contre les vêtements. A cause du taux d'erreur trop élevé et qui risque de ne pas faire gagner de temps au candidat, la diarisation par cette méthode est abandonnée.

5.5 Bouton pour l'autofocus.

A chaque début d'entretien, le focus doit être manuellement réglé par l'enquêteur. Il peut généralement être mieux réglé qu'il ne l'est déjà, notamment grâce à un autofocus. Pour cela, un bouton "Autofocus" est créé dans la page d'accueil admin afin que l'enquêteur puisse actionner ce bouton entraînant la mise au point automatique de la caméra *Canon*. Pour cela, une librairie disponible sur *GitHub* utilisant un logiciel de contrôle de caméra *Canon* et *Nikon* nommé *DigiCam*, qui permet de capturer une image avec autofocus activé via *Python*. L'image créée est ensuite supprimée (voir Figure 4.2).

5.6 Score des candidats.

Le score des candidats correspond aux suppositions correctes faites par l'enquêteur : par exemple, si le candidat a eu la consigne de mentir puis de dire la vérité et que l'enquêteur suppose que le candidat a menti et dit la vérité, le candidat obtient un score de 0/2 (l'enquêteur a fait une bonne supposition). Ce score est affiché dans la "page admin", lorsque la session se termine, et permet de savoir si le candidat a réussi à tromper l'enquêteur ou non (pour rappel, un succès rapporte 30€ supplémentaire). Néanmoins, cette implémentation sera ensuite retirée car une étude est réalisée en parallèle sur les enquêteurs afin de savoir s'ils réussissent à déceler les mensonges dans un récit, donc afficher ce score sur la "page admin" pourrait créer des biais dans cette étude car les enquêteurs ont accès à cette page.

5.7 Amélioration de la qualité vidéo.

A chaque lancement du serveur, une fonction va vérifier si des caméras ne sont pas reconnus dans une base de données contenant leur meilleur résolution/images par secondes. Cette base est obtenue en multipliant la longueur et la largeur des résolutions possibles par caméra, avec plus de 24 images par secondes, et on prend

le plus grand résultat. Le nom de la caméra, la meilleure résolution et le plus haut taux d'image par secondes possible sont sauvegardé dans la base. Grâce à cela, on pourra modifier les paramètres de la caméra un instant avant le lancement d'un entretien afin d'obtenir la meilleure qualité d'enregistrement possible. Exemple de données recueilli pour la webcam *Logitech StreamCam* :

```
"Logitech StreamCam" : {  
    "width" : 1920.0,  
    "height" : 1080.0,  
    "total_pixels" : 2073600.0,  
    "frame_rate" : 30.0  
}.
```


Chapitre 6

Premiers résultats