L3F - Conception d’un environnement technique pour l’adaptation de contenu basé sur les caractéristiques cognitives et émotionnelles des utilisateurs

Encadrant:GNAHO Christophe

BOUAZO Ada

CHABANE-CHAOURAR Ahmed-Lamin

CHAKER Zakaria

JIANG Emile

Référence du document	Rapport_de_recherches_L3F_V1.0
Version du document	1.0
Date du document	20/04/2025
Auteurs:	Ada BOUAZO Ahmed-Lamin CHABANE CHAOURAR Zakaria CHAKER Emile JIANG

Validé par:	Christophe GNAHO
Soumis le:	28/04/2025
Type de diffusion	Document électronique (.pdf)
Confidentialité:	Réservé aux étudiants de l'UFR Maths-Info d'Université Paris Cité

Sommaire

1 - Introduction

Dans le cadre de notre projet de fin de licence, nous avons été chargés de réaliser un projet intitulé **"Conception d'un environnement technique pour l'adaptation de contenu basé sur les caractéristiques cognitives et émotionnelles des utilisateurs"**, identifié sous le code **L3F**.

Pour mener à bien ce projet, des recherches approfondies ont été effectuées afin de déterminer la solution la plus optimale permettant de le livrer dans les meilleures conditions possibles.

Ce projet vise à collecter un ensemble d'informations liées aux émotions des utilisateurs. Sa version finale sera sous la forme d'une application Android, que nous avons nommée **"Emotion Scope"**. Cette application repose sur trois types d'analyses pour recueillir des données émotionnelles : **faciale**, **cognitive** et **vocale**.

Les utilisateurs pourront créer un compte leur permettant de consulter les résultats des différentes analyses ou une analyse globale. De plus, il sera possible d'accéder aux rapports d'analyse à partir de n'importe quel support (plateforme en ligne ou application mobile). Chaque rapport pourra également être téléchargé et enregistré pour un usage ultérieur.

Ce document détaille notamment les technologies mises en oeuvre et leur implémentation, en soulignant comment elles ont été utilisées pour garantir l'efficacité et la pertinence de notre solution.

1.1 - Guide de lecture

Afin de rendre le sujet plus clair, les termes essentiels seront signalés par un astérisque (*****), avec leurs définitions regroupées dans le glossaire. Chaque mot-clé bénéficiera d'explications détaillées, offrant à la fois des perspectives techniques et fonctionnelles. Ces précisions visent à clarifier les concepts fondamentaux et à enrichir la compréhension globale du sujet.

1.2 - Maîtrise d'oeuvre

Durant ce travail de recherche, la conception et le fonctionnement du projet seront assurés par la maîtrise d'oeuvre. Pour rappel, la maîtrise d'oeuvre correspond à une équipe qui veillera au bon suivi de ce document et des autres relatifs au projet en cours. Elle permettra la réalisation du projet et traduira les besoins de la maîtrise

d’ouvrage. La maîtrise d’oeuvre est actuellement constituée de 4 membres en vue de sa réalisation finale. Cette équipe est composée des membres suivants :

- Ada BOUAZO
- Ahmed-Lamin CHABANE CHAOURAR
- Zakaria CHAKER
- Emile JIANG

Afin de parfaire la réalisation de ce document et des autres, mais aussi dans le but de bénéficier d’un soutien maximal, se tient à leur disposition un encadrant, Christophe GNAHO qui intervient également.

1.3 - Maîtrise d’ouvrage

Dans notre projet, la maîtrise d’ouvrage est partagée entre l’encadrant, qui représente le client en définissant les besoins et validant les documents clés, et nous, les étudiants, qui participons également à cette fonction en intégrant les attentes dans la conception. Ensemble, nous veillerons à ce que le produit final réponde aux exigences définies, tout en assumant aussi le rôle de maître d'oeuvre pour sa réalisation.

2 - Concepts de base

Pour mener à bien ce projet, il est essentiel d'adopter une approche rigoureuse combinant recherche approfondie et esprit d'analyse. En effet, nos recherches permettront, dans un premier temps, d'identifier des outils existants capables de mesurer des caractéristiques cognitives et émotionnelles. Elles viseront également à comprendre en profondeur leur fonctionnement et leurs mécanismes sous-jacents.

Adopter un esprit d'analyse sera crucial pour évaluer la pertinence de ces outils en fonction des besoins spécifiques de notre projet. Parmi les technologies identifiées, nous explorerons notamment des outils utilisant des intelligences artificielles*, comme la reconnaissance faciale* des émotions ou l'analyse sonore*. Une fois que les outils les plus adaptés sont sélectionnés, ils seront intégrés par nos soins dans un prototype*.

Ce prototype sera conçu pour collecter des données en temps réel et, à partir de celles-ci, fournir dynamiquement un contenu adapté aux émotions de l'utilisateur à travers un flux de données (base de données*, fichiers, etc). Cet aspect constitue le coeur même de notre projet et s'inscrit dans une optique de personnaliser les interactions numériques pour améliorer l'expérience utilisateur tout en réduisant la charge mentale.

Puisque ce projet s'inscrit dans la continuité des travaux réalisés les années précédentes, il bénéficiera des bases solides déjà établies tout en apportant des améliorations significatives. En combinant nos recherches et nos analyses, nous viserons à développer un environnement technique efficace et innovant, répondant aux besoins des utilisateurs.

3 - Interface et paramètres

3.1 - Environnement de travail

Notre projet est une application Android. Pour développer cette application, nous utiliserons Android Studio. Android Studio permet de gérer la partie back-end en Java et la partie front-end en XML. Une configuration du Gradle* sera nécessaire pour spécifier le niveau Android requis.

3.2 - Paramétrages à effectuer

Notre projet est conçu pour fonctionner sur smartphone. Il sera compatible avec Android 15.0 et un SDK* de niveau 35.0.

Nous avons choisi cette version car il s'agit de la dernière version disponible à ce jour. Elle garantit une compatibilité avec les appareils récents et ceux à venir.

```
android {
    namespace = "com.example.emotionscope"
    compileSdk = 35
    defaultConfig {
        applicationId = "com.example.emotionscope"
        minSdk = 30
        targetSdk = 35
        versionCode = 1
        versionName = "1.0"
        testInstrumentationRunner = "androidx.test.runner.AndroidJUnitRunner"
```

Figure 1 : Configuration de notre projet Android situé dans le fichier gradle.kts

Cependant, comme cette version est encore très récente et que peu d'utilisateurs disposent de cette technologie, nous avons abaissé la version minimale au SDK 30.0 (minSdk = 30), correspondant à Android 11.0.

Android 11 est encore largement utilisé, ce qui permet de garantir une compatibilité avec un large public. De plus, Android 11 est parfaitement adapté aux projets modernes comme le nôtre, qui exploitent des fonctionnalités avancées telles que l'analyse des données utilisateur.

Enfin, ce choix simplifie le développement. En effet, il élimine les problèmes liés aux versions inférieures à Android 11, notamment les anciennes permissions ou les restrictions de stockage.

4 - Les 3 types d'analyses

Les trois types d'analyses ont été développés tout au long de ce projet et sont détaillés dans ce document par trois membres du groupe. Chaque membre a été responsable du développement d'une partie spécifique:

- Zakaria Chaker ⟶ Analyse faciale
- Ahmed-Lamin Chabane Chaourar ⟶ Analyse vocale
- Ada Bouazo ⟶ Analyse cognitive

4.1 - Analyse faciale

4.1.1 - Lancement

Pour lancer le processus d'analyse faciale, il suffit de cliquer sur le bouton Choix d'analyse, puis de sélectionner Analyse faciale ou directement de lancer l'analyse globale. Dans notre projet, cette fonctionnalité est gérée par la classe **`AnalyseFacialeActivity.java`**.

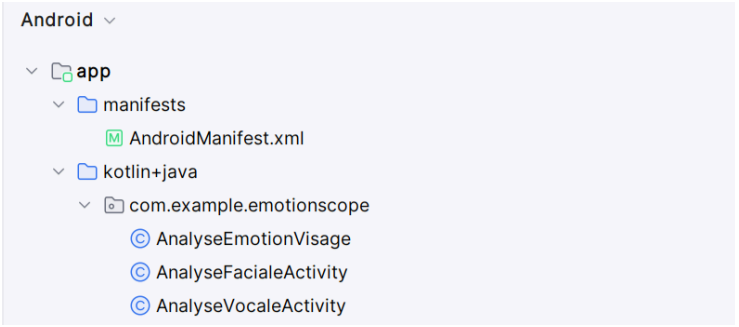

Figure 2 : Structure et arborescence du projet Emotion Scope

Lorsqu'on lance l'analyse, un fichier **`AnalyseFacialeActivity.class`** est automatiquement généré et exécuté pour démarrer le processus.

4.2.2 - Fonctionnement de la caméra

Au lancement de l'activité, la caméra sera activée en continu. Cette méthode permet d'enrichir l'expérience utilisateur en capturant les émotions en temps réel. La caméra frontale est utilisée pour analyser les expressions faciales de l'utilisateur.

Pour activer la caméra, il est nécessaire d'utiliser un objet de type **`ProcessCameraProvider`**, qui gère le cycle de vie et les connexions à la caméra de l'appareil.

Pour intégrer la caméra dans l'interface utilisateur, un composant **`Preview*`** est utilisé, permettant l'affichage en temps réel.

Enfin, grâce à **`CameraSelector`**, il est possible de spécifier le type de caméra à utiliser (frontale ou arrière).

4.2.3 - Détection du visage

Pour détecter un visage, le processus capture une image toutes les millisecondes. L'image capturée est ensuite analysée à travers la méthode **`analyseImageCapturee()`**. C'est au sein de cette méthode que la détection du visage est effectuée.

La détection utilise la bibliothèque **ML Kit*** et, plus précisément, une instance de la classe **`FaceDetector`**. Si un visage est détecté et qu'il répond aux critères définis, il sera traité et capturé.

Afin d'améliorer l'expérience utilisateur, un rectangle sera affiché autour du visage détecté dans l'interface de la caméra (**`Preview`**). Cette fonctionnalité permet à l'utilisateur de visualiser clairement la zone capturée par le programme. Le rectangle flottant affiché à l'écran est géré par la classe **`SuperpositionVisage`**.

Comme illustré dans la figure ci-dessous (mode debug), le programme dessine un rectangle autour du visage de l'utilisateur. Par exemple, un rectangle avec les dimensions suivantes: **`Rect(105, 210, -362, 467)`**.

```
2025-04-25 11:43:56.299 28373-28373 PhoneWindow            com.example.emotionscope  V  DecorView setVisiblity: visibility = 4, Parent = android.view.ViewRootImpl@cc207ce, this = DecorView@9b6f493[MainActivity]
2025-04-25 11:43:56.494 28373-28477 CaptureSession         com.example.emotionscope  D  Attempting to send capture request onConfigured
2025-04-25 11:43:56.495 28373-28477 CaptureSession         com.example.emotionscope  D  Issuing request for session.
2025-04-25 11:43:56.503 28373-28477 CaptureSession         com.example.emotionscope  D  CameraCaptureSession.onConfigured() mState=OPENED
2025-04-25 11:43:56.506 28373-28477 CaptureSession         com.example.emotionscope  D  CameraCaptureSession.onReady() OPENED
2025-04-25 11:43:57.004 28373-28477 StreamStateObserver    com.example.emotionscope  D  Update Preview stream state to STREAMING
2025-04-25 11:43:57.387 28373-28373 DEBUG_RECT             com.example.emotionscope  D  Coordonnées rectangle: Left=105 Top=210 Right=362 Bottom=467
2025-04-25 11:43:57.387 28373-28373 SuperpositionVisage    com.example.emotionscope  D  Rectangle mis à jour: Rect(105, 210 - 362, 467)
2025-04-25 11:43:57.396 28373-28373 skia                   com.example.emotionscope  D  onFlyCompress
2025-04-25 11:43:57.420 28373-28373 skia                   com.example.emotionscope  D  SkJpegCodec::onGetPixels +
2025-04-25 11:43:57.431 28373-28373 skia                   com.example.emotionscope  D  SkJpegCodec::onGetPixels -
2025-04-25 11:43:57.681 28373-28373 EmotionScores          com.example.emotionscope  D  [0.16723332, 0.024732267, 0.16948625, 0.20705555, 0.16412008, 0.16868733, 0.098685235]
2025-04-25 11:43:57.700 28373-28373 SuperpositionVisage    com.example.emotionscope  D  onDraw exécuté
2025-04-25 11:43:57.700 28373-28373 SuperpositionVisage    com.example.emotionscope  D  Dessine rectangle : Rect(105, 210 - 362, 467)
```

Figure 3 : Log d'exécution du programme lors de la détection faciale

Des exceptions sont prévues pour gérer les cas d'erreur, comme la détection de plusieurs visages **ou** l'absence de visage.

Enfin, chaque visage capturé à intervalle régulier (chaque milliseconde) est analysé pour en extraire les informations nécessaires.

4.1.4 - Processus d'analyse des émotions du visage

Lorsqu'une image contenant un visage est capturée, elle est immédiatement transmise à une instance de la classe **`AnalyseEmotionVisage`**, créée par **`AnalyseFacialeActivity`** lors du démarrage de l'activité.

L'instance appelle la méthode **`analyze()`**, en lui transmettant directement l'image capturée par CameraX (sous forme d'un **`ImageProxy`** au format *YUV_420_888**).

Dans un premier temps, la méthode **`analyze()`** vérifie que le format est bien pris en charge, puis convertit l'image en **`Bitmap`** grâce à la méthode **`toBitmap()`**, qui recompose les plans Y/U/V en JPEG* avant décodage.

Le bitmap ainsi obtenu est ensuite redimensionné à 224×224 pixels à travers:

```
resizedBitmap = Bitmap.createScaledBitmap(bitmap, 224, 224, true);
```

Ce qui permet d'adapter l'image à l'entrée attendue par notre modèle Tensor Flow Lite*.

Après redimensionnement, le bitmap est parcouru pixel par pixel pour alimenter un **`ByteBuffer`** direct (taille: 4 × 224 × 224 × 3 octets), chaque composante RGB étant normalisée entre 0 et 1. Ce buffer est alors passé à l'interpréteur TFLite (**`tflite.run(inputBuffer, output)`**), qui renvoie un tableau de scores bruts pour les sept émotions (« Colère », « Dégoût », « Peur », « Joie », « Neutre », « Tristesse », « Surprise »).

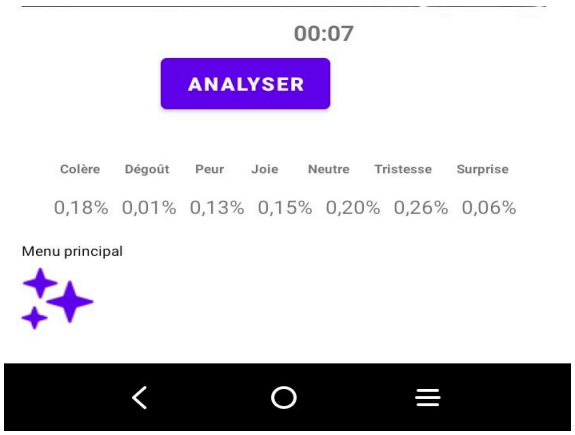

Figure 4: Résultats de l'analyse des émotions du visage par le modèle entraîné.

Enfin, les scores sont convertis en pourcentages par la méthode **`getEmotionPourcentages()`** et transmis au listener* de l'activité à travers:

```
listener.onEmotionDetected(pourcentages);
```

Cet appel, est effectué durant le processus d'interface utilisateur (thread* courant (UI): **`runOnUiThread`**), ce qui va permettre à la méthode **`onEmotionDetected(...)`** de **`AnalyseFacialeActivity`**, de mettre à jour, en temps réel (Cf. runOnUiThread) l'affichage des résultats (affichage dans un tableau), tout en stockant les résultats dans une matrice adaptée à cet instant.

Pour lancer l'analyse faciale, il faudra cliquer sur le bouton "Analyser". Durant le processus d'analyse, une matrice stockera les résultats durant les 7 secondes, (7 secondes correspondant à la période à laquelle l'utilisateur effectuera l'analyse faciale). Pendant ces 7 secondes, la méthode **`lancementAnalyseFaciale()`** capture chaque seconde les niveaux de chaque émotion à l'instant **t**.

Pour cela, une matrice de dimensions 7x7 est utilisée, où :

- L'indice **n** correspond à une seconde.
- L'indice **m** représente le niveau d'une émotion donnée.

$$\begin{pmatrix} 10\% & 15\% & 20\% & 25\% & 10\% & 10\% & 10\% \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 15\% & 10\% & 20\% & 5\% & 30\% & 10\% & 10\% \end{pmatrix}$$

Figure 5: Remplissage de la matrice durant 7 secondes chaque colonne correspond à une émotion.

L'ordre des émotions dans cette matrice est le suivant: Colère, Dégoût, Peur, Joie, Neutre, Tristesse, Surprise. Cet ordre est significatif, car lors de l'entraînement de notre modèle TensorFlow*, les niveaux d'émotions ont été enregistrés selon l'ordre alphabétique des sous-dossiers du dataset. En anglais, cet ordre est: angry, disgust, fear, joy, neutral, sadness, surprise.

Tout ce qui concerne l'entraînement du modèle machine et les données utilisées est détaillé en section **4.1.5**.

4.1.5 - Entraînement model Tensor Flow et conversion

Pour permettre l'analyse faciale, nous avons dû choisir entre utiliser un modèle pré entraîné ou entraîner notre propre modèle. Nous avons opté pour l'entraînement d'un modèle personnalisé.

***Dataset sélectionné: FER_2013**

Pour entraîner un modèle de reconnaissance des émotions faciales, il était essentiel de trouver un dataset contenant un nombre suffisant d'images pour chaque émotion, garantissant ainsi une reconnaissance efficace. Nous avons choisi le dataset **FER_2013**, largement connu dans ce domaine. Cet ensemble de données comprend plus de 30 000 images pour chacune des émotions ciblées.

Cependant, ce dataset présente quelques limitations. Les images fournies sont en niveaux de gris avec une résolution de 48x48 pixels. Nous avons donc redimensionné les images à une résolution de **224x224 pixels**, afin de mieux s'adapter à notre modèle tout en minimisant la perte d'informations. Bien que cette opération entraîne une légère perte de qualité, celle-ci reste négligeable dans notre cas.

Figure 6: Comparaison de deux images issues du Dataset à gauche avant redimensionnement et à droite après.

***Entraînement du modèle avec TensorFlow**

L'entraînement du modèle a été réalisé à l'aide de TensorFlow en Python. Les données sont organisées en sous-dossiers correspondant aux 7 émotions reconnues (joie, tristesse, colère, peur, surprise, dégoût, neutre). L'entraînement dépend du nombre d'époques spécifiées.

Pour rappel, une époque correspond à un passage complet de l'ensemble des données d'entraînement à travers le modèle.

Nous avons utilisé TensorFlow 2.13 et effectué l'entraînement sur CPU*. Bien que l'utilisation d'un GPU* puisse accélérer le processus, cela dépend des capacités matérielles et de la version de TensorFlow utilisée. Après **40 époques**, notre modèle a atteint une précision d'environ **50 %**.

***Conversion du modèle pour Android Studio**

Une fois le modèle entraîné, celui-ci a été sauvegardé au format .h5 (environ 500 Mo). Afin de l'intégrer dans notre projet Android, il a fallu convertir ce modèle au format TensorFlow Lite (.tflite). Cette conversion permet d'obtenir un fichier compatible avec les applications mobiles, qui a ensuite été intégré dans notre projet.

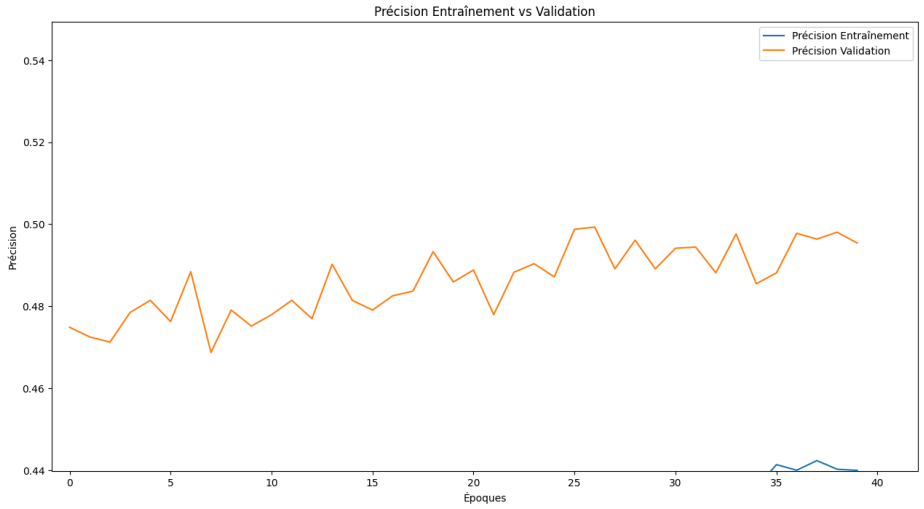

Figure 7: Évolution de la précision en entraînement et validation au fil des époques.

4.2 - Analyse vocale

Pour réaliser cette analyse, l'approche initialement envisagée consistait à entraîner un modèle à l'aide de TensorFlow. Cependant, l'obtention d'un modèle performant aurait nécessité des ressources conséquentes ainsi qu'un grand nombre de fichiers audio représentant différentes émotions (joie, tristesse, colère, neutralité, surprise et peur). Face à ces contraintes, le choix s'est finalement porté sur l'utilisation d'un modèle TensorFlow déjà entraîné, trouvé en ligne.

Afin d'exploiter les informations fournies par ce modèle, une méthode extractMCC issue de la librairie TarsosDSP a été intégrée dans le code Java. Cette méthode permettait d'extraire les caractéristiques audio du fichier .wav enregistré par l'application, pour ensuite les comparer à celles du modèle. Malheureusement, les résultats obtenus se sont révélés décevants, apparaissant complètement aléatoires.

Une nouvelle approche a donc été adoptée : utiliser un code Python associé à un modèle .h5 , qui a été converti en .tflite afin d'assurer la compatibilité avec Android. Cette solution a permis d'obtenir des résultats bien plus cohérents, ou du moins moins aléatoires que précédemment, où seules deux émotions étaient détectées de manière binaire.

Cette transition n'a cependant pas été sans difficulté. Android Studio étant principalement conçu pour Java et Kotlin, l'intégration de Python a nécessité l'utilisation de Chaquopy, un outil permettant d'exécuter du code Python au sein d'une application Android. Il a également fallu installer des versions compatibles des bibliothèques nécessaires, notamment librosa et Tensorflow, ce qui a entraîné de nombreuses erreurs de compilation et occasionné une perte de temps considérable.

Malgré ces obstacles, un résultat fonctionnel a finalement été obtenu, testé avec des fichiers audio pré-enregistrés issus de l'application de base trouvée sur Hugging Face, ainsi qu'avec des enregistrements de voix personnels.

4.3 - Analyse cognitive

Dans un premier temps, qu'est-ce que signifie "**cognitive**" ?

Voici une définition :

L'ensemble des processus mentaux qui se rapportent à la fonction de connaissance et mettent en jeu la mémoire, le langage, le raisonnement, l'apprentissage, l'intelligence, la résolution de problèmes, la prise de décision, la perception ou l'attention.

La première version de ce programme était d'abord une dizaine de questions, et l'utilisateur devait choisir un chiffre de 1 à 7. Pourquoi jusqu'à 7 ? Le chiffre pouvait aussi s'étendre jusqu'à 10. Le chiffre 7 et 10 sont dû aux études faites pour savoir si quelqu'un a l'Alzheimer.

4.3.1 - Pour le 7, ce sont les 7 stages de l'Alzheimer: