

CONSERVATOIRE NATIONAL DES ARTS ET METIERS
CENTRE REGIONAL DE TOULOUSE



MEMOIRE

En vue d'obtenir
Diplôme d'ingénieur AISL
(Architecture et Intégration des Systèmes Logiciels)
Spécialité : INFORMATIQUE - Parcours Ingénierie de projets

Présenté par
BLAVETTE Davy

DataTrainX
**Outil de recherche des singularités cognitives d'un apprenant, basé
sur le deep-learning en vue d'améliorer la neuropédagogie**

Soutenu le 07/07/2022

JURY

PRESIDENTE : Mme. Nicole LEVY
MEMBRES : M. Thierry MILLAN
M. Marc CANNAC
.....

Professeur CNAM Paris
Tuteur CNAM, Maître de conférences UPS
Tuteur AMIO, Formateur informatique
.....

REMERCIEMENTS

Il est par convention de dédier une page de remerciements lorsque nous réalisons un mémoire. C'est un exercice que je veux au-delà de toute considération académique, car je reconnais que la façon de remercier dépend de ce que l'on reçoit et les idées reçues n'exigent pas de remerciements.

J'ai pris un réel plaisir à réaliser ce mémoire, car c'est un sujet qui me tenait à cœur et m'a permis de développer des compétences sur des domaines informatiques variés mais aussi dans la pédagogie ou les neurosciences. C'est un moyen aussi d'expression où j'ai concrétisé et formalisé des compétences, acquises, sur plus de 20 années d'expérience en informatique. Alors certes, réaliser un mémoire quand on a 44 ans peut paraître un tantinet tardif comparativement à des jeunes qui réalisent cette étude lorsque l'on a 20 ans. Néanmoins à cet âge j'avais un CAP bûcheron en poche et n'étais pas spécialement prédestiné à de telles formalités. Le chemin fût donc quelque peu atypique, mais selon Albert Camus « En vérité, le chemin importe peu, la volonté d'arriver suffit à tout. ». Je compléterai aussi par un peu de chance, car sur ce chemin, j'ai trouvé des personnes impliquées, encourageantes et professionnelles que je remercie chaleureusement.

Tout d'abord le CNAM et les institutions qui proposent des formations continues, diplômantes tout au long du parcours de vie et qui a rendu possible cette réalisation.

L'Open Source, Internet et tous ces anonymes qui partagent leurs connaissances, à qui je dois presque tout dans ma formation et ma carrière d'autodidacte et qui encore aujourd'hui m'a permis de réaliser DataTrainX et ce mémoire.

Mes collègues, Marion Jaeger-Amieux, Marc Cannac et Marie-France De Matos, qui m'ont soutenu dans leurs approches bienveillantes et leurs conseils avisés.

Enfin M. Thierry Millan, qui a tenu jusqu'au bout dans la lecture et la révision de la centaine de pages que compte ce mémoire. J'ai apprécié sa disponibilité et son professionnalisme sans langue de bois.

ABREVIATIONS & GLOSSAIRES

AU	Unité d'action
CCC	Concordance Correlation Coefficient
CNN	Convolutionnal Neural Network
ECNN	Emotions CNN
FACS	Facial Action Coding System
FE	Expression faciale
GPU	Graphical Physical Unit
IHM	Interface homme machine
LBP	Local Binary Pattern
LSTM	Long-Short Term Memory
MAE	Erreur Moyenne Absolue (Mean Absolute Error)
MIL	Multiple Instance Learning
MTCNN	Multi-Task Convolutional Neural Network
PCC	Pearson Correlation Coefficient
PEAM	Pourcentage d'erreur absolue Moyenne
RAD	Développement rapide d'application
REF	Reconnaissance d'Expressions Faciales
RLU	Rectified Linear Unit
RNN	Recurrent Neural Network
SIFT	Scale Invariant Feature Transform
SVM	Support Vector Machine

TABLE DES MATIERES

Remerciements	2
Abréviations & glossaires.....	3
Table des matières	4
Introduction	6
Chapitre 1 - Analyse et étude de faisabilité	10
1.1 - Etude préliminaire et mode opératoire.....	10
1.2 - L'orientation algorithmique	16
1.3 - Neuropédagogie et typologie d'apprentissage.....	20
1.4 - La localisation du visage	26
1.5 - La reconnaissance d'expressions faciales (REF)	32
1.6 - Deep-learning, Bibliothèques et Dataset.....	36
Chapitre 2 - La conception DataTrainX	41
2.1 - Mise en œuvre du projet	41
2.2 - Architecture informatique	48
2.3 - Entraînement du modèle CNN	55
2.4 - Maquettage de l'application	64
2.5 - Système de gestion des données	67
Chapitre 3 - Développement du projet.....	72
3.1 - Solution Node.JS.....	72
3.2 - Développement Frontend.....	81
3.3 - Développement Backend.....	89
3.4 - Data Mining et classification des données	92
3.5 - Sécurité et charge serveur	98

3.6 - Environnement de développement	102
Conclusion & perspectives.....	106
Bibliographie	108
Liste des figures, tableaux et codes	117
Annexes.....	122
1 – Etude de Coffield.....	122
2 – Modèle CNN Hyperparameter pour Fer2013	123
3 – Grille de scores Chapman, Kolb	124
Resumé	125
Abstract.....	126

INTRODUCTION

Actuellement en poste, en tant que formateur technique en conception et développement d'application dans un centre de réadaptation professionnelle pour un public avec des déficiences psychiques, physiques ou sensorielles, notre établissement a des objectifs d'inclusion portés par un programme européen.

Ces établissements sont composés d'une équipe pluridisciplinaire (médecin, psychologue, formateur, assistante sociale...) permettant de proposer des parcours adaptés en formation qualifiante dont l'objectif est de promouvoir l'inclusion.

Dans cette optique, l'équipe pédagogique doit régulièrement réadapter ses cours au public accueilli, cela s'oriente parfois sur de la pédagogie différenciée et les moyens pédagogiques peuvent être nombreux tels que « LMS (Learning management system), Mobile learning, Mooc, Spoc (cours en ligne privés en petit groupe), Serious Game... », or l'intuition ou les adaptations des pédagogies, ne permettent pas pour autant d'avoir une efficacité suffisante sur l'assimilation des savoirs. D'un point de vue pédagogique, ces objectifs d'inclusion en formation sont à reporter sur l'apprenant afin que ce dernier puisse avoir un service alternatif en vue de favoriser sa réussite.

De ce constat, inclusion et apprenant sont au cœur d'une problématique commune et l'on m'a missionné pour le développement d'une application qui a pour objet de préciser, analyser et comprendre les stratégies d'apprentissage de chaque individu pour adapter des modes de formation « sur mesure » à même d'optimiser l'acquisition de compétences.

L'outil doit permettre à terme, une pédagogie non plus posée sur la masse indécible d'une classe dont on assure la gestion, mais sur la capacité du formateur à appréhender la singularité de chacun. Des parcours pourront être proposés « à la carte » passant d'une logique de places à une logique de solutions.

En pédagogie, l'approche inclusive se définit par une diversification de ses méthodes d'apprentissage telles que revisiter son plan de cours ou adapter son enseignement, varier ses

stratégies d'évaluation tout en respectant les intentions pédagogiques du cours et sans réduire le niveau d'exigence.

Dans l'apprentissage, une logique de la restitution qui prévaut encore sur une logique de la compréhension serait à l'origine de nombreux échecs de l'apprenant. Pour se comprendre, comprendre le monde et autrui, tout apprenant produit et met en œuvre des ressources métacognitives. Aussi, tout formateur doit élaborer et mettre en pratique des stratégies d'intervention qui permettent à l'apprenant de se construire et de construire sa propre compréhension du monde.

Toute situation d'apprentissage se construit à partir d'un environnement spécifique et d'un individu. Dans l'acte d'apprendre, chaque individu adopte un comportement qui peut évoluer, s'adapter et dont le formateur doit tenir compte. Dans ces situations, analyser la manière dont apprend un stagiaire peut aider à diagnostiquer les difficultés d'apprentissage [1].

On observera que cela fait appel à de nombreuses disciplines, comme la psychologie, la pédagogie, les neurosciences, mais aussi le management, qui demandera à l'équipe enseignante une formation continue pour s'appuyer sur de tels dispositifs, légers et adaptables.

Aussi, pour le développement de cette application, je m'appuierai sur les neurosciences et l'imagerie cérébrale, qui démontrent que l'émotion chez l'être humain est un partenaire fondamental de la cognition, de sa créativité et de sa prise de décision [2].

L'intelligence artificielle et plus précisément, les algorithmes d'apprentissages automatiques permettent de nos jours d'obtenir des résultats significatifs dans la reconnaissance de formes complexes.

Nous observons une montée significative des performances de ces algorithmes, avec la capacité de la machine d'améliorer ses performances sans que les humains s'expliquent exactement comment elle y arrive. L'évolution rapide dans ce domaine, ces dernières années, permet aujourd'hui d'accéder à de nombreux frameworks permettant l'accessibilité de ces technologies au plus grand nombre.

L'IA semble être en mesure d'avoir un impact profond, dans de nombreux domaines comme l'éducation ou la psychologie. L'objet de ce mémoire traite du développement d'un prototype

de reconnaissance faciale des émotions (REF), à travers les expressions qui a pour objectif principal de s'intéresser à l'analyse des comportements de l'apprenant dans un but de neuropédagogie. Cela peut être très utile dans l'adaptation des modèles d'apprentissage où, en analysant les comportements, on serait en mesure de s'adapter à l'apprenant (adaptive-learning).

Sur ce mémoire, la problématique sera abordée en développant un agent intelligent capable de reconnaître les émotions traduites par les expressions faciales, couplées au test sur le style d'apprentissage (Kolb [3]) permettant de définir un profil de l'apprenant.

Pour ce faire, nous utiliserons des algorithmes de deep-Learning et plus particulièrement les réseaux de neurones convolutifs (CNN [4]) dans la reconnaissance des émotions de bases définies par le psychologue P. Ekman [5].

L'idée est de se concentrer sur la démarche que privilégie chaque individu pour appréhender l'acte d'apprendre. Certains styles seraient plus adaptés dans certains contextes que d'autres. Tout dépend du point de vue de l'apprenant face à la situation d'apprentissage et de sa motivation.

Bien que des auteurs comme Jean Houssaye [6] démontrent que ce qui semble essentiel c'est de différencier la pédagogie bien plus que de déterminer des typologies destinées à connaître individuellement les apprenants et à dresser leur profil. Il paraît néanmoins intéressant de les utiliser comme complémentaires à un panel d'outils diversifiés que peut utiliser le formateur.

Initialement, pour mesurer des émotions, il est nécessaire de combiner plusieurs techniques classiques, comme la reconnaissance des expressions faciales, la reconnaissance de la parole et l'analyse des signaux physiologiques. Dans ce travail, je me concentrerai uniquement sur les expressions faciales pour la reconnaissance des émotions. Une expression faciale est une manifestation visible de l'état émotionnel, de l'activité cognitive, de l'intention, de la personnalité et de la psychopathologie d'une personne [5].

Partant du constat que Mehrabian [7] a mis en évidence le fait que 55 % du message émotionnel est communiqué par l'expression faciale alors que 7 % seulement par le canal linguistique et 38% par le paralangage, cette application est une étape générique sur l'identification des émotions.

Le premier chapitre a pour vocation de présenter l'étude préliminaire sur les principaux défis de la reconnaissance d'émotions au travers des recherches dans ce domaine. Nous énumérons les différentes sources d'informations et les modèles de représentation de l'émotion ainsi que les différentes méthodes de détection de l'émotion, développées au cours des dernières décennies.

Ensuite, nous exposons successivement, la chaîne de processus pour la détection de l'émotion (prétraitement, domaine de représentation des expressions faciales, prédictions) ainsi que les notions de bases sur les frameworks de deep-learning qui nous seront utiles et les différentes techniques d'analyse d'expressions faciales qui visent à reconnaître les émotions.

Je clôturerai ce chapitre par une revue des différentes bases de données nécessaires à l'apprentissage des systèmes automatiques de détection de l'émotion disponibles en open data pour entraîner nos algorithmes.

Le deuxième chapitre présente notre approche par l'intermédiaire des différentes étapes de conception d'un modèle de reconnaissance de l'émotion. Ceci comprend différentes étapes, comme le déploiement de l'application, le choix d'architecture fonctionnelle, les fonctionnalités et principaux composants, la base de données et le serveur. A cette étape, nous examinons les différents ensembles de données utilisés pour entraîner notre application, puis analysons nos résultats expérimentaux démontrant la pertinence ou non, de certains paramètres variables inhérents à notre méthodologie.

Le troisième chapitre présentera le développement et la gestion du projet, la procédure des choix des frameworks, les tests, la sécurité et l'exploitation des résultats. En mettant en relief les principaux résultats, nous identifierons les éléments ayant limité les performances ou pouvant les améliorer. Puis finalement, dans quelle direction de futurs travaux relatifs à notre étude, pourraient être conduits.

Chapitre 1 - Analyse et étude de faisabilité

Dans ce chapitre, nous fixons l'objectif à atteindre qui nous permet de faire une étude de l'existant d'un point de vue technologique pour arriver à se situer et savoir comment procéder. Nous détaillons notre étude préliminaire et notre méthodologie, puis nous décrivons les algorithmes et les différentes méthodes qu'il sera possible d'adopter pour atteindre l'objectif. Nous exposons les orientations qui nous ont amenés sur le sujet et nous ferons un bref historique des travaux dans le domaine des différents modèles de représentation de l'émotion. Nous développons les principales implications du développement d'un système REF ainsi que les éléments essentiels à sa conception. Nous couvrirons, par un examen détaillé, les différentes phases comme le prétraitement, l'algorithme d'apprentissage machine, et le post-traitement.

1.1 - Etude préliminaire et mode opératoire

Nous introduirons le schéma directeur de l'organisation de notre application informatique découpé en cinq phases et représenté par une étude de cas. Nous concluons ce chapitre par la description de quelques exemples de scénarios envisagés.

1.1.1 - Objectifs du prototype

A minima, DataTrainX est considéré comme un prototype, c'est-à-dire, qu'il doit permettre de se référer, de différentes manières interactives, à des scénarios d'utilisation du logiciel pour en homologuer les orientations. Nous devons étudier les grandes orientations techniques afin de déterminer sa faisabilité. A ce stade, nous sommes conscients que de nombreux facteurs restent à explorer, comme le choix des frameworks de deep-learning et les orientations futures de l'application en cas de succès. Des questions resteront donc en suspend selon la réussite du projet qui impacteront l'application finale sur un logiciel plus abouti, comme par exemple :

- Orientation vers l'adaptative learning : l'application devra proposer des supports de cours adaptés en fonction des résultats obtenus, typologie versus émotion.
- Orientation vers le service préparatoire ou la pré-orientation : l'application devra fournir un rapport détaillé en amont à la formation sur des orientations pédagogiques à utiliser comme outil complémentaire pour le stagiaire et le formateur.
- Orientation dans le domaine de la psychologie : l'application doit être testée à plus grande échelle en formation sur des établissements connexes, proposer plus de typologies et renforcer l'étude des résultats.
- Orientation vers l'open data : l'application peut être amenée à évoluer vers le micro-service afin de s'interconnecter avec d'autres outils complémentaires.

Notre application pourrait donc être méconnaissable par rapport au produit fini, car la simulation ne couvre que certains aspects de la stratégie finale. Par ailleurs, la Direction juge que l'application peut être mal considérée dans sa perception par notre public en situation d'handicap. Par précaution, nous n'utiliserons donc pas notre application lors de notre phase de test sur notre public jugé trop fragile. DataTrainX, nom de notre application, sera néanmoins disponible sur le réseau Internet, ouvert au public, et susceptible de fonctionner pour n'importe quel utilisateur. Cette application accessible en ligne sera testée par une soixantaine d'étudiants de l'université de Rodez, en partenariat avec le projet. Les utilisateurs pourront être sélectionnés sur des critères de profils en corrélation avec l'open data trouvée, qui servira de modèle étiqueté (RAVDESS [8] ou WIDERFACE [9] sur des profils jeunes européen/nord-américain par exemple). Nous ferons donc l'impasse sur les optimisations qui demanderont peut-être à terme, la réécriture de tout ou partie du prototype permettant son usage intensif, sans consommation excessive de ressources. Il est probable qu'il faudra adapter les frameworks, en vue d'une adaptation à un matériel graphique plus performant, mais plus onéreux. Le choix d'un développement de type prototype nous évite aussi de nous enfermer dans des choix prématurés, erronés, qu'ils nous seraient difficiles de remettre en cause une fois le programme écrit dans le langage final. D'un point de vue méthodologique, nous nous orienterons sur ce qu'on appelle le prototype vertical selon le modèle Nielsen [10] et une méthodologie agile de développement rapide d'application (RAD [11]). L'objectif est de mettre en œuvre certaines fonctionnalités de l'application afin que l'utilisateur puisse réaliser complètement un scénario typique d'utilisation du logiciel, ce qu'on appelle le test

d'utilisabilité (figure 1). Cela permet d'identifier les problèmes et d'analyser leurs causes. Des solutions sont élaborées et mises en œuvre dans une seconde version du prototype qui va faire l'objet d'une nouvelle série de tests, et ainsi de suite, jusqu'à ce que les problèmes soient corrigés. Les problèmes d'utilisabilités se réduisent à chaque itération, ce qui implique une recette RAD beaucoup moins conséquente qu'une recette classique.

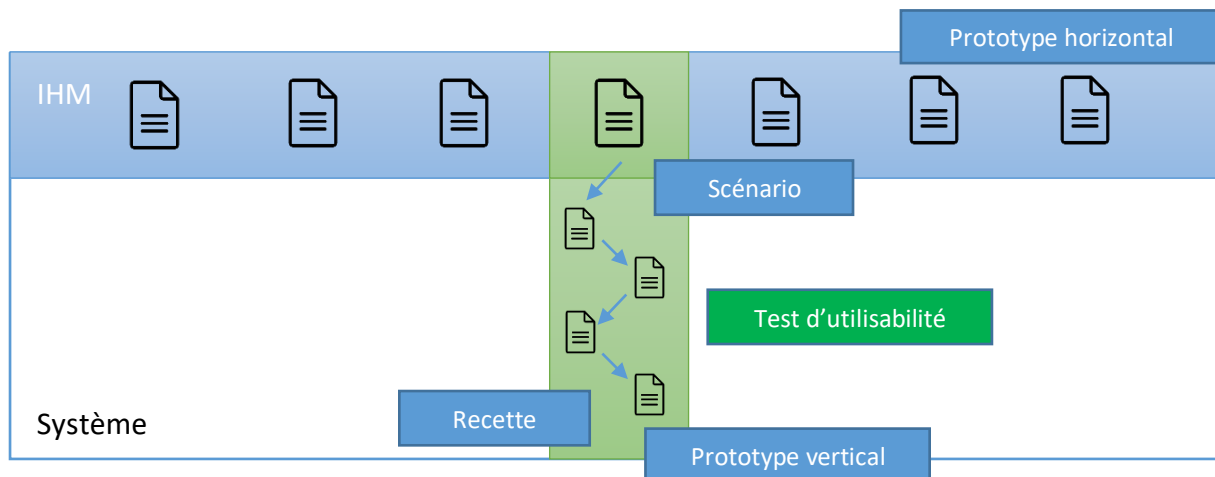


Figure 1 – Principe de développement de DataTrainX orienté sur le prototype vertical et le test d'utilisabilité. L'utilisateur doit réaliser complètement un scénario typique d'utilisation du logiciel avant de passer à une autre fonctionnalité.

L'utilisateur doit pouvoir simuler un scénario d'utilisation standard, une activité complète et significative du logiciel. Cela permet une plus grande adaptation sur le dimensionnement du réseau, les nécessités d'interaction, la logique des données, de l'interface et du niveau de réussite. En règle générale, trois phases de prototypage suffisent pour lever la plupart des problèmes. Pour finir, le prototype vertical nous permettra l'évaluation du volume de données à envisager. Associer à un développement agile de type RAD, cela nous permet de privilégier un délai de développement plus court, un coût en ressources moins risqué, une fiabilité plus grande du produit ainsi qu'un budget réduit. La méthode RAD permet rapidement d'obtenir des prototypes sans négliger la qualité applicative et technique.

1.1.2 - Méthode agile RAD et description globale des phases

Nous avons préconisé une méthodologie agile de développement rapide d'application (RAD) dont l'un des aspects principaux est l'absolu respect d'une dimension temporelle sur la livraison du projet qui ne doit pas dépasser 120 jours. Bien que nous n'ayons pas vraiment de

contrainte de temps sur ce projet, nous l'imposer, permet cependant de nous donner un rythme et un calendrier afin de ne pas nous disperser.

L'objectif de l'application est d'enregistrer sous forme de vidéo par la caméra du sujet, son expérience utilisateur, via la saisie d'un questionnaire et la résolution d'un puzzle. Le questionnaire sert à déterminer un profil d'apprenant et le puzzle est une mise en situation en vue de capter les états émotionnels de l'utilisateur par les expressions de son visage. Pour arriver à ces conditions, nous avons donc porté notre étude préliminaire sur un processus global défini en 5 phases et inspiré de la méthode RAD. La méthode RAD structure le cycle de vie du projet que nous décrivons par la suite, illustrée ci-dessous en figure 2.

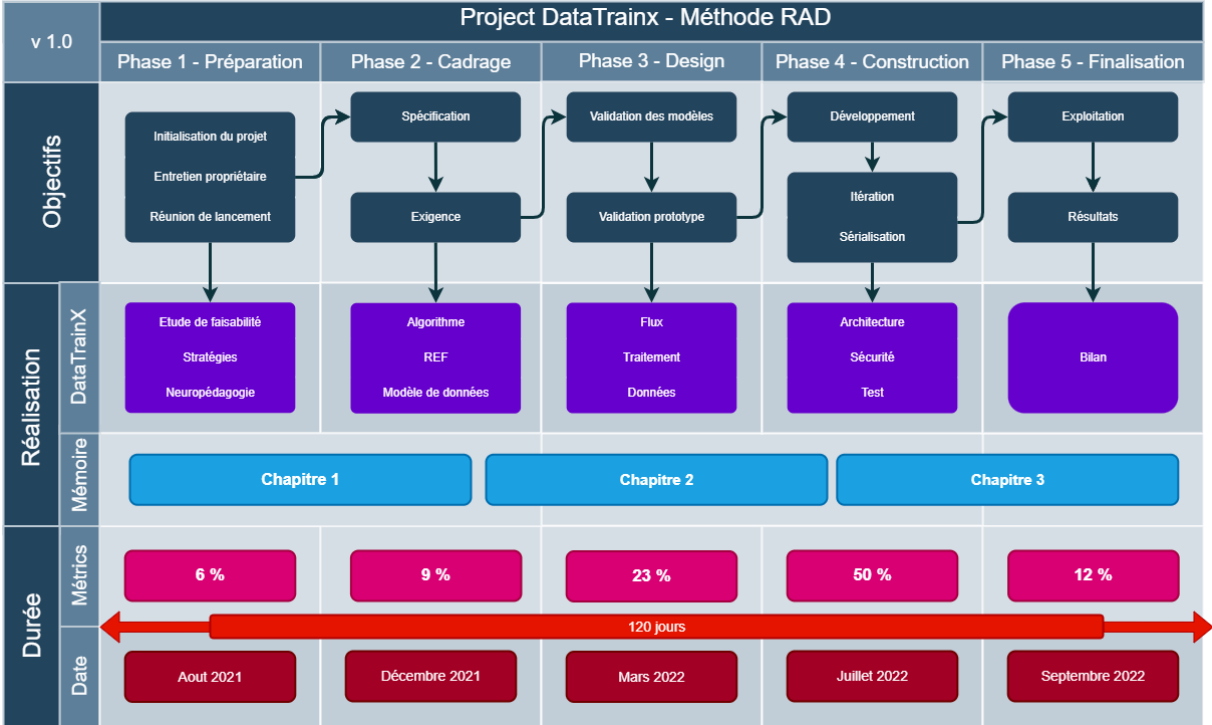


Figure 2 – Les 5 phases d’organisation de DataTrainX inspiré de la méthode RAD

La phase 1 (Préparation), est orientée sur la réflexion du mode opératoire. Cette phase permet de définir le périmètre général du projet, de structurer le travail par thèmes, de sélectionner les acteurs pertinents (psychologue, partenaires) et d’amorcer une dynamique de projet. C’est à cette occasion, que nous avons réalisé un inventaire des ressources algorithmiques disponibles en open source, les moyens techniques matériels à disposition (serveur dédié, puissance calcul vidéo, etc.), les bases de données disponibles en open data pour la reconnaissance faciale des expressions. Nous avons cadré aussi les attentes, les

résultats, déterminé un planning et une méthodologie. Cette phase représente environ 6% du projet.

La phase 2 (Cadrage), est l'analyse et l'expression des exigences. Nous avons ici réalisé les recherches adéquates de faisabilité en testant différents frameworks de deep-learning. Cela nous a permis de nous orienter sur la mise en œuvre de l'architecture, les stratégies de développement de l'application et les choix fonctionnels. Nous avons précisé les objectifs et besoins d'un logiciel de reconnaissance faciale des émotions, ce qui nous a orientés sur les bases de données modèles disponibles dans ce domaine. Cette phase représente environ 9% du projet. La phase de préparation et de cadrage sont respectivement abordées dans le chapitre 1 de ce mémoire.

La phase 3 (Design), est la conception et la modélisation. C'est à cette étape que nous avons affiné et validé les modèles organisationnels dans le traitement, les flux et les données. Nous avons validé le premier niveau du prototype, incluant l'ergonomie et la cinématique générale de l'application. C'est le domaine que nous aborderons dans la prochaine section avec une modélisation UML et des scénarios qui permettent la mise en œuvre d'une architecture de conception évolutive. Pour des raisons de réutilisabilité et d'efficacité, la phase design s'est réalisée en parallèle de la phase 4 « construction ». La parallélisation nous permet aussi d'anticiper le fait de se réajuster si la première phase d'utilisation est non concluante sur l'analyse des données. Elle vise à améliorer progressivement l'interface, en s'appuyant sur l'analyse du comportement des utilisateurs finaux, lorsqu'ils se servent du produit (phase test utilisateur). La phase 3 intégrera les tests, les choix des frameworks de deep-learning et la réflexion sur la collecte des données de masse et leur nettoyage (data-mining). Cette phase représente environ 23% du projet.

La phase 4 (Construction), est la réalisation finale du prototype. Elle comprend la mise en production et le déploiement de l'application auprès des sujets à tester. Du fait de la collecte des données par les saisies utilisateurs, cette phase concerne aussi l'exploitation des données et l'utilisation de modèle de comparaison sur les frameworks de deep-learning retenus. Il est prévu une phase de nettoyage et d'analyse des données, ce qu'on appelle aussi Data Mining (processus d'analyse de volumes massifs de données). La parallélisation avec la phase trois, prévue à ce stade, permet une refonte éventuelle du design si un problème significatif dans la collecte des données apparaît. D'où la nécessité de segmenter l'ensemble des fonctionnalités

à produire. C'est ce qu'on a abordé en figure 1 dans la notion de prototype vertical. Nous devons développer l'intégralité d'une fonctionnalité qui correspond à un scénario typique d'utilisation du logiciel, avant de passer à une autre fonctionnalité. Dans un prototype horizontal, nous aurions maqueté l'ensemble des IHM et privilégié un développement sur une architecture maintenable, ce qui ne sera pas le cas ici. Les tests d'utilisabilité, que l'on appelle aussi Focus en méthodologie RAD demandent l'intervention des utilisateurs. Comme illustré en figure 3, les tests d'utilisabilité sont susceptibles de déclencher des événements que le développeur n'aurait pas imaginés. A ce point, seules les divergences bloquantes et majeures sont corrigées. Cette opération implique la prise en compte des remarques et une étape complète d'intégration. Cette phase représente environ 50% du projet. La phase design et construction sont respectivement abordées dans le chapitre 2 de ce mémoire.

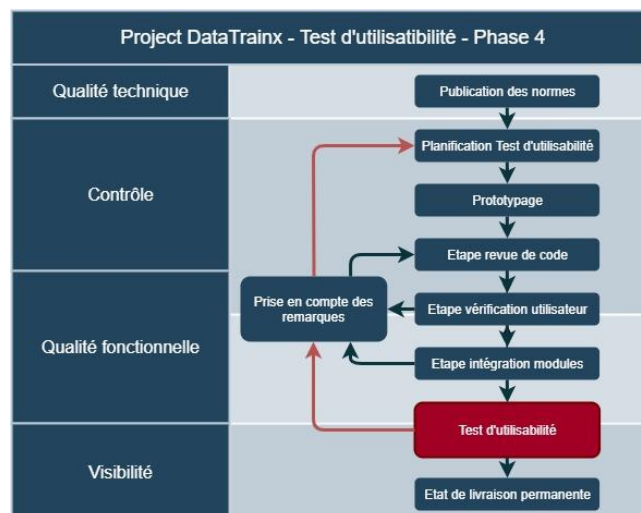


Figure 3 – Architecture de construction, test d'utilisabilité

La phase 5 (Finalisation), sera destinée à la réflexion sur la neuropédagogie, l'exploitation des résultats, la recherche de corrélations entre émotions et profil d'apprenant, le bilan du projet, son retour de connaissance et son éventuelle poursuite. Des recettes partielles ayant été obtenues à l'étape précédente, il s'agit dans cette phase d'officialiser une livraison globale, de transférer le système en exploitation et maintenance ou du moins, de l'entériner. Cette phase représente environ 12% du projet. Cette dernière phase sera abordée dans le dernier chapitre de ce mémoire.

Maintenant que l'organisation conceptuelle est définie, nous nous orientons sur l'un des premiers objectifs du système, qui est le traitement algorithmique sur la localisation du visage.

1.2 - L'orientation algorithmique

Nous avons pour objectif de réaliser une application informatique qui permettrait de corrélérer profil d'apprentissage et émotions. Nous avons nommé notre projet DataTrainX, en relation avec les mots « Train » pour « Former », « Data » pour données et « X » pour inconnues, qu'on pourrait traduire par « comment former au mieux un individu ». L'ensemble formant un mot évocateur dans les technologies de l'intelligence artificielle. L'intelligence artificielle, des algorithmes actuellement en plein essor, dont les promesses semblent pouvoir prédire les émotions d'une personne, mais qu'en est-il réellement ? Pouvons-nous l'envisager pour notre projet ? C'est la question que nous tenterons de démontrer dans cet article en faisant un état de l'art sur le sujet.

C'est sur la base des travaux du psychologue Paul Ekman publiés dans les années 1960 et 1970 qu'une typologie sur les émotions a été formulée dans leurs relations aux expressions faciales.

Ces travaux ont été repris en informatique à partir des années 1980, mais c'est surtout dans les années 2000, avec l'essor des réseaux sociaux, que des algorithmes puissants ont été développés. En effet, la multiplication de messages permettant d'inclure des images ou des vidéos, a entraîné une attention très particulière de la part des chercheurs de nombreuses communautés scientifiques, pour des raisons liées à la sociologie, la sécurité ou la business intelligence. Ce sont les besoins d'analyse d'images de ces nombreuses recherches qui ont entraîné le développement des algorithmes d'analyse de formes. Les réseaux sociaux utilisent ces résultats pour automatiser la reconnaissance d'expressions faciales à partir d'images ou de vidéos. Ces progrès ont suscité des investissements privés, universitaires et publics importants, notamment de la part des GAFAM (Google, Apple, Facebook, Amazon, Microsoft) [12].

Initialement ces algorithmes suivaient une logique d'apprentissage supervisée, cela signifie qu'ils sont soumis à un ensemble d'images étiquetées par des experts humains, qui déterminent si le visage de la personne est plutôt ovale, carré, rond, c'est ce qu'on appelle l'apprentissage automatique ou machine-learning.

La description fournie peut être illimitée comme la couleur de peau, l'origine ethnique, l'identité d'une personne ou encore, être associée à une émotion comme la peur ou la

tristesse. Pour compléter, l'algorithme est enrichi de descripteurs, qui sont un ensemble prédéfini de mesures de points clés du visage (forme du menton, de la mâchoire, des yeux...). On en compte une vingtaine. Puis vient l'analyse systémique, Grimm et al [13] proposent une approche traitant localement une région suivie d'une analyse plus fine, par exemple, sur une région centrée sur la bouche, pour un état de bouche "Ouvverte", la recherche se porte sur "Sourire" ou "Pas sourire" (voir la figure 4).

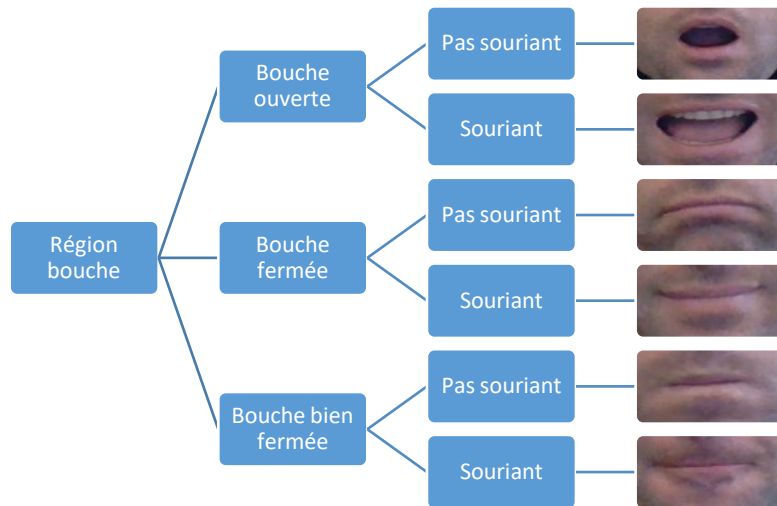


Figure 4 – Exemple de la description de Grimm et al. sous forme d'arbre de décision sur la région de la bouche.

Dans le cas qui nous intéresse, ils sont complétés de descripteurs d'émotions, beaucoup plus complexes et dont le nombre se rapproche d'un arbre de décision. Généralement chaque algorithme est spécialisé dans un domaine de reconnaissance et c'est l'ensemble des algorithmes, qui associent leurs résultats pour obtenir la classification désirée. L'algorithme d'apprentissage intervient sur ses historiques d'analyse, et regroupe de façon statistique la probabilité de trouver une image en lien avec l'étiquetage humain et une image non étiquetée. L'algorithme teste différents descripteurs et à chaque étape, il compare l'information humaine, les « mauvaises » réponses sont éliminées et renvoyées vers les niveaux en amont pour ajuster le modèle mathématique, c'est ce qu'on appelle l'apprentissage supervisé. L'algorithme est guidé avec des connaissances préalables de ce que devraient être les valeurs de sortie du modèle. Par conséquent, le modèle ajuste ses paramètres de façon à diminuer l'écart entre les résultats obtenus et les résultats attendus.

Les techniques ont évolué depuis 2010 et nous sommes désormais capables de réduire la supervision, c'est-à-dire de réduire le facteur humain dans l'étiquetage des informations. C'est

ce qu'on appelle des algorithmes d'apprentissage profond plus couramment appelés deep-learning. Les recherches dans ce domaine s'inspirent de la neuroscience, c'est-à-dire du fonctionnement du cerveau humain. L'algorithme s'efforce, grâce à des réseaux de neurones artificiels, de créer des modèles capables d'apprendre ces représentations à partir de données non labellisées à grande échelle. Prenons l'exemple, d'un jeu de données représentant des visages exprimant des émotions. On cherche à les regrouper en classes. Ici, nous ne connaissons pas l'expression du visage, mais nous voulons essayer de les regrouper. Ainsi, si les formes sur le visage de l'expression de la colère sont similaires alors elles sont en rapport avec une même expression correspondante. L'apprentissage profond utilise la machine de « Boltzmann restreinte [14] », caractérisée par des couches de neurones artificiels cachées et visibles, appelées réseau neuronal convolutif (CNN). Cela permet au programme de réorganiser les informations en blocs plus complexes. Lorsque ce modèle est par la suite appliqué à d'autres cas, il est normalement capable de reconnaître l'expression sur un modèle non renseigné par un humain. Concernant les émotions, les descripteurs ne sont pas spécialement prédéfinis, ils seraient trop nombreux. L'intérêt du deep-learning est de permettre à l'algorithme d'identifier lui-même une caractéristique de l'image qui lui permettrait d'améliorer son score de réussite par rapport à l'étiquette humaine. La figure 5 ci-dessous montre la différence dans l'évolution de l'algorithme supervisé à non supervisé avec le traitement des différentes couches permettant de classifier l'éléphant.

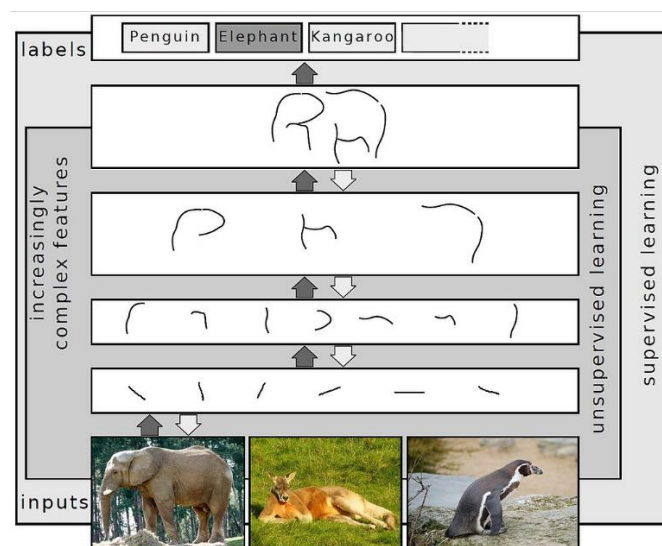


Figure 5 – En deep-learning, l'éléphant ici n'est pas étiqueté, du moins il n'est pas nécessaire de définir de nombreux descripteurs (trompe, oreille, etc.) l'algorithme arrive par lui-même à classifier l'image [15].

Sur les frameworks de deep-learning, nous avons réalisé différents tests afin de déterminer celui qui apporterait les meilleurs résultats. Plus de détails sur ce point seront fournis ultérieurement dans les chapitres suivants, mais ce qui s'avère finalement le plus important c'est le modèle d'entraînement, c'est-à-dire l'étiquette minimale renseignée par l'expert humain qui est à la base du mode de comparaison. Or cela dépend de critères psychologiques dont les recherches divergent lorsqu'il s'agit de les identifier. Plusieurs auteurs comme Izard [16], Ekman [5], Tomkins [17] ont des approches différentes sur leurs classifications. A notre niveau, nous avons pris parti de nous orienter plutôt sur le choix des modèles déjà disponibles sur le marché. La grande majorité des bases de données préparées pour la reconnaissance des émotions catégorisent les expressions faciales parmi six émotions de bases proposées par Ekman : colère, peur, tristesse, joie, dégoût, surprise. C'est le cas des bases comme « The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS) », MMI FEEDTUM, ou de Cohn-Kanade (CK+) dont la figure 6 est représentative.

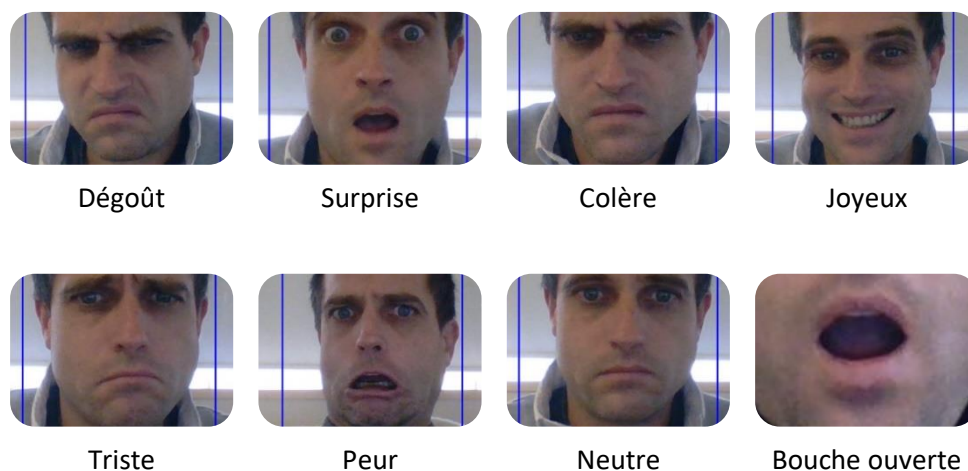


Figure 6 – Exemples d'images étiquetées que l'on trouve à disposition. Les six émotions de base (Ekman 2006) sont représentées sur le côté gauche ainsi que l'affichage de l'ouverture de la bouche sur le côté droit [18]

Ces bases peuvent être complétées d'émotions comme l'anticipation ou la confiance selon les travaux de Plutchik qui a organisé ses émotions primaires en paires d'opposés : la joie et la tristesse, la peur et la colère, le dégoût et la confiance, la surprise et l'anticipation [15].

Après avoir exposé les grandes orientations algorithmiques sur la captation des émotions, nous nous sommes orientés sur la manière de coupler les résultats avec une typologie d'apprentissage.

1.3 - Neuropédagogie et typologie d'apprentissage

La neuropédagogie est une discipline visant à améliorer l'apprentissage, nous verrons dans ce chapitre, comment l'exploiter par le développement d'un outil informatique. Nous nous baserons notamment sur la saisie d'un questionnaire permettant d'obtenir le profil d'un apprenant.

1.3.1 - Etude de la neuropédagogie

Toute situation d'apprentissage se construit à partir d'un environnement spécifique et d'un individu, qu'il soit élève, étudiant ou apprenant adulte. Dans l'acte d'apprendre, chaque individu adopte un comportement qui peut évoluer, s'adapter et dont le formateur doit tenir compte. Analyser la manière dont apprend un élève ou un stagiaire peut aider à diagnostiquer les difficultés d'apprentissage [1]. Pour le psychologue Jean Piaget [19] l'apprentissage résulte d'une interaction entre le sujet et son environnement, concept que confirmera le mathématicien Gérard Vergnaud [20], en indiquant que l'enseignement et l'apprentissage d'une discipline ne sont pas indépendants du domaine concerné. En neuropédagogie, le but est d'améliorer l'apprentissage en comprenant mieux le fonctionnement du cerveau. Elle est souvent associée aux neurosciences, mais elle est bien plus large et interdisciplinaire faisant appel à la pédagogie, la psychologie et les neurosciences [21]. La neuropédagogie permet d'analyser la manière dont apprend un stagiaire, d'aider à diagnostiquer les difficultés d'apprentissage et de détailler les différents profils cognitifs des individus afin que le formateur s'adapte et en tienne compte. Ce sont des concepts qui proviennent du monde de l'entreprise dont l'objectif était la recherche d'efficacité dans l'acquisition de compétences, notion que j'illustre dans la figure 7 ci-dessous, en citant une phrase de Sarralié [22].



Figure 7 – «Adapter c'est éviter de transformer une situation d'apprentissage en situation de handicap.» [22].

Ces concepts s'orientent sur les manifestations et préférences individuelles de traitement de l'information liées au système de pensée, et s'appuient sur l'étude de la personnalité comprenant la culture, le style empirique (l'expérience), le style rationnel (la cohérence logique) et le style métaphorique (l'intuition).

L'idée est de se concentrer sur la démarche que privilégie chaque individu pour appréhender l'acte d'apprendre. Certains styles seraient plus adaptés dans certains contextes. Tout dépend du point de vue de l'apprenant face à la situation d'apprentissage et de sa motivation. Le style d'un apprenant devient alors le processus opératoire issu de l'interaction entre l'individu, le contexte et le mode privilégié de fonctionnement. Le formateur doit alors apprendre à distinguer ce qui relève tantôt d'une préférence de surface modulable selon les contextes, tantôt d'un processus cognitif stable et immuable.

Bien que des auteurs comme Jean Houssaye [6] démontrent que ce qui semble essentiel c'est, différencier la pédagogie bien plus que de déterminer des typologies destinées à connaître individuellement les apprenants et à dresser leur profil. Il paraît néanmoins intéressant de les utiliser comme complémentarité à un panel d'outils diversifiés que peut utiliser le formateur. C'est ce qui définit la neuropédagogie qui est donc une discipline qui fait le lien entre "l'ordinateur" (le cerveau) et les "logiciels" (connaissance, compétence, savoir, savoir-être, information...). Son champ d'application est donc large.

Le projet DataTrainX a pour objet de mettre en application, sous forme algorithmique, le principe de la neuropédagogie qui est fondamentalement pluridisciplinaire. Le projet s'appuie sur une phase de prototype. L'importance n'est pas tant d'obtenir des résultats significatifs sur la pédagogie, à même de déterminer si le projet peut avoir une portée concrète en neuropédagogie. L'enjeu porte plutôt sur l'intérêt pratique de développer à une échelle réduite, une application informatique, permettant de mettre en œuvre ces principes avec le deep-learning. Le sujet consiste à réaliser une application qui à travers un questionnaire défini, engendre un profil d'apprenant associé aux expressions émises.

1.3.2 - Typologie d'apprentissage

Pour déterminer le choix du questionnaire, nous nous basons sur le concept de style cognitif proposé par Allport en 1930. Il existe de nos jours une profusion des typologies proposées par

la littérature depuis des décennies pour désigner les styles. On peut cependant les classer en 5 familles de modèles couramment utilisés de nos jours :

- Les modèles centrés sur des facteurs génétiques (visuel, audio, kinesthésique, tactile...), comme celui de Dunn et Dunn [23].
- Les modèles centrés sur des facteurs cognitifs, très en vogue actuellement (différences individuelles dans la manière dont nous percevons, pensons, résolvons les problèmes, apprenons, sommes liés aux autres), comme celui de Riding [24].
- Les modèles centrés sur des facteurs stables de la personnalité (outils les plus utilisés pour analyser la personnalité d'un individu et sa façon d'interagir avec les autres), comme celui de Apter [25] ou de Myers-Briggs [26].
- Les modèles centrés sur des préférences contextuelles évolutives, mais stables (préférence individuelle qui varie légèrement en fonction du contexte. Test le plus influent dans le monde), comme celui de Allinson et Hayes [27] ou de Kolb [28].
- Les modèles centrés sur des stratégies d'apprentissage (basés sur les stratégies ou approches d'apprentissage, prennent en compte les effets de l'expérience et l'influence du contexte comme celui de Entwistle [29] ou Vermunt [30]).

En 2004, Coffield [31] et son équipe ont mené une étude parmi 71 de ces modèles qu'ils ont identifiés. Les chercheurs ont sélectionné 13 modèles parmi les plus influents, dont ceux que nous citons plus haut. Leurs objectifs étaient de déterminer une validité scientifique sur ces modèles et de démontrer une cohérence (l'instrument de mesure doit démontrer une cohérence interne), être fiable (mêmes réponses sur au moins 2 tests), avoir une validité conceptuelle (vérifier une validité théorique) et une validité prédictive (permettre de prévoir des faits à partir des éléments donnés).

Bien que l'étude de Coffield et l'utilité du concept de « styles d'apprentissage » en éducation soient parfois contestés et selon Luc Rousseau toujours une « hypothèse de recherche encore en quête de validation » [32], nous retiendrons les résultats de cette étude. Les modèles les plus scientifiques que nous mettons en annexe (1) et en bibliographie comme le CSI de Allinson et Hayes [27], le test de Kolb utilisé par l'Union européenne [28] ou encore celui de Myers-Briggs [26]. C'est donc à partir de cette étude que nous avons déterminé le choix du questionnaire de Kolb pour le projet DataTrainX.

1.3.3 - Le modèle de l'apprentissage expérientiel de Kolb

Les modèles de styles d'apprentissage individualisés ont donc pour objectif de définir un type d'apprenant afin de permettre aux enseignants d'adapter leurs méthodes d'apprentissage. Peu d'études ont réussi à valider le concept de styles d'apprentissage en éducation. L'étude de Pashler et al [33] indique bien une corrélation entre typologies et les préférences exprimées par certains individus sur la façon dont ils préfèrent assimiler l'information. Mais cette même étude contredit le fait que l'apprenant assimilerait mieux avec une méthode jugée comme appropriée à son style d'apprentissage. Nombreux sont les scientifiques qui parlent de mythe ou invitent les professionnels de l'éducation et de l'apprentissage à faire preuve de scepticisme devant ces concepts. Le modèle de Kolb n'échappe pas non plus aux critiques, mais il est de loin celui qui a connu la plus large diffusion et celui qui est à la base de modèles d'autres chercheurs. Il est inspiré des travaux de psychologues reconnus comme John Dewey, Kurt Lewin et Jean Piaget. Il est plutôt approprié pour les apprentissages de disciplines académiques relativement abstraites, mais moins adapté pour la plupart des apprentissages professionnels. D'un autre côté, notre cible d'étude est réalisée sur des universitaires en informatique, ce test nous semble donc plutôt approprié.

Ce qu'il faut retenir des principales critiques, c'est qu'étiqueter un apprenant et le cantonner dans une posture est strictement contre-productif et va à l'inverse de l'objectif. Néanmoins, ce qu'il ressort aussi, ce sont des concepts génériques qui fonctionnent. Par exemple, un profil « indépendant » a tendance à questionner les novices alors que le profil « socialisant » propose des aides. Les « socialisants » s'adaptent à leur interlocuteur, quel que soit le style du novice. Ceux qui travaillent avec un expert « socialisant » progressent le plus. On peut donc constater que les styles d'apprentissage nous ouvrent une voie pour des recherches sur les interactions sociales entre formateurs et apprenants. Pour des raisons à la fois pratiques et théoriques, il faut retenir la « maniabilité » des styles. Cependant ils paraissent suffisamment stables pour qu'on puisse les prendre en compte dans une situation donnée afin de contribuer à expliquer ces comportements d'apprentissage. Enfin, garder à l'esprit qu'en modifiant de façon significative la situation d'apprentissage, il est possible qu'un individu modifie son style préférentiel d'apprentissage. Le facteur que l'on souhaite ajouter à notre projet est l'expression de l'émotion corrélée au type d'apprentissage.

1.3.4 - Typologie d'apprentissage et émotions

Il y aura donc une phase de saisie d'un questionnaire afin de déterminer un profil d'apprenant de type Kolb dans le projet DataTrainX. L'utilisateur sera enregistré en vidéo et une comparaison sera effectuée entre la typologie trouvée par le questionnaire et les émotions qu'il aura exprimées. Comme l'illustre la figure 8 ci-dessous, l'objectif est de trouver une corrélation entre la typologie d'apprentissage et le panel d'émotions de l'utilisateur.

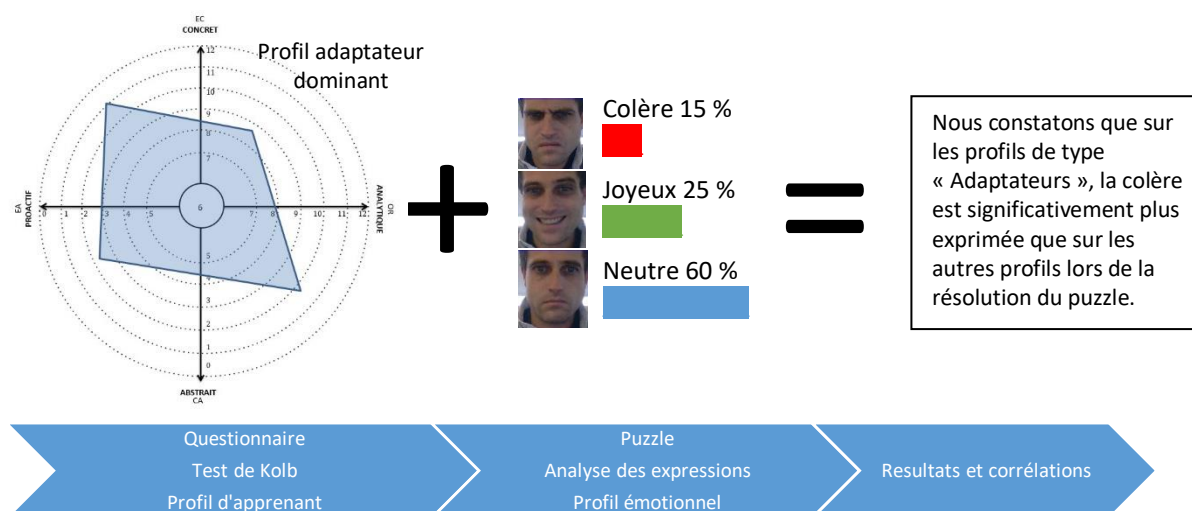


Figure 8 – Exemple de restitution des résultats de l'application DataTrainX. Obtenir un profil d'apprenant, analyser l'émotion dans une mise en situation, trouver une corrélation.

Dans l'hypothèse où nous aurions des résultats significatifs entre expressions et typologies, nous pourrions envisager que le profil émotionnel détermine la typologie d'apprentissage. Nous pourrions ainsi envisager, par exemple, une orientation du logiciel vers l'adaptive learning, c'est-à-dire une interaction entre l'IA et l'utilisateur dans la proposition des supports pédagogiques.

L'objet de ce projet, et donc de ce mémoire, a pour orientation de se focaliser concrètement sur les moyens fonctionnels informatiques à mettre en œuvre pour ce type d'application. Nous n'approfondirons pas l'aspect psychologique, qui se limitera au test de Kolb et à la résolution d'un Puzzle. Le but est de démontrer comment mettre en place l'architecture fonctionnelle, logicielle et matérielle pour arriver à un résultat applicatif de la reconnaissance faciale de l'émotion. Si aucune relation n'est trouvée, entre émotion et typologie d'apprentissage, cela indiquera éventuellement que le test de Kolb et/ou la mise en situation ne sont pas significatifs. Or la finalité n'est pas de changer de questionnaire, mais de tester différentes méthodes de deep-learning et de démontrer en termes d'ingénierie informatique

comment le réaliser. Les résultats d'ordre psychologique à prendre en considération dans l'analyse de ces données auront donc une part importante de subjectivité clairement assumée. Il s'agit d'une part de permettre à l'établissement une montée en compétences dans ce domaine afin de l'utiliser, d'autre part, d'acquérir une culture d'adaptive learning. Dans une approche inclusive, cet applicatif devrait permettre à travers des évaluations progressives de cibler les parties de la formation les moins bien assimilées par le stagiaire et fournir des méthodes supplémentaires ainsi que des exercices d'application mieux ciblés.

Cette section clôt le chapitre sur la neuropédagogie et les typologies d'apprentissage, elle permet de conclure avec l'approche algorithmique les objectifs stratégiques du projet. Nous allons dorénavant nous orienter sur la faisabilité technique de l'application et en premier lieu, sur la première étape d'un logiciel de reconnaissance faciale des émotions qui est la localisation du visage.

1.4 - La localisation du visage

Dans ce chapitre, nous introduirons la problématique de la localisation du visage et de l'orientation de la méthode Viola et Jones [34]. Nous présenterons ensuite les fonctions de représentations et de classifications des données. La dernière partie du chapitre décrira les métriques d'évaluation utilisées et le taux de réussite.

1.4.1 - Étape préliminaire

Pour réaliser un système d'analyse des émotions, il faut tout d'abord développer un algorithme de localisation du visage. C'est une tâche préliminaire nécessaire à la plupart des techniques d'analyse des expressions du visage. Elle constitue la première étape de notre système permettant de définir une région d'intérêt pour l'extraction des caractéristiques.

De nombreux travaux ont été effectués à ce sujet et plusieurs techniques sont proposées pour localiser un visage. Il existe l'approche basée sur l'apparence, l'approche basée sur les connaissances acquises, le template-matching et enfin l'approche basée sur des caractéristiques invariantes. Chacune de ces techniques a des avantages et des inconvénients, mais ce que nous retiendrons ce sont les approches basées sur l'apparence. En effet, elles ont l'avantage d'obtenir les meilleurs résultats avec pour inconvénient un temps de calcul plus conséquent et une phase d'apprentissage plus difficile à mener. Pour le temps de calcul, notre projet s'est donné un budget raisonnable en matériel pour combler ce problème.

Pour la phase d'apprentissage, nous n'avons pas de contrainte de temps, et pouvons utiliser une phase de recherche et développement, néanmoins comme expliquée sur la méthodologie RAD, nous jalonnons nos expériences en évitant de tester toutes les solutions. On se fie donc aux retours d'expériences génériques et on constate que beaucoup de frameworks, en raison des performances, proposent déjà des types d'approches basés sur l'apparence.

C'est ce que propose, par exemple OpenCV [35] avec notamment la méthode de Viola et Jones couplée à l'extension de Lienhart qui est une méthode inventée pour détecter des visages basée sur l'apparence. La bibliothèque OpenCV a aussi l'avantage de tester ces algorithmes de manière interactive, ce qui est plutôt pratique pour obtenir des résultats rapides. La méthode de Viola et Jones se base principalement sur l'extraction de 14 caractéristiques

pseudo-Haar [36] (figure 7) et d'arbres de décision pour la classification, permettant ainsi de calculer très rapidement les caractéristiques d'une image. De nombreux articles reconnaissent cette méthode comme la plus performante à l'heure actuelle. L'algorithme de détection de visage Viola et Jones peut ainsi se résumer en 4 phases clefs qui sont, la sélection des fonctionnalités Haar, la création d'une image intégrale, la formation Adaboost et la classification en cascade.

1.4.2 - Les caractéristiques pseudo-Haar et image intégrale

Les caractéristiques pseudo-Haar sont des fonctions permettant le calcul du contraste entre deux régions adjacentes, représentées par des régions en noir et des régions en blanc. Plutôt que de travailler directement sur les valeurs de pixels, ces caractéristiques facilitent l'extraction de plusieurs informations comme les lignes, les contours et les centres.

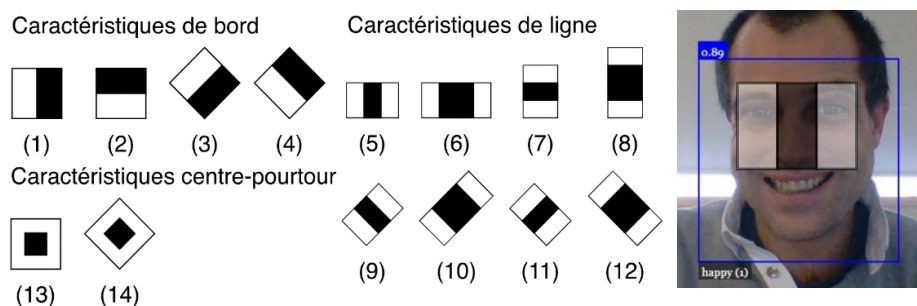



Figure 9 – L'extension des caractéristiques pseudo-Haar [36], à droite une caractéristique de ligne (5) trouvée par la variation de l'intensité de la lumière entre les yeux et le nez.

Ces caractéristiques sont en fait des masques, permettant de déterminer plusieurs patterns, des caractéristiques très simples mais très nombreuses. Ces caractéristiques permettent de détecter des motifs. Par exemple, la reconnaissance des visages est rendue possible par la variation de l'intensité de la lumière entre les yeux et le nez (figure 9).

Ce qui est avantageux en matière de calcul, c'est que les filtres n'ont que deux valeurs (1 pour le blanc et -1 pour le noir). Les valeurs des filtres sont donc issues d'une addition et d'une soustraction, beaucoup plus rapide en calcul que des multiplications et des divisions.

Voyons concrètement comment cela fonctionne pour déterminer le bord d'un visage  (1), exprimé H_a :

1. Une zone prédéfinie parcourt l'image, disons un carré de 20px (figure 10)
2. Dans cette zone on applique le calcul du vecteur, $V_A[n,m] = \sum (\text{intensité des pixels dans la zone blanche}) - \sum (\text{intensité des pixels dans la zone noire})$
3. Donc le coût total en terme de calcul pour une image est de :
 $(N \times M - 1)$ additionné par pixel par filtre par échelle



Figure 10 – Balayage de type raster

C'est donc un calcul relativement efficace, mais cette notion de rapport entre pixel, filtre et échelle, nous amène à nous poser la question, pouvons-nous faire mieux ? C'est là qu'intervient en complément, des caractéristiques pseudo-Haar, ce qu'on appelle le traitement « image intégrale [37] ». C'est une représentation sous la forme d'une image, de même taille que l'image d'origine, qui en chacun de ses points contient la somme des pixels situés au-dessus de lui et à sa gauche, ainsi que lui-même (figure 11).

98	110	121	125	122	129
99	110	120	116	116	129
97	109	124	111	123	134
98	112	132	108	123	133
97	113	147	108	125	142
95	111	168	122	130	137
96	104	172	130	126	130

Image

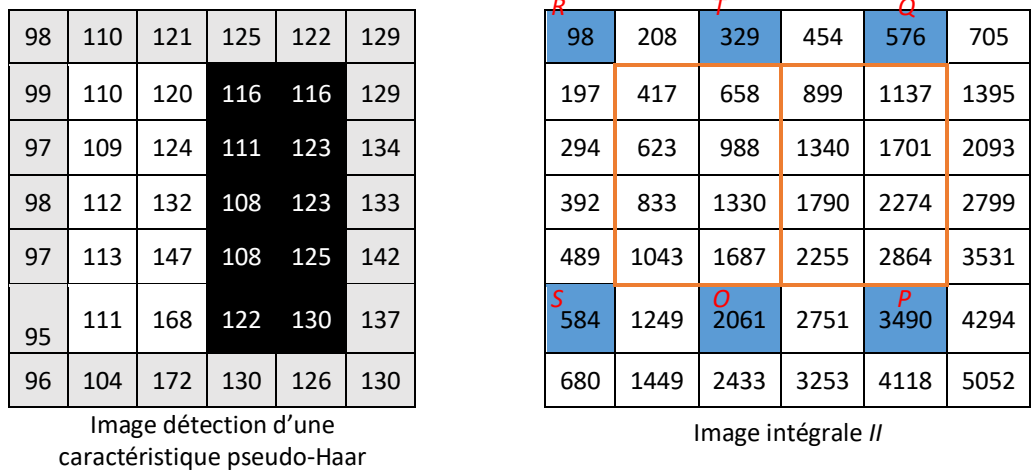
98	208	329	454	576	705
197	417	658	899	1137	1395
294	623	988	1340	1701	2093
392	833	1330	1790	2274	2799
489	1043	1687	2255	2864	3531
584	1249	2061	2751	3490	4294
680	1449	2433	3253	4118	5052

Image intégrale

Figure 11 – L'image intégrale comprend la somme de l'intensité d'un pixel additionnée de toutes les intensités pixels au-dessus de lui et à sa gauche

En utilisant l'image intégrale, on peut calculer la valeur de n'importe quelle somme de rectangle dans un temps constant. Grâce à cette représentation, une caractéristique pseudo-Haar formée de deux zones rectangulaires (1), peut être calculée, avec uniquement des

additions, en seulement six accès à l'image intégrale (figure 12). C'est donc un temps constant quelle que soit la taille de la caractéristique [38].



$V_A = \sum (\text{intensité des pixels dans la zone blanche}) - \sum (\text{intensité des pixels dans la zone noire})$

$$= (I_O - I_T + I_R - I_S) - (I_P - I_Q + I_T - I_O)$$

$$= (2061 - 329 + 98 - 584) - (3490 - 576 + 329 - 2061) = 64$$

Figure 12 – Calcul du vecteur sur un filtre pseudo-Haar à l'aide de l'image intégrale.

Dans cet exemple, nous avons pris le filtre le plus simple dans son calcul, d'autres filtres sont plus techniques. La puissance de l'image intégrale vient du fait qu'elle est indépendante de la taille du filtre que l'on applique et son calcul ne se fait qu'une fois.

Maintenant que nous avons à chaque pixel un vecteur de caractéristiques, nous voulons savoir si c'est un visage ou non, c'est là qu'intervient la dernière partie de la reconnaissance d'un visage, la classification.

1.4.3 - Formation Adaboost et classificateurs en cascade

Adaboost est un algorithme d'optimisation de catégorie boosting. Il est couramment utilisé pour le classement des images en intelligence artificielle. Sa découverte a valu à ses auteurs Yoav Freund et Robert Schapire, le prix Gödel en 2003. Adaboost se base sur un apprentissage supervisé comprenant des caractéristiques pseudo-Haar. Il est réalisé sur un ensemble d'images positives « c'est un visage » et négatives « ce n'est pas un visage ». Le principe est

de classifier les caractéristiques d'une image et de déterminer le poids des différents vecteurs par corrélation croisée. Ce qui nous permet par statistique, de désigner la covariance des vecteurs aléatoires X et Y, de quantifier l'écart conjoint entre deux vecteurs par rapport à leurs espérances respectives (figure 13, point 1).

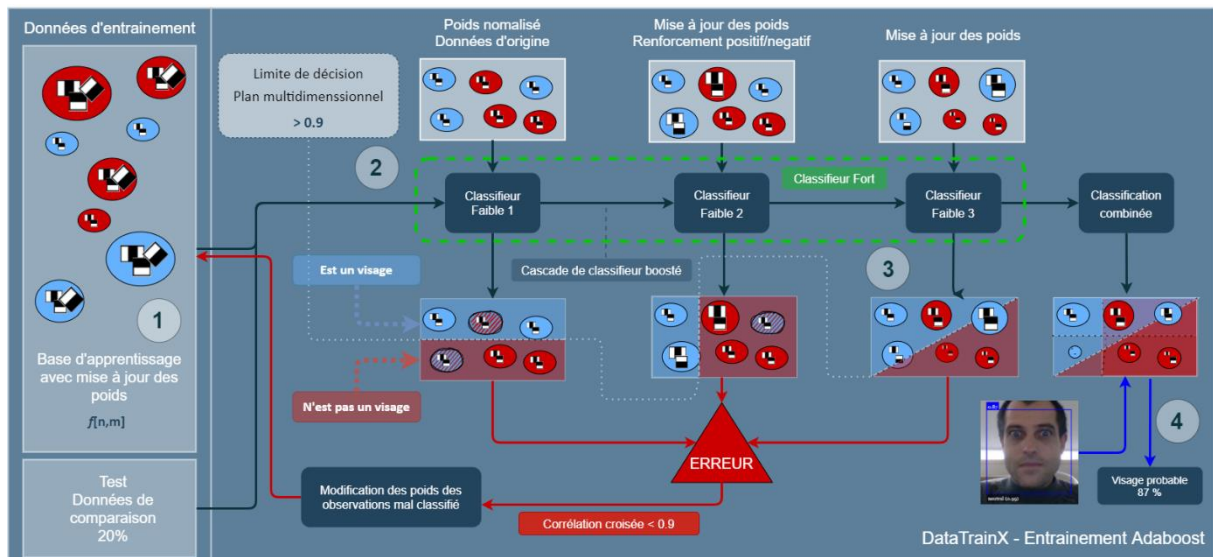


Figure 13 – Classification en cascade permettant une recherche rapide du visage

Adaboost crée donc un modèle de classification permettant à partir d'éléments pseudo-Harr de déterminer dans une image inconnue s'il y a la présence d'un visage ou non. Le principe consiste à tester différentes fenêtres de l'image, dont la taille et la position varient, sous forme d'arbre de décision à deux feuilles maximum « stumps ». Ces derniers sont créés séquentiellement et de poids différents, ce qu'on appelle les « classifieurs faibles ». Le but est d'obtenir un taux de vrais positifs très élevés (figure 13, point 2). Pour atteindre cet objectif, Viola et Jones proposent d'utiliser plusieurs classifieurs faibles successifs, organisés en cascade, c'est ce qu'on appelle un « classifieur fort ». Dès qu'un classifieur fort détermine par corrélation croisée que la fenêtre « n'est pas un visage », alors le processus s'arrête. Cette approche, illustrée sur la figure 13 (point 3), permet de rejeter la majorité des fenêtres dans les premiers niveaux de la cascade, ce qui assure des performances élevées. Le nombre de classifieurs faibles peut être ajouté jusqu'à l'obtention de performances conformes aux taux de détection et de fausse alarme souhaités pour l'étage. C'est ce qu'on appelle aussi la « limite de décision ». L'apprentissage est réalisé sur un très large ensemble d'images positives et négatives. Après la phase d'apprentissage, les différents seuils pour chaque classifieur sont enregistrés dans un fichier de métadonnées généralement inférieur à 10 Mo. C'est ce fichier

qui sera exploité lors de la phase de détection figure 13 (point 4) et figure 14. Viola et Jones ont testé leur algorithme sur la base de visages constitués de 130 images contenant 507 visages de face. Ils présentent leurs résultats sous la forme d'une courbe qui donne le taux de détection correct en fonction du nombre de fausses alarmes totales sur toutes les images du corpus. À titre d'exemple, pour 50 fausses alarmes, ils obtiennent un taux de détection de 88,8 %, sachant que de très nombreuses améliorations ont été proposées par la suite. Comparativement à d'autres détecteurs similaires, pour des taux de détection et de fausse alarme comparables [34], l'approche Viola et Jones est 15 fois plus rapide que celui de Rowley-Kanade et 600 fois plus rapide que celui de Schneiderman-Kanade. La détection du visage sur la photo ci-dessous (figure 14) a été obtenue avec le framework OpenCV qui utilise l'approche Viola et Jones. Nous y observons l'importance de l'entraînement entre un modèle WIDERFACE de 5.4 MB (ssd_mobilenetv1_model) qui donne un taux de confiance de 99% et un modèle de 190 KB (tiny_face_detector_model) moins fiable, avec une erreur et un taux de confiance de 86%, pour quasiment le même temps de calcul.

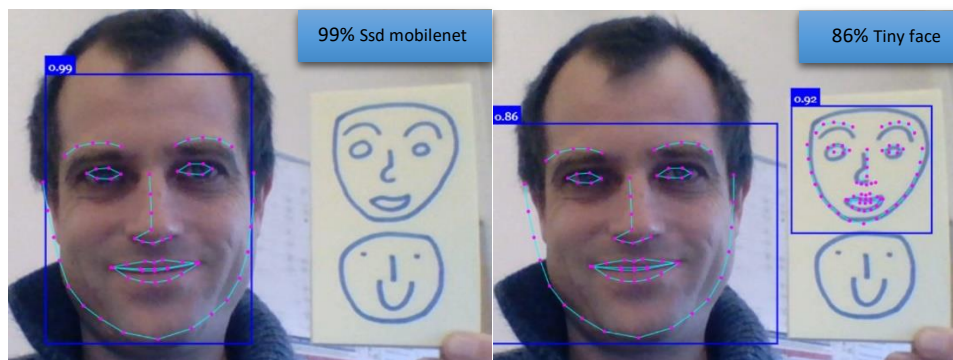


Figure 14 – La détection des visages comprend les visages issus de l'identification et des photos. On observe la différence de résultats entre deux modèles entraînés sur des limites de décision différents.

La méthode de Viola et Jones comporte cependant des limites. Un visage dans la position de profil n'est pas détectable. En revanche, cette limite n'influe pas sur notre système puisque nous ne traitons pas les occultations et l'ensemble des sujets sont face à la caméra. D'autres techniques auraient pu être utilisées, parfois plus précises, mais qui demandent plus de temps [39]. L'application Datatraining sur la détection du visage, donne des taux de détection supérieurs à 90%, avec OpenCV sur la base WIDERFACE en mode interactif. Une fois le visage localisé, sous la forme d'une zone périmétrique sur l'image (figure 14), il s'agit d'en extraire les informations pertinentes qui conduiront à l'analyse des expressions. C'est ce qu'on appelle la reconnaissance d'expressions faciales (REF) et qu'on approfondira dans la section suivante.

1.5 - La reconnaissance d'expressions faciales (REF)

Dans ce chapitre nous dresserons une liste non exhaustive des différents travaux sur la REF. Nous présenterons notre approche et le système que nous avons retenu. Ce chapitre a pour objectif de contextualiser et positionner les recherches dans ce domaine et plus particulièrement, de nous orienter sur les critères d'influence à prendre en considération pour DataTrainX.

1.5.1 - Caractérisation des expressions faciales

Après une première étape de localisation du visage, il est nécessaire de détecter certains points de caractéristiques du visage. En effet, le visage humain permet à une personne d'envoyer des signaux sociaux comme les expressions faciales (EF). Les EF sont la corrélation des changements des muscles faciaux sous la peau du visage. Ils sont une facette principale pour estimer les activités liées aux émotions, telles que notre centre d'attention actuel, notre état émotionnel, la compréhension ou le désaccord. L'analyse automatisée des expressions faciales est une tâche complexe, influencée par de nombreux critères que nous allons énumérer (figure 15).

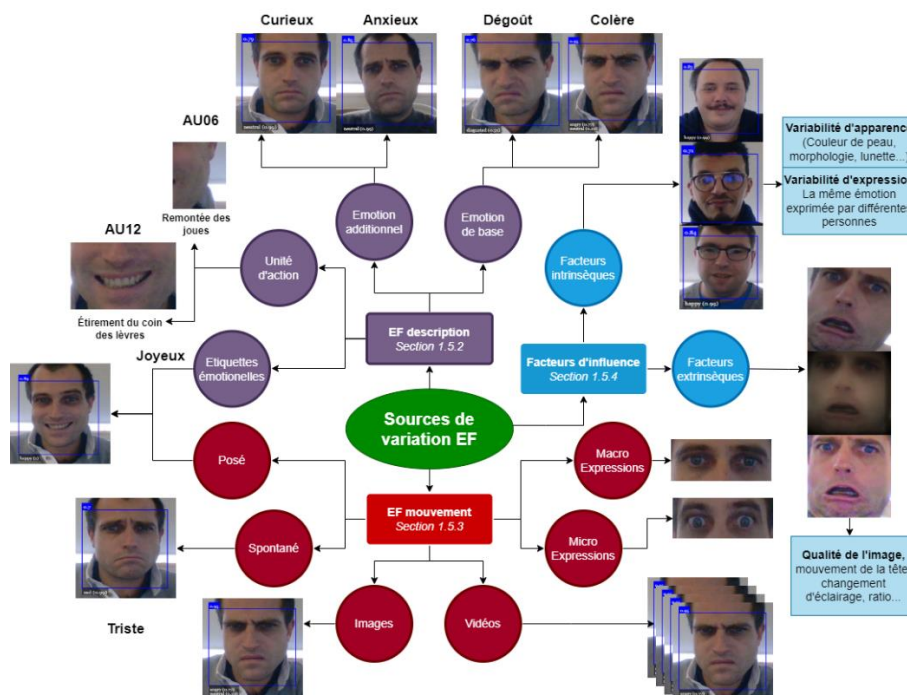


Figure 15 – Schéma sur les influences de différents facteurs à prendre en considération sur la reconnaissance des expressions

Premièrement, cela comprend : le niveau de description de l'EF, les étiquettes par rapport aux attributs visuels du visage, leurs descriptions par unité d'action (AUs) via le système de codage d'action faciale (FACS). Mais aussi, le type de catégories d'émotions, que nous avons déjà décrit et approfondirons avec les émotions non basiques (Section 1.5.2). Deuxièmement, nous nous intéresserons sur la spontanéité de l'EF en termes d'apparence visuelle qui justifie notre approche sur les cas d'utilisation casting (pose de l'expression) et puzzle (spontanéité de l'expression) afin de les différencier. En effet, il existe des subtilités, de vitesse et de mouvement sur le type d'EF, ce que l'on définit comme macro et micro mouvements des muscles faciaux. La source d'entrée, images ou vidéos, a donc une influence sur l'analyse de l'expression (Section 1.5.3). Puis nous terminerons, sur les facteurs d'influence intrinsèques et extrinsèques sur la reconnaissance des émotions, et la méthodologie que cela implique (Section 1.5.4).

1.5.2 - Unités d'action et système de codage d'action faciale

La détection des émotions, associée à la détection de l'action des muscles, est l'approche de Jeffrey F. Cohn [40] qui domine actuellement dans l'interprétation d'une EF. Elle consiste à décrire des mouvements musculaires faciaux en unité d'action (AU) via le système de codage d'action faciale (FACS) réalisé par Ekman et Friesen [41]. Le FACS est un système basé sur l'anatomie pour décrire toutes les déformations visuelles du visage. Les AU avec leurs descriptions décrivent les actions du visage en ce qui concerne leur emplacement ainsi que leur intensité. Dans le système FACS il existe 46 AU identifiés. C'est une combinaison d'AU avec leur intensité qui est déterminée pour représenter une émotion. Par exemple, l'association AU06 et AU12 (voir figure 15) définit la joie. Dans la figure 16, la surprise est représentée par le code AU01+AU02+AU05B+AU26, qui se traduit par « Remontée de la partie interne des sourcils », « Remontée de la partie externe des sourcils », « Ouverture entre la paupière supérieure et les sourcils d'intensité légers » et « Ouverture de la mâchoire ».

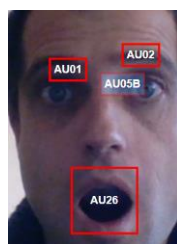


Figure 16 – Représentation de la surprise avec le système FACS AU01+AU02+AU05B+AU26

Nous avons décrit les émotions universelles de bases d'Ekman, mais il existe une pluralité d'émotions plus subtiles qui peuvent dépendre du contexte culturel. Elles ne sont pas aussi bien définies que les émotions de bases, le mouvement du muscle est parfois moins fort, c'est la raison pour laquelle elles ne sont pas référencées dans le système FACS. Les émotions de bases ne comprennent qu'un petit sous-ensemble des états mentaux que les gens peuvent exprimer et qui sont sans doute les plus fréquents dans les interactions quotidiennes. Nous pourrions citer par exemple, les émotions comme le doute, la soif, la douleur, la dépression, l'anxiété, l'inquiétude, la panique, la fierté, l'embarras, la honte ou encore la jalousie. Peu de chercheurs ont commencé à explorer la manière de les modéliser, ce qui explique le peu d'approches informatiques dans ce domaine. Nous avons conscience que les émotions complexes non fondamentales sont plus proches des comportements naturels humains, mais nous sommes limités par la recherche dans ce domaine. Aussi nous nous baserons pour notre application que sur les émotions de bases.

1.5.3 - Le mouvement de l'expression faciale

Les expressions faciales sont généralement classées comme délibérées (posées) ou spontanées, comme indiqué dans la figure 13. Les expressions faciales spontanées diffèrent des expressions posées à la fois dans les muscles, qui sont déplacés et dans la dynamique du mouvement. On parle aussi de macro et de micro-expressions faciales pour différencier ces deux états. La macro expression faciale pour une expression posée et la micro-expression faciale pour un état spontané et/ou si l'on réprime ses émotions. Contrairement aux EF délibérées qui exagèrent l'expression, les EF spontanées sont produites d'une manière involontaire ou volontairement faussée conduisant à des effets plus subtils tout en reflétant le vrai comportement facial. Il faut donc prendre en considération ces différences, c'est ce que nous ferons en analysant la pose « casting » et la résolution du « puzzle » dans la recherche d'une émotion spontanée. Par ailleurs, le modèle d'entraînement doit limiter les états de pose du sujet, car il existe de nombreuses preuves (Ekman, Hess et Kleck [42], Sauter et Fischer [43]) que les personnes font des choses différentes avec leurs visages entre une expression posée et une expression spontanée. Cependant, pour la vidéo, la plupart des bases de données utilisées pour l'apprentissage sont réalisées dans des laboratoires de façon contrôlée, par des acteurs, le côté spontané est donc peu représenté, contrairement aux

images que nous offrent de nombreux dataset sur l'expression spontanée, mais sans mouvement. Ce sont des facteurs que nous devons prendre en compte dans l'analyse de nos résultats et nos choix des dataset. Enfin, pour le mouvement, comme l'illustre la figure 17, les émotions sont composées généralement de quatre parties, la phase sans contraction musculaire (Neutre), une contraction musculaire qui monte en intensité (Onset), un plateau de contraction musculaire stable (Apex), est un relâchement des muscles pour revenir en position neutre (Offset). C'est l'analyse de l'ensemble de ces facteurs qui permettent une meilleure reconnaissance de l'expression.

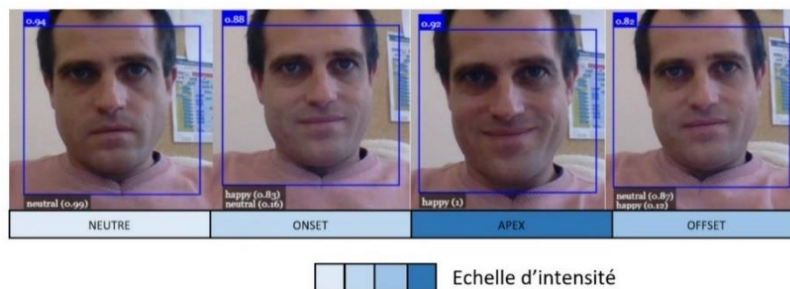


Figure 17 – Les 4 étapes d'analyse pour une expression en mouvement.

1.5.4 - Facteurs d'influence de l'expression faciale

Les images faciales peuvent également être affectées par différentes variations. Des facteurs intrinsèques, comme les occlusions (barbe, lunettes...), des changements d'apparence dans la forme du visage, la texture, la couleur selon l'origine ethnique, l'âge ou le sexe. Par ailleurs, une même expression peut fondamentalement être exprimée différemment entre plusieurs personnes (intensité différente, problème musculaire...). Il existe aussi des facteurs extrinsèques à prendre en considération comme l'alignement du visage, l'éclairage et la qualité de l'image. Des études récentes montrent que les approches d'apprentissage en profondeur de type deep-learning, peuvent atteindre des performances impressionnantes sur ces deux facteurs. C'est pourquoi nous nous sommes orientés sur une architecture en cascade avec trois étapes de réseaux convolutifs profonds conçus pour prédire l'emplacement des points de repère qui constituent les caractéristiques de l'émotion. Cela diverge de ce que nous avons vu et appliqué sur la détection du visage avec la méthode Viola et Jones. La principale contrainte étant les trop grandes variations visuelles du visage humain même avec des fonctionnalités et des classificateurs plus avancés. Nous décrivons dans le prochain chapitre l'approche MTCNN de Zhang et al [44] (Multi-Task Cascaded Convolutional Neural Networks) pour reconnaître les expressions faciales et comment traiter ces différents facteurs.

1.6 - Deep-learning, Bibliothèques et Dataset

Dans ce chapitre nous allons décrire l'architecture CNN et déterminer comment obtenir la reconnaissance faciale des émotions. Nous nous baserons principalement sur les travaux de Zhang et al (2016) et du modèle MTCNN l'un des plus utilisés pour sa grande précision et ses performances de calculs rapides, mais aussi, proposés dans les frameworks de deep-learning que nous utiliserons et présenterons comme Keras et Tensorflow. Enfin nous clorons ce chapitre sur les dataset disponibles pour entraîner notre modèle.

1.6.1 - Architecture CNN

Avec la méthode Viola et Jones nous avons vu le principe des classifieurs qui, associés aux caractéristiques pseudo-Haar permettent d'obtenir de très bons résultats. Mais pour une expression, comme nous l'avons énoncé, le nombre de variations est beaucoup plus important. En 2012, une révolution se produit, AlexNet, un nouvel algorithme de deep-learning à base de neurones convolutifs (CNN) explose les records. Les réseaux de neurones convolutifs ont une méthodologie similaire à celle des méthodes traditionnelles d'apprentissage supervisé. Ils détectent les caractéristiques, puis entraînent un classifieur dessus, à la différence près que les caractéristiques sont apprises automatiquement. Un CNN applique généralement 5 types de couches différentes à une image afin d'en extraire les informations pertinentes, la couche de convolution, la couche de pooling, la couche de correction ReLU et la couche fully-connected (figure 18).

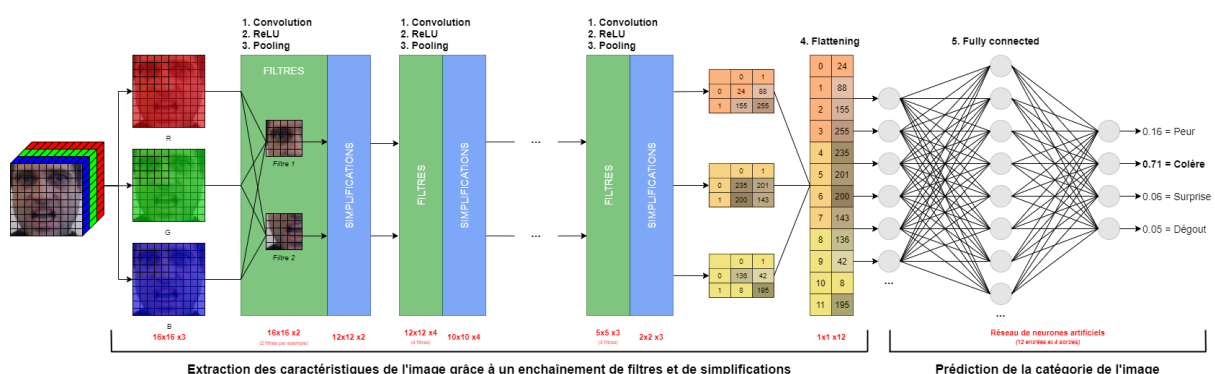


Figure 18 – Principe d'une architecture CNN

La couche de convolution permet de trouver des caractéristiques, elle est donc la composante la plus importante. Elle est assez proche de la mécanique que nous avons vue sur les

caractéristiques pseudo-Haar. Le principe est le même, faire "glisser" une fenêtre représentant un filtre sur l'image. La différence ici, c'est que l'on cherche une convolution calculée et non une corrélation croisée (comme la recherche d'une caractéristique pseudo-Haar). Le filtre sera la caractéristique à étudier, cette caractéristique est trouvée par convolution. Pour comprendre ce principe, nous simplifions l'image du visage par une croix noire et blanche (figure 20). La caractéristique (feature) dans cet exemple prend la taille de 3x3 puis recherche des points communs par balayage de l'image. Dans la démonstration nous avons pris la valeur d'un pixel en 2D, 1 pour blanc et -1 pour noir. Entre deux caractéristiques comparées, nous réalisons un calcul d'addition entre les 9 pixels, ce qui nous permet d'obtenir un score de similarité, ici 80%. Ce même calcul est réalisé sur l'ensemble de l'image avec un stride de 1 (la zone comparée bouge d'un seul pixel entre chaque comparaison).

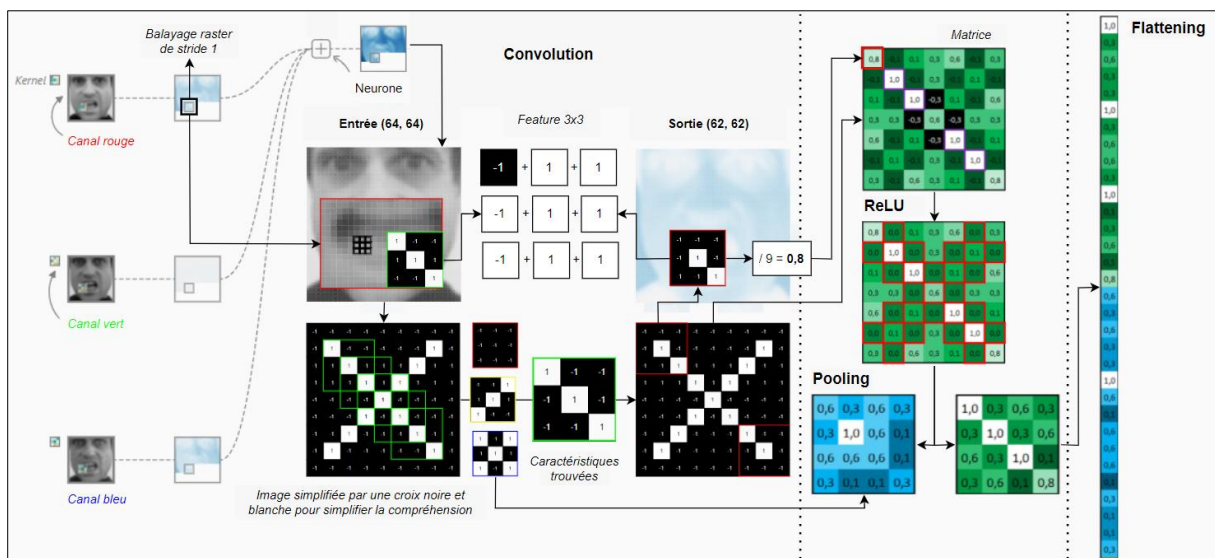


Figure 19 – Principe de convolution et de l'activation ReLU, ici la comparaison entre les deux images permettent de trouver trois caractéristiques, l'une des caractéristiques (en vert) est nettoyée ensuite des valeurs négatives (ReLU)

Cela permet de générer une matrice, avec l'ensemble des valeurs trouvées. Ceci est réalisé pour l'ensemble des caractéristiques trouvées, ce qui génère autant de matrices. Ensuite est appliquée la fonction d'activation de type ReLU (unité de rectification linéaire) qui permet de « nettoyer » les valeurs négatives, qui prennent la valeur zéro ceci pour améliorer l'efficacité du traitement entre chaque couche de convolution. Sur cette matrice « nettoyée » on applique ensuite le max-pooling qui consiste à balayer la matrice (taille de 2x2, strip 1) et de prendre la valeur maximale de chaque « morceau ». Enfin, on termine par le flattening (mise à plat) qui consiste tout simplement à prendre la totalité des valeurs de nos matrices

précédemment calculées, et à les empiler, en vue de les exploiter dans la couche d'entrée d'un réseau de neurones, ce qu'on appelle le « fully connected ». Le nombre de neurones que composera la couche de sortie déterminera le nombre de classifications. Par exemple, avec les 6 émotions de bases et le neutre, il nous faudra 7 neurones de sorties pour déterminer la probabilité de l'image (figure 20).

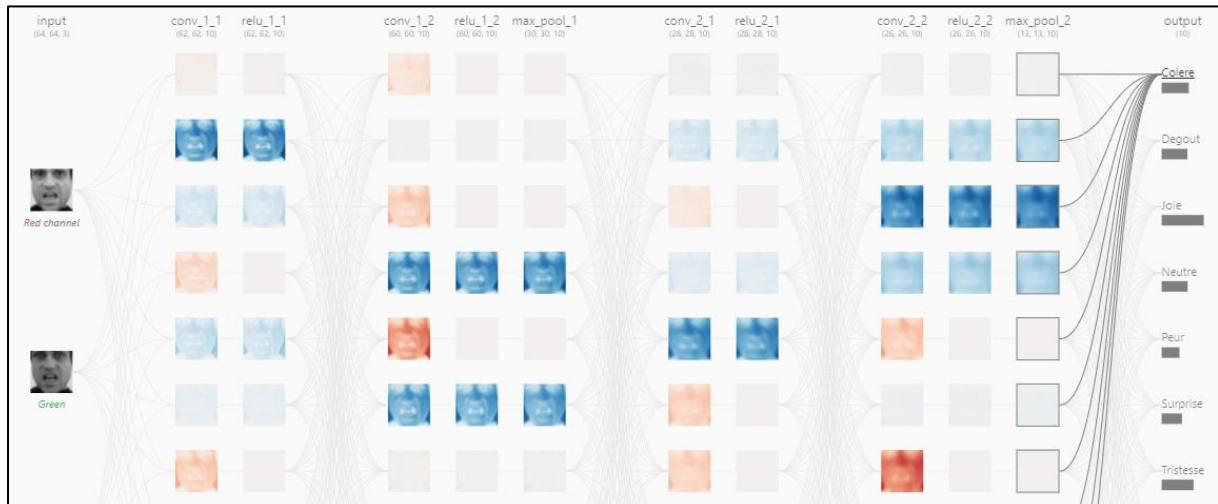


Figure 20 – Réseau CNN avec trois couches de convolution et 7 neurones de sorties obtenus avec un test local sur CNN
Explainer [45] (dataset non représentatif)

Pour parvenir à cette quantité de calcul, nous sommes aidés de bibliothèques comme Tensorflow [46] utilisé pour l'apprentissage automatique, mais aussi des dataset permettant l'entraînement des modèles. Ce sont les deux techniques importantes que nous verrons dans les prochaines sections.

1.6.2 - Les bibliothèques de deep-learning

Avant réservé aux initiés, il faudra attendre 2011 avec l'arrivée de DistBelief, première bibliothèque de deep-learning, pour avoir des outils qui facilitent la construction d'architectures d'apprentissage en profondeur. C'est en 2017 que Google propose Tensorflow, en open source, basé sur l'architecture de DistBelief. Le concept de tenseur (Tensor) est fortement associé aux mathématiques appliquées dans l'ingénierie et la physique. Tensorflow propose des API aux utilisateurs de niveaux expert et débutant pour développer des applications destinées au cloud, le mobile, le Web et le bureau. Pour les débutants, Tensorflow recommande l'API Keras pour développer et entraîner les modèles d'apprentissage en profondeur. L'architecture de Tensorflow est divisée en trois parties

fonctionnelles, à savoir le processeur de données, le générateur de modèles, la formation et l'estimation du modèle. Il accepte les entrées en tant que tenseurs ou tableau multidimensionnel, construit un organigramme des opérations qui explique les multiples opérations soumises de l'entrée à la sortie. D'où le nom Tensorflow, parce que les tenseurs parcourent une liste d'opérations et produisent la sortie souhaitée de l'autre côté. Pour visualiser le réseau de neurones construit par Tensorflow, nous utilisons le kit de visualisation Tensorboard. Les algorithmes pris en charge par Tensorflow sont la classification, la régression linéaire, la classification d'apprentissage en profondeur, la classification d'arborescence boostée et la régression d'arborescence boostée. De nos jours, il existe, de nombreux frameworks qui facilitent la construction d'architectures d'apprentissage en profondeur. Leur interopérabilité devient un problème majeur, c'est la raison pour laquelle nous nous sommes orientés uniquement sur des bibliothèques open source. Tensorflow, mais aussi Keras [47] qui permettent d'interagir avec les algorithmes de réseaux de neurones profonds. La bibliothèque Pandas [48] qui permet la manipulation et l'analyse des données ou encore Numpy [49] destinée à manipuler des matrices ou tableaux multidimensionnels. On a aussi privilégié les bibliothèques compatibles avec CUDA [47]. Cette technologie développée par NVIDIA permet d'utiliser le processeur graphique (GPU) pour le calcul des modèles. On notera aussi l'utilisation d'OpenCV, qui utilise les bibliothèques citées et particulièrement adaptées pour l'application web DataTrainX dans les traitements d'images interactifs.

1.6.3 - Les dataset sur les émotions

L'enjeu d'un modèle CNN réside dans la capacité d'apprendre sur un jeu de données (dataset). Plus elles sont nombreuses, plus la capacité du modèle sera entraînée et performante. Après extraction des caractéristiques d'un point de vue algorithmique, il faut comparer les résultats avec des données tests. Dans ce cadre, il faut s'orienter sur des données calibrées et préalablement définies pour cet objectif. Il existe dans l'open data, des bases de données axées sur les émotions pour justement évaluer les algorithmes de manière comparative et qui sont, par ailleurs, disponibles publiquement et librement. Ces ensembles de données sont appliqués à la recherche en apprentissage automatique et ont été cités dans des revues universitaires. Les jeux de données font partie intégrante du domaine de l'apprentissage automatique. Des avancées majeures dans ce domaine peuvent résulter des progrès des

algorithmes d'apprentissage. Nous utiliserons des modèles étiquetés, mais aussi non étiquetés. La plupart des données sont soumises à autorisation et proviennent de centres de recherches universitaires. Voici une liste des dataset que nous avons sollicitée et que nous prévoyons d'utiliser. Certains malheureusement ne nous ont pas été accordés du fait que ce mémoire ne se réalise pas dans le cadre de la recherche. Néanmoins nous disposons d'assez d'éléments pour entraîner nos modèles (tableau 1) :

Tableau 1 – Sources de dataset recherchées

NOM	DESCRIPTION	AUTEUR	AUTORISER
AFFECTNET [48]	420 299 images classées par expression	Dr. Mahoor	NON
DISFA [48]	24 vidéos expressions non posées	Dr. Mahoor	NON
FER2013 [49]	28 709 images grayscale par expression	PL Carrier	OUI
GEMEP [50]	145 vidéo expressions posées français	Université de Genève	NON
CK ET CK+ [51]	593 vidéo expressions posées (FACS)	Patrick Lucey et al.	NON
MMI [52]	2900 vidéos expressions posées (FACS)	Maja Pantic et al.	NON
RAVDESS [8]	2880 vidéos expressions posées	S.R. Livingstone F.A. Russo	LIBRE ACCES
MUG [53]	80 facial landmark (FACS)	University of Thessaloniki	OUI
JAFFE [53]	213 images expressions, femme asiatique	Michael J. Lyons	OUI
WIDER FACE [9]	32 203 images et 393 703 labels	University of Hong Kong	LIBRE ACCES
YALE FACE [54]	165 images GIF expression grayscale	UC San Diego	OUI

Dans ce chapitre nous avons préparé et orienté la réflexion du mode opératoire. Cela nous a permis de définir le périmètre général du projet et de réaliser un inventaire des ressources à employer. Nous avons réalisé les recherches adéquates de faisabilité, en testant différentes approches et en précisant les impacts et besoins d'un logiciel de REF. Cela clôt ce premier chapitre. Nous allons maintenant nous orienter sur la conception et le développement de notre application.

Chapitre 2 - La conception DataTrainX

Nous avons décrit les grandes lignes de notre projet, nous verrons dans ce chapitre comment nous organiser et les scénarios informatiques que nous avons envisagés. Dans ce chapitre nous décrivons la phase de conception et de construction du projet.

2.1 - Mise en œuvre du projet

Les grandes orientations méthodologiques étant définies, nous nous sommes orientés sur une étude de cas et les scénarios envisagés afin de débiter une réflexion sur le mode opératoire technique comme convenu en phase d'analyse.

2.1.1 - Étude de cas et scénarios

L'étude préliminaire comprend une analyse des cas d'utilisation (ci-dessous, figure 21). Cette étude de cas a pour but de fournir le déroulé de la cinématique proposée au MOA (la direction d'AMIO) mais aussi une vue informatique de notre système.

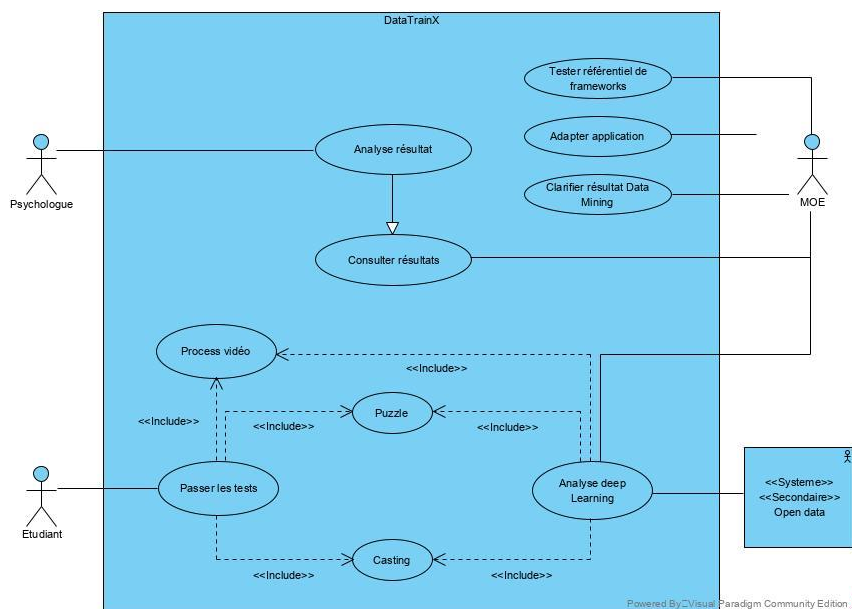


Figure 21 – Diagramme des cas d'utilisation de l'étude préliminaire

Nous sommes à ce stade dans la phase de conception, nous avons souhaité dans le déroulé de ce mémoire présenter tout d'abord un niveau d'abstraction élevée suffisamment

généraliste, pour couvrir en un seul axe de structuration, la partie concernée par le projet DataTrainX. Cela pour faciliter la compréhension du développement que nous réaliserons sur la REF dans les prochaines sections. À l'issue de cette activité, l'analyse des cas d'utilisation est produite, mais non encore consolidée ni validée définitivement. Trois intervenants interviennent dans ce processus, l'apprenant, sujet d'expérience, qui réalisera le test, la psychologue qui validera la pertinence des résultats et le MOE, chef de projet technique, mais également lead développeur de la solution DataTrainX. Cette étude de cas nous permet de poursuivre l'analyse dans la description de scénarii sur deux acteurs principaux l'apprenant et la psychologue : réaliser le casting, passer les tests, résoudre le puzzle et analyser les résultats. Nous clorons notre représentation UML par un diagramme de séquence concernant « Passer les tests ».

2.1.2 - Description du DCU « Réaliser le casting »

Ce CU permet de caler l'étiquetage du sujet avec la prédiction de l'algorithme de reconnaissance faciale des émotions et de déterminer si ce dernier obtient de bons résultats. Il est demandé au sujet d'exprimer avec son et visage, les six émotions de base, sans lui donner de contexte ni de feed-back. L'utilisateur doit préalablement se connecter au système pour réaliser un test. Une interface lui permet l'attribution d'un identifiant unique (token) ou de réaliser une nouvelle session avec le même token.

CU : Réaliser le casting
Acteur : Apprenant
Pré conditions : Nouvel apprenant ou apprenant ayant déjà réalisé un test, l'utilisateur est authentifié, la webcam fonctionne
Post conditions : Émotions reconnues de manières interactives avec OpenCV, les taux de réussites/échecs ainsi qu'une capture d'image sont enregistrés pour chaque émotion.
Scénario nominal
DESCRIPTION DU SCÉNARIO NOMINAL « DEBUT » 01 : Le système propose un nouveau token 02 : Le système invite l'utilisateur à entrer son token 03 : L'utilisateur saisit son token

04 : Le système vérifie si token déjà utilisé ou nouveau token valide, voir scénario alternatif SA1

05 : Le système informe l'objet de l'étude et de la mise en condition

06 : Le système démarre le premier test

07 : Le système invite l'utilisateur à exprimer une émotion type

08 : Un algorithme en temps interactif vérifie si l'émotion attendue est détectée

09 : Un taux de réussite est déterminé sur l'émotion reconnue, voir scénario alternatif SA3 et scénario d'anomalie SAN1

10 : Le taux de réussite ainsi qu'une capture image sont enregistrés

11 : Le système vérifie si le taux de réussite est valable, voir scénario alternatif SA2 et scénario d'exception SE1

12 : Le système passe à l'interface suivante

« FIN »

Scénario alternatif de validité

DESCRIPTION DU SCÉNARIO ALTERNATIF DE VALIDITÉ

Scénario alternatif SA1 : Le token est incorrect, ce n'est ni le nouveau ni un ancien token utilisé.

SA1 commence au point 04.

Le système informe l'utilisateur que les données saisies sont erronées.

Le scénario reprend au point 01 du scénario nominal.

Scénario alternatif SA2 : Les six émotions d'Ekman doivent être testées par l'utilisateur.

SA2 commence au point 11 du scénario nominal et passe à l'émotion suivante.

Le scénario reprend au point 07 du scénario nominal avec une nouvelle émotion.

Scénario alternatif SA3 : L'émotion attendue n'est pas reconnue par la REF pour la première ou la deuxième fois.

SA3 commence au point 09 du scénario nominal.

Le système indique que l'émotion attendue n'est pas reconnue.

Le système enregistre l'échec, voire le scénario d'anomalie SAN1.

Le scénario reprend au point 07 du scénario nominal avec la même émotion.

Scénario d'anomalie

Scénario d'anomalie SAN1 : L'émotion attendue n'est pas reconnue par la REF pour la troisième fois.

SAN1 commence au point 09 du scénario nominal.

Le système indique à l'utilisateur qu'il ne reconnaît pas l'émotion de l'utilisateur.

SAN1 continue au point 10 du scénario nominal et comptabilise le résultat en anomalie.

Scénario d'exception (expression volontairement faussée)

Scénario d'exception SE1 : 50% des émotions attendues sont comptabilisées en anomalies.

SE1 commence au point 11 du scénario nominal.

Le système indique à l'utilisateur que les résultats attendus ne sont pas compatibles avec le test.

SE1 met fin à la session.

2.1.3 - Description du DCU « Passer les tests »

Ce CU permet à un utilisateur de compléter un questionnaire déterminant un profil d'apprenant. Il est composé de 80 questions avec deux choix de réponses possibles A ou B. L'apprenant lie la question, il est filmé et le son enregistré. Pendant cette phase, il peut être envisagé de réaliser une analyse de REF (impact sur le scénario d'exception).

CU : Passer les tests
Acteur : Apprenant
Pré conditions : L'utilisateur est authentifié, la webcam fonctionne et l'utilisateur a réussi le casting.
Post conditions : Les données de saisie du questionnaire ainsi que le flux vidéo sont enregistrés.
Scénario nominal
DESCRIPTION DU SCÉNARIO NOMINAL « DEBUT » 01 : Le système invite l'utilisateur à répondre à une question. 02 : L'utilisateur choisit la réponse qui lui semble la plus adaptée, A ou B. 03 : le système vérifie que l'une des réponses est cochée. 04 : Le système enregistre la réponse et la séquence vidéo. 05 : Le système vérifie si question complémentaire, voir scénario alternatif SA1.

06 : Le système passe à l'interface suivante « FIN »
Scénario alternatif
DESCRIPTION DU SCÉNARIO ALTERNATIF Scénario alternatif SA1 : Il reste des questions à saisir par l'utilisateur. SA1 commence au point 05 du scénario nominal. Le scénario reprend au point 01 du scénario nominal avec une nouvelle question.

2.1.4 - Description du DCU « Résoudre puzzle »

Ce CU propose une interface de résolution d'un puzzle dans un temps limité. Le but est de mettre l'utilisateur en situation d'une résolution d'une problématique et d'y observer des émotions afin de déterminer une corrélation entre le test de Kolb et le profil émotionnel du sujet. Il est convenu que le choix du puzzle sera purement subjectif, en dehors d'un cadre scientifique, qui serait de l'ordre de la recherche en psychologie.

CU : Résoudre le puzzle
Acteur : Apprenant
Pré conditions : L'utilisateur est authentifié, la webcam fonctionne et il a passé le questionnaire de Kolb.
Post conditions : Le score de résolution du puzzle, la durée et le flux vidéo sont enregistrés.
Scénario nominal
DESCRIPTION DU SCÉNARIO NOMINAL « DEBUT » 01 : Le système propose une résolution d'un puzzle et affiche un compteur. 02 : L'utilisateur peut mettre fin à cette épreuve, soit en attendant la fin du compteur, soit en ayant résolu le puzzle. 03 : Le système vérifie la validité des informations saisies. 04 : Le système enregistre ces informations dans la base de données. 05 : Le système notifie l'utilisateur du bon déroulement de l'opération. 06 : L'épreuve est terminée, des résultats temporaires peuvent être proposés à l'apprenant (pré analyse, résultat de Kolb). « FIN »

2.1.5 - Description du DCU « Analyse de résultats »

Ce CU permet au psychologue de valider les résultats proposés par la REF.

CU : Analyse résultats
Acteur : Psychologue
Pré conditions : Le psychologue est authentifié, des résultats sont présents.
Post conditions : Les données de validation sont enregistrées.
Scénario nominal
DESCRIPTION DU SCÉNARIO NOMINAL « DEBUT » 01 : Le système propose à l'utilisateur une liste de résultats en attente de validation. 02 : L'utilisateur choisit le résultat à valider. 03 : Le système propose une interface de vérification du résultat. 04 : L'utilisateur vérifie la capture vidéo et la cohérence avec le résultat de la REF, voire le scénario d'anomalie SAN1. 05 : L'utilisateur valide le résultat. 06 : Le système enregistre la réponse. 07 : Le système vérifie les paramètres, voir le scénario alternatif SA1. « FIN »
Scénario alternatif
Scénario alternatif SA1 : L'émotion reconnue par l'IA est en accord avec ce que constate l'utilisateur, elle est validée, mais elle est de type « neutre ». SA1 commence au point 07 du scénario nominal. Le système vérifie si la majorité des participants ont aussi une émotion neutre sur cette question, dans le cas contraire l'émotion génère une anomalie au point 07 du scénario d'anomalie SAN1.
Scénario d'anomalie
Scénario d'anomalie SAN1 : L'émotion reconnue par l'IA est différente de ce que constate l'utilisateur. SAN1 commence au point 04 du scénario nominal. 05 : L'utilisateur invalide le résultat de l'émotion retournée par l'IA.

07 : Une exception est enregistrée par le système pour cette question, pour test complémentaire ultérieur (chercher à améliorer l’algorithme ou définir ce résultat comme invalide ou remise en question de la question elle-même « non significative pour l’émotion »).

2.1.6 - Diagramme de séquence « passer les tests »

Nous terminons cette partie d’analyse par la présentation d’un diagramme de séquence (figure 22), le but est de décrire, dans un ordre chronologique, comment se déroulent les interactions entre les objets. C’est à cette étape que nous avons affiné et validé les modèles organisationnels dans le traitement, les flux et les données.

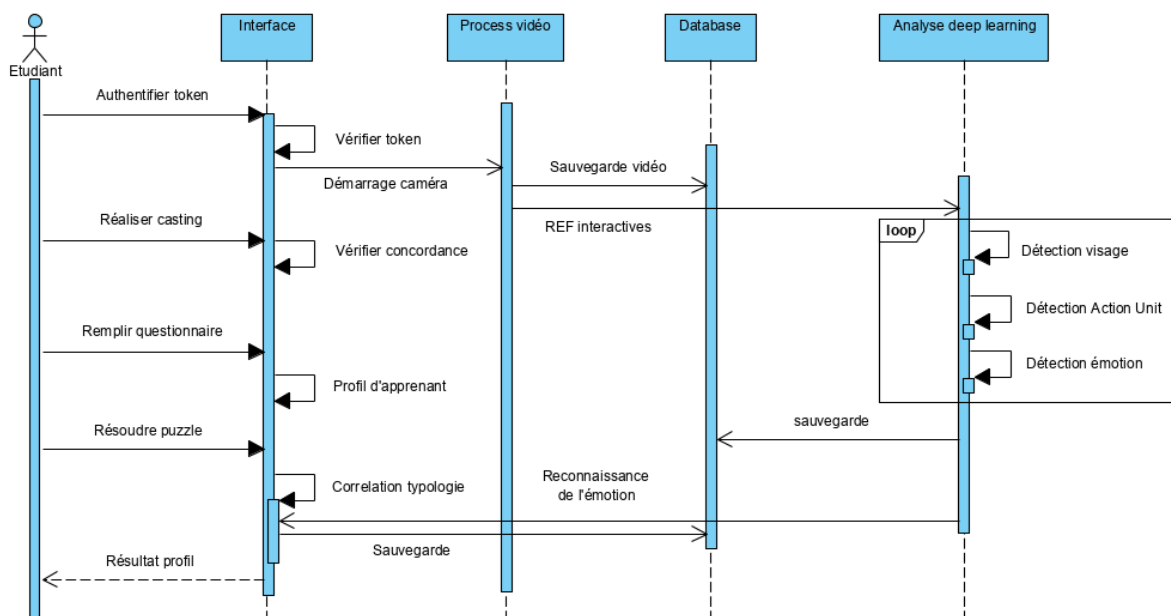


Figure 22 – Diagramme de séquence "passer les tests" dans le cas nominal

Ces différents diagrammes UML démontrent le côté interactif de l’application qui est relativement présent à tous les niveaux. Que ce soit dans le processus vidéo, l’analyse des expressions en mode interactif ou encore dans la restitution des résultats. Cela devra être pris en considération dans le choix de notre architecture informatique, c’est ce que nous développerons dans le prochain chapitre.

2.2 - Architecture informatique

Ce chapitre fait suite à l'ensemble des décisions stratégiques prises durant la conception. Nous sommes au stade de définir l'organisation des différents éléments et de leurs relations. Cela comprend quatre types d'architectures que nous détaillerons. L'architecture matérielle, en section 2.2.1, qui nous orientera sur la recherche de performances tout en respectant les contraintes de coût. L'architecture des informations, en section 2.2.2, qui concerne la manière dont les informations sont organisées et agrégées. L'architecture logicielle, en section 2.2.3, qui définira nos différents modules et l'organisation de la programmation. Nous terminerons ce chapitre par l'architecture technique, en section 2.2.4, tournée sur l'organisation logique du projet DataTrainX, cela inclut le matériel informatique, les logiciels systèmes, les middlewares, ainsi que les réseaux et les relations entre ces différents éléments.

2.2.1 - Architecture matérielle

Développer un projet de deep-learning demande une architecture matérielle spécifique. Le principe d'un apprentissage automatique par convolution génère un processus multitâches qui demande au matériel de partager des ressources entre des processus concurrents et indépendants. Le parallélisme est une solution pour exécuter un programme dont les besoins vont au-delà des capacités d'un seul système. Plusieurs stratégies sont possibles, augmenter la vitesse du processeur, les performances du système, adapter le jeu d'instructions, déléguer certaines tâches à des périphériques spécialisés ou répartir les tâches sur plusieurs systèmes équivalents. Ce sont des stratégies utilisées en deep-learning qui impactent des décisions sur le choix de la carte graphique (GPU), la puissance du processeur (CPU) ou l'architecture distribuée (Big-Data). Ce dernier point concerne le fait que lorsque la donnée est trop volumineuse pour être stockée sur un seul nœud ou nécessite beaucoup de mémoire vive pour s'exécuter, il est avantageux de déplacer le programme vers la donnée, plutôt que de transférer la donnée vers le programme. Par exemple, en s'orientant sur des bases de données NoSQL plus performantes dans l'analyse de masse, le traitement massivement parallèle comme Hadoop [55], ou encore le stockage des données en mémoire (Memtables). En février 2019, sous l'impulsion de John L. Hennessy and David A [56], on voit émerger des architectures matérielles spécialisées en fonction des calculs à réaliser. C'est de là qu'interviennent des

cartes graphiques spécifiques pour réaliser des calculs développés par AMD, Intel, NVidia ou encore la TPU de Google. C'est la raison pour laquelle, nous nous sommes orientés sur une carte graphique compatible avec CUDA et la bibliothèque cuDNN [57] de NVidia qui permet de réaliser les calculs sur le processeur graphique et non sur le processeur central, permettant un gain de temps considérable. Du côté des architectures plus traditionnelles, des progrès très importants ont été réalisés, mais l'impact sur la conception des logiciels n'est pas encore mesurable sur chacun d'entre eux. Nous avons pris en considération ces points, notamment on s'orientant sur des bibliothèques comme MKL et AVX préconisées par Intel [58] concernant le processeur. Pour l'utilisation des bibliothèques, que nous avons exploitées comme Keras ou Tensorflow, il faut aussi des pilotes permettant de brancher des matériels hétérogènes, raison pour laquelle nous n'exploitons pas l'ensemble des dernières versions. Côté langage de programmation, nous avons utilisé Javascript avec TensorFlow.js qui s'est avéré plus rapide en y ajoutant le multithreading via XNNPACK [59]. Pour python, étonnamment, compte tenu de son adoption, il n'est pas nativement un langage multithread. Voici un récapitulatif de notre architecture matérielle que nous avons utilisée pour DataTrainX. Nous notons seulement ici les bibliothèques les plus significatives :

Tableau 2 – Bibliothèques logicielles utilisées pour le CNN

ARCHITECTURE	DRIVER VERSION*	BIBLIOTHEQUES VERSION (CNN)*
INTEL ICORE I7-4790 1CPU 4CORE 8THREADS 3.6 GHZ	Intel MKL 2020 (2021) oneDNN 2.4.4	Python 3.6.13 (3.10) Keras 2.3.1 (2.7) Tensorflow 2.1 (2.7)
NVIDIA GPU RTX 2060 RAM 6GO	GPU drivers 472.75 CUDA Toolkit 10.1 (11.3) cuDNN 7.6.5 (8.2.1)	Tensorflow-gpu 2.1 (2.7) Numpy 1.19 (1.21) h5py 2.10 (3.6)
OS WINDOWS 10 - X86-64	Vs2015_runtime XNNPACK	Pytorch 1.10.2
COMPILATION TFJS / NODE : TENSORFLOW.JS 3.12, TENSORFLOW 2.7, KERAS 2.7		
DIVERS, NON SIGNIFICATIF - SSD 180 GO RAM 32 GB - ASUS H97-PLUS		
*() VERSION PLUS RECENTE		

À titre de comparaison, cette architecture nous permet la génération d'un modèle comportant l'analyse de 35800 images (FER2013) en seulement 40 minutes au lieu de 13 heures sans l'activité GPU (cuDNN), 16 heures sans MKL [60]. En conclusion, nous avons un budget modeste, mais suffisant pour l'objectif fixé. Si nous avions eu d'autres moyens, l'effort se serait porté sur un processeur comportant plus de CPU comme le i7-7700k cadencé à 4,2 GZ qui nous aurait permis un gain de temps de calcul estimé à 20%.

2.2.2 - Architecture des informations

DataTrainX est une application web dont l'objectif en deep-learning de la REF se fait en deux temps. Nous souhaitons une analyse interactive basée sur un modèle préentraîné. Le modèle peut lui-même s'enrichir avec les données collectées des utilisateurs afin d'améliorer son score de prédiction. Cela suppose une analyse approfondie sur différentes phases avec comparaison des résultats selon les techniques utilisées, ce qui induit une architecture spécifique que nous développons à partir du schéma ci-dessous (figure 23).

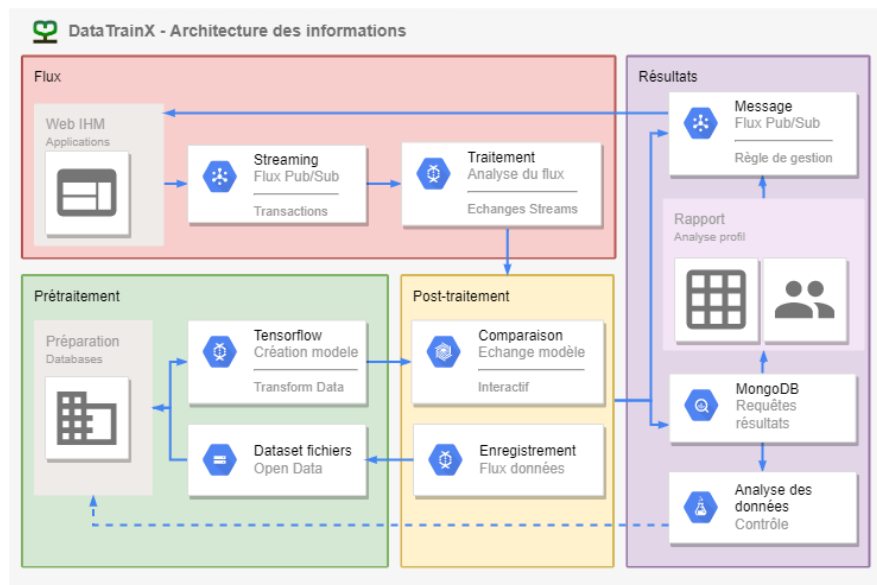


Figure 23 – Architecture des flux d'informations DataTrainX

La phase de Flux est dédiée à l'expérience utilisateur. Ici nous mettons l'accent sur l'ergonomie et la compréhensibilité de l'application. Elle met en scène la cartographie de l'application afin de permettre à l'utilisateur de comprendre la finalité et le fonctionnement de DataTrainX. En dehors des objectifs propres de l'application décrits plus haut, il est aussi prévu d'expliquer de manière interactive le principe d'un réseau convolutif CNN. La phase prétraitement correspond à la construction du modèle. Elle regroupe les fichiers Dataset et les différentes bibliothèques qui seront utilisés pour sa construction. Il est prévu que ces dataset s'enrichissent lors de la phase de flux. La phase post-traitement permettra de manière interactive de restituer les résultats liés entre la phase de prétraitement et la phase de flux. Elle est considérée comme une phase tampon entre les différentes couches d'informations. La dernière phase dans le traitement des informations est appelée résultats. C'est ici que nous

restituerons l'ensemble des données pour afficher le profil de l'apprenant. C'est dans cette phase aussi que nous ferons les contrôles adéquats en cas d'erreurs sur la REF.

2.2.3 - Architecture logicielle

Pour développer DataTrainX, nous avons besoin d'utiliser différents langages informatiques pour des raisons de stratégies techniques différentes. Premièrement, concernant l'appli centré utilisateur, nous avons besoin d'interaction avec le flux vidéo. Le flux de données semble donc prédominant. Deuxièmement, concernant la génération du modèle, les composants se distribuent sur une seule ligne et ne font que passer l'information transformée à leurs voisins, ce que nous avons décrit en prétraitement dans l'architecture d'information. Dernier point, pour des raisons d'organisation, d'efficacité, mais surtout de par une utilisation pléthorique de nombreuses bibliothèques, nous nous spécialisons sur l'approche métier. La contrainte d'utiliser de nombreuses bibliothèques dépendantes les unes des autres, pousse à nous orienter sur une architecture en couches qui semble être la conséquence inéluctable d'une telle approche. En effet, les nouveaux composants utilisent les anciens et ainsi de suite. La bibliothèque tend donc à devenir une sorte d'empilement de composants. Pour répondre à ces problématiques, nous avons envisagé une solution hybride et centralisée autour d'une architecture NodeJS. C'est un environnement d'exécution permettant d'utiliser le JavaScript côté serveur, il répond à l'ensemble des points grâce à son fonctionnement non bloquant (asynchrone). Il permet de concevoir des applications en réseau performantes et interactives, telles qu'un serveur web, ce qui répond à notre premier besoin. NodeJS dispose d'une API de bas niveau basée sur la structure entrées-sorties, qui permet d'adopter une approche de programmation événementielle, ce qui répond à notre deuxième besoin. Enfin, la bibliothèque Tensorflow propose une version pour NodeJS, permettant de faire les passerelles avec les autres bibliothèques basées sur python. Par ailleurs, nous avons pris en considération une organisation de la structure de notre projet NodeJS afin d'éviter la duplication du code et d'améliorer la stabilité de nos services. Nous nous sommes orientés sur une architecture 3 tiers en séparant la couche de données, la couche applicative et la couche d'interface utilisateur. Le but est de découpler la logique applicative des opérations de base de données afin de faire évoluer notre application et notre base de données séparément. Différentes technologies et langages seront utilisés dans ces couches. Le HTML5, JavaScript et CSS dans la

couche interface utilisateur, Python et NodeJS pour la couche applicative et MongoDB accompagné des ressources fichiers vidéos/images pour la couche de données. La plupart des échanges se feront en PUB/SUB via du JSON et une API REST ce qui nous permet de découpler l'interface utilisateur du stockage des données. En effet, cela renforce la séparation et permet aux composants d'évoluer indépendamment. Dernier point, la couche applicative NodeJS est sur une architecture traditionnelle de type MVC. Cela donne une architecture complexe et hybride que nous représentons graphiquement dans le schéma ci-dessous (figure 24). L'API cliente présente dans ce schéma ne sera pas développée, mais elle est prise en considération dans une logique d'évolution de l'application.

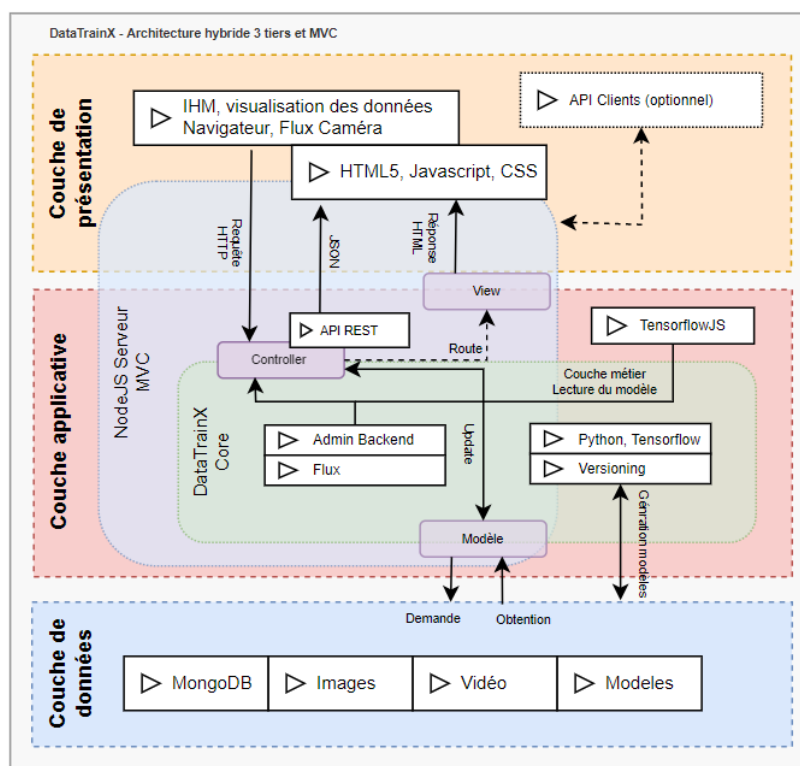


Figure 24 – Architecture hybride 3 tiers avec design MVC sur une partie de l'applicatif

2.2.4 - Architecture technique

Notre application est développée sur le serveur qui l'héberge. C'est la machine la plus puissante dont nous disposons et par ailleurs, c'est un serveur « maison » branché sur le réseau Internet uniquement dédié au besoin du prototype DataTrainX. L'architecture technique réseau est donc simplifiée et a demandé peu de configurations, le serveur étant hébergé à mon propre domicile. L'IP fixe est obtenue par Free sur un forfait de type familial,

le routeur « Freebox » est configuré pour rediriger les requêtes HTTP sur l'IP de destination fixe du serveur 192.168.0.222. Le routeur est configuré avec un bail DHCP permanent vers cette IP sur l'adresse MAC du serveur. Il est redirigé sur le port 443 pour un accès SSH en protocole TCP. L'IP est 82.64.152.102, que nous avons rattachée au nom de domaine datatrainx.akairnet.fr. Le nom de domaine akairnet.fr est hébergé chez le prestataire OVH c'est un domaine personnel que nous utilisons pour des besoins divers. Le sous-domaine datatrainx est géré au niveau des DNS de type A. Le serveur est un PC conventionnel que nous utilisons comme tel, sur un système avec licence d'exploitation Windows 10 Pro. Il n'a pas une configuration type serveur en dehors de l'applicatif et n'est pas destiné à être en ligne, en continu. Deux applicatifs sont lancés, le serveur web principal (frontend) sur le port 443 et une API REST (backend) sur le port 4000. Cette dernière gère les transactions avec la base de données locale NOSQL MongoDB. Les transactions entre ces deux applicatifs gèrent un certificat SSL officialisé sur « Let's Encrypt » [61] assurant ainsi le cryptage des données qui transitent. Nous représentons l'ensemble de cette architecture dans la figure 25 ci-dessous.

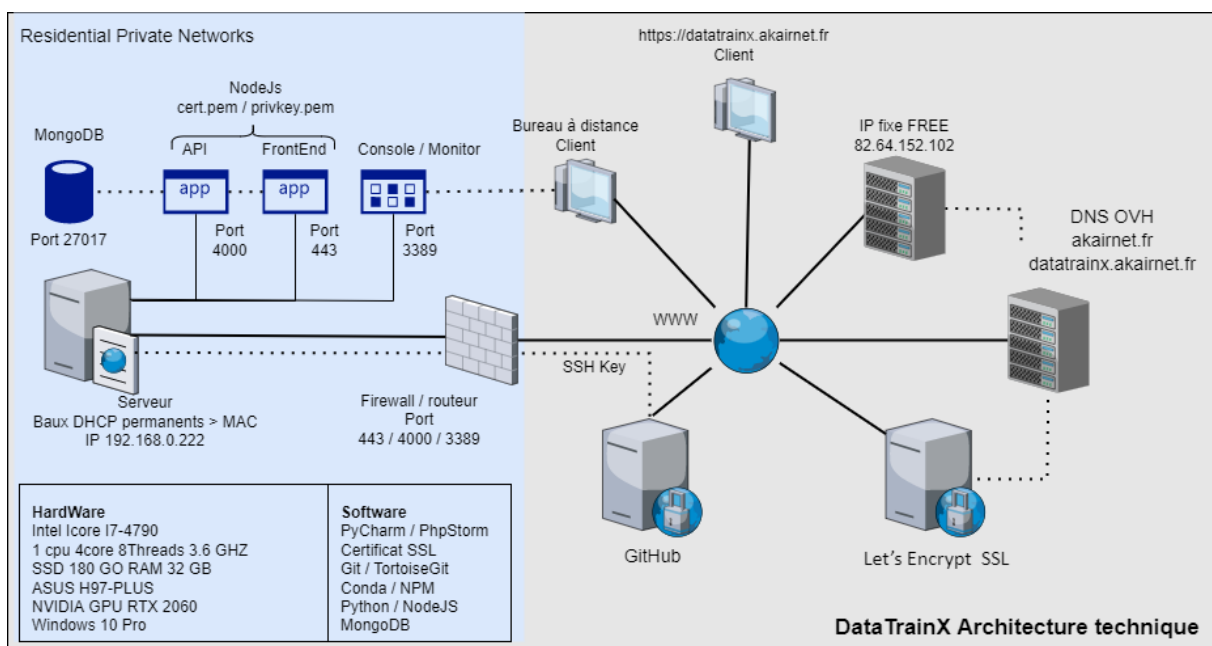


Figure 25 – Architecture technique DataTrainX

Côté applicatif nous utiliserons comme IDE PyCharm (2021.3) pour l'entraînement du modèle. Il offre un réel avantage dans la multitude de configurations possibles dans la gestion des paquets et redondances (choix de l'interpréteur, dépendances des bibliothèques). C'est un atout considérable en deep-learning quand il faut jongler avec les différentes versions.

Cependant en développement Frontend et Backend nous utiliserons plutôt l'IDE PhpStorm (2020.2) plus adapté pour gérer convenablement notre architecture logicielle hybride en reconnaissant l'ensemble des langages sur un même projet (Python, JS, HTML, CSS). Côté serveur nous utiliserons NodeJS et MongoDB pour une base de données NOSQL.

Les bibliothèques et dépendances sont installées avec CONDA [62], NPM [63] ou PIP. L'ensemble de notre projet est géré sur les versions avec GIT, TortoiseGit et déposé sur GitHub à l'adresse <https://github.com/davy-blavette/DatatrainingX>.

Les architectures étant déterminées, nous avons démarré le développement de notre application. La première étape a consisté à générer le modèle de données par entraînement CNN en fonction des données que nous avons pu trouver. C'est ce que nous étudierons dans le prochain chapitre.

2.3 - Entraînement du modèle CNN

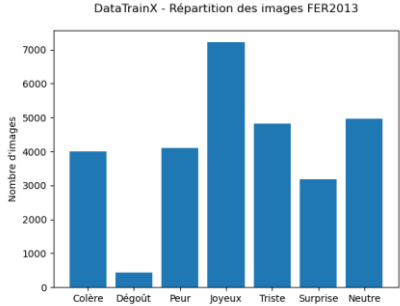
Nous décrivons dans ce chapitre les différentes étapes que nous avons réalisées pour construire un modèle de reconnaissance des émotions sur la base d'un CNN. Nous détaillerons notre développement, les grandes étapes de convolution, nos choix techniques et les différents résultats que nous avons obtenus. L'ensemble de ce chapitre sera développé avec le langage Python et le code source est disponible sur GitHub [64].

2.3.1 - Le dataset FER2013

Le plus important c'est d'avoir le plus d'images et si possible de manière qualitative. Notre premier choix s'est orienté sur le dataset de FER2013 [49]. Comparativement à nos autres dataset, FER2013 a l'avantage d'être le plus simple à mettre en œuvre. Les images classées sur les 6 émotions de base sont en niveau de gris (grayscale) ce qui simplifie la convolution à une seule couche d'entrée au lieu de trois (RGB), pour une image couleur. Les images sont prédécoupées sur le visage et sont de petites résolutions de 48 px (figure 26).

La base de données est prédécoupée avec une « base de données d'entraînement » comportant 29 068 images et une « base de données de validation » comportant 3 589 images. Le dataset ne fournit pas les images sous forme de fichier, mais un unique fichier CSV, chaque ligne comportant les pixels de l'image. La figure 26 est une transformation inverse de ces données. Ces données ne sont pas au format image, mais dans un format appelé DataFrame. Voici une description de ce fichier CSV (tableau 3) :

Tableau 3 – Caractéristiques du fichier CSV FER2013

Décomposition d'une ligne CSV : <i>Exemple : 0,70 80 82 72 58 58 60 (...) 109 82, Training</i>	 <table border="1"><caption>DataTrainX - Répartition des images FER2013</caption><thead><tr><th>Émotion</th><th>Nombre d'images</th></tr></thead><tbody><tr><td>Colère</td><td>4000</td></tr><tr><td>Dégoût</td><td>500</td></tr><tr><td>Peur</td><td>4000</td></tr><tr><td>Joyeux</td><td>7000</td></tr><tr><td>Triste</td><td>4800</td></tr><tr><td>Surprise</td><td>3200</td></tr><tr><td>Neutre</td><td>5000</td></tr></tbody></table>	Émotion	Nombre d'images	Colère	4000	Dégoût	500	Peur	4000	Joyeux	7000	Triste	4800	Surprise	3200	Neutre	5000
Émotion		Nombre d'images															
Colère	4000																
Dégoût	500																
Peur	4000																
Joyeux	7000																
Triste	4800																
Surprise	3200																
Neutre	5000																
1 ^{er} champ : valeur de 0 à 6 (Colère, Dégoût, Peur, Joyeux, Triste, Surprise, Neutre) 2 ^{ème} champ : série de pixel 3 ^{ème} champ : catégorie d'entraînement 'Training' 29 068 images, 'PublicTest' 3 589 images, 'PrivateTest' 3 230, images <i>Fichier comportant 35887 lignes de taille 294 mo.</i>																	

Il y a 7 catégories d'émotions dans cet ensemble de données et le dégoût a le minimum d'images autour de 5 à 10 % des autres classes. L'ensemble de ces facteurs nous permet

d'obtenir des résultats significatifs dans un temps de calcul raisonnable. FER2013 a organisé un concours en 2013 consistant à obtenir le meilleur score de prédiction de l'émotion. À l'époque, le meilleur score était de 71%, actuellement il est de 73.70 %. Nous verrons que nous obtenons lors de nos tests un score de 66.73 % pour 1 h 00 de calcul. L'amélioration de ce score dépendra du modèle CNN à utiliser et des performances de la machine.



Figure 26 – Exemple d'images FER2013 après transformation des pixels en image

Les données contiennent un large éventail d'images telles que des hommes, des femmes, des enfants, des personnes âgées, des blancs, des noirs, etc. Ils contiennent des images non humaines, comme des dessins animés ou des images sans aucun visage. Ces images ne sont pas obtenues en laboratoire dans des conditions d'éclairage appropriées (comme CK+). Mais ils sont plutôt extraits du Web. Notons qu'il s'agit d'un problème d'apprentissage supervisé, comme défini sur la figure 27, le modèle appris y est une fonction des données x .

$$y = f(x)$$
$$model = f(data)$$

Figure 27 – Equation définissant l'apprentissage supervisé du modèle

2.3.2 - Préparer le jeu de données

La première étape pour l'analyse d'un fichier CSV comportant une liste de pixels est de reconstruire les images et de stocker l'ensemble dans un tableau. Pour cela, nous utilisons la bibliothèque NumPy, destinée à manipuler des matrices ou tableaux multidimensionnels ainsi que des fonctions mathématiques opérant sur ces tableaux. Nous formalisons cette étape par la fonction *preprocessing()*. L'objectif est de générer deux fichiers représentant par leur index un tableau d'images et un tableau de labels. Ces deux fichiers n'ont pas pour objectif d'être régénérés par la suite. Le but est de convertir chaque image en une image carrée tridimensionnelle de taille 48x48x1. Il s'agit d'une image en niveaux de gris, il n'y a donc qu'un seul canal, contrairement à une image RGB où nous aurions débuté un layer de 48x48x3. Nos images (x) sont prêtes, mais nous devons également rendre nos étiquettes (y) compatibles avec notre modèle. C'est ce que nous réalisons en mappant l'étiquette de classe d'origine à la

nouvelle étiquette. Nous représentons cette opération par le code Python du code 1 ci-dessous :

```
def preprocessing():
    # lecture du dataset d'origine
    data = pd.read_csv('../data/fer2013.csv')
    datapoints = data['pixels'].tolist()
    # génération d'un tableau d'images en préparation CNN
    X = []
    for xseq in datapoints:
        xx = [int(xp) for xp in xseq.split(' ')]
        xx = np.asarray(xx).reshape(width, height)
        X.append(xx.astype('float32'))

    X = np.asarray(X)
    X = np.expand_dims(X, -1)
    # génération d'un tableau de labels émotions en préparation CNN
    y = pd.get_dummies(data['emotion']).to_numpy()
    # sauvegarde des informations pour traitement ultérieur
    np.save('./preprocessing/ferdataX', X)
    np.save('./preprocessing/flabels', y)
    np.save('./preprocessing/features', X[0])
```

Code 1 – Préparation du jeu de données avant convolution, mappage des images avec les étiquettes en rouge

Le tableau x (flèche rouge) est donc un ensemble d'images abstraites représenté sous forme de 48 caractéristiques, obtenu sur une série de 2304 pixels divisés par la taille de l'image (48).

2.3.3 - Entraînement et choix du modèle

L'algorithme d'entraînement CNN est déterminé par l'architecture de convolution choisie. C'est à ce niveau que les évolutions en termes de performance se font. C'est ce qu'on appelle aussi le choix des hyperparamètres. Pour une base de données comme FER2013 il existe un benchmark [49], permettant de référencer les scores obtenus selon le modèle utilisé. Je reporte dans le tableau ci-dessous ces scores en comparaison de nos résultats sur les 8 modèles testés :

Tableau 4 – Benchmark mondial par modèle sur FER2013 et comparaison avec nos résultats sur 8 modèles

Modèle FER2013	Record	Année	Nos résultats						
			Moyenne	Meilleur	Epoch	Filtres	Batch	Temps	Total
ResNet152V2[65]	73.70 %	2021	54.30 %	72 %	100	64	256	2 ms	1 h 50 min
VGG19[65]	73.28 %	2021	52.55 %	78 %	100	64	256	0,9 ms	0 h 50 min
tinyVGG[66]	*	2022	52.24 %	73 %	1000	10	32	0,1 ms	1 h 05 min
CNN Custom[67]	72.16 %	2021	65.12 %	82 %	100	64	256	0,7 ms	0 h 36 min
DCNN[68]	72.16 %	2021	65.90 %	85 %	100	64	256	0,9 ms	2 h 30 min
EDNN [69]	*	2019	46.76 %	68 %	1000	32	256	0,1 ms	0 h 50 min
DeepEmotion[70]	70.02 %	2016	66.73%	87 %	100	64	256	1 ms	1 h 00 min
NASNetLarge[65]	*	2021	53.41 %	72 %	100	64	64	10 ms	7 h 30 min

Chacun de ces modèles est issu de la recherche et fait l'objet de thèses. Notre objectif était donc de les utiliser et de les mettre en œuvre sans pour autant chercher à les améliorer. Nous avons testé plusieurs modèles sans réussir à atteindre de meilleurs scores dans ce domaine. « DCNN Custom » le plus simple dans sa compréhension fut également un des meilleurs en

termes de précision/temps. C'est d'ailleurs pour cette raison que nous focaliserons nos explications sur ce dernier. Le choix du modèle dans sa génération, dépend de plusieurs facteurs, sa complexité à être mis en œuvre et son entraînement dans un temps 'raisonnable' voir tout simplement compatible avec nos ressources matérielles. Un modèle se paramètre sur le nombre de passages complets dans l'ensemble de données d'apprentissage, c'est ce qu'on appelle le nombre de « Epochs ». Dans l'absolu, nous pourrions réaliser un nombre infini d'itérations, le modèle apprenant sur lui-même, il pourrait toujours trouver une meilleure solution pour se rapprocher du taux de 100% de réussite. Néanmoins, il finit par se stabiliser à un certain stade, aussi 100 Epochs nous paraissent correctes dans un rapport durée/précision. Selon une taille de batch défini, un « Epoch » traite à chaque fois l'ensemble des 29 068 images du modèle. Le temps d'un « Epoch » complet dépend de la complexité du modèle (complexité algorithmique), de la taille du batch (mémoire allouée) et de la taille de l'échantillon. On parle aussi de temps d'inférence, c'est-à-dire le temps à l'exécution d'un modèle CNN une fois celui-ci entraîné sur un dataset d'apprentissage puis testé sur un dataset de validation. Pour nos tests, nous avons expérimenté différentes configurations, mais globalement, nous avons paramétré l'ensemble de nos modèles au nombre de 100 « Epochs » et une taille de batch à 256. Plus haut, la mémoire de notre processeur graphique est saturée, ce qui peut expliquer la raison d'un taux de réussite fluctuant. La figure 28 ci-dessous est une capture d'une convolution CNN sur le traitement de l' « Epoch » 32 avec la taille du batch à 64 et un temps d'inférence à 27 secondes (modèle CNN Hyperparameter).

```

28992/29068 [=====>.] - ETA: 0s - loss: 0.8748 - accuracy: 0.6883
29056/29068 [=====>.] - ETA: 0s - loss: 0.8746 - accuracy: 0.6883
29068/29068 [=====] - 27s 928us/step - loss: 0.8746 - accuracy: 0.6884 - val_loss: 1.0226 - val_accuracy: 0.6449
Epoch 32/100

 64/29068 [.....] - ETA: 27s - loss: 0.8901 - accuracy: 0.6250
128/29068 [.....] - ETA: 26s - loss: 0.8093 - accuracy: 0.7266
192/29068 [.....] - ETA: 26s - loss: 0.8225 - accuracy: 0.7292
256/29068 [.....] - ETA: 26s - loss: 0.7843 - accuracy: 0.7422
320/29068 [.....] - ETA: 26s - loss: 0.7575 - accuracy: 0.7531
384/29068 [.....] - ETA: 26s - loss: 0.7431 - accuracy: 0.7552
448/29068 [.....] - ETA: 26s - loss: 0.7483 - accuracy: 0.7522
512/29068 [.....] - ETA: 25s - loss: 0.7753 - accuracy: 0.7402
576/29068 [.....] - ETA: 25s - loss: 0.7636 - accuracy: 0.7396
640/29068 [.....] - ETA: 25s - loss: 0.7904 - accuracy: 0.7297

```

Figure 28 – Visualisation du traitement d'un EPOCH en 27 secondes sur le modèle CNN Hyperparameter

Pour réaliser notre convolution, nous avons utilisé la bibliothèque Keras. Certains de nos tests nous ont orientés sur l'utilisation de la bibliothèque Pytorch que nous avons à terme abandonnée pour des problèmes de ressources matérielles ou de complexité. Nous retiendrons que Pytorch est une excellente bibliothèque pour entraîner un réseau convolutif, sûrement plus performant pour des mathématiciens et des chercheurs expérimentés, mais

Keras fut à terme beaucoup plus adaptée à notre façon de faire : travailler sur de petits ensembles de données et réaliser un prototypage rapide. Keras propose aussi des modèles préformés [65] qui peuvent être directement utilisés, ce que nous avons fait pour ResNet50V2 ou VGG19 comme dans l'exemple code 2 ci-dessous. Nous utilisons la méthode `getAttr()` pour récupérer un modèle en fonction d'une chaîne de caractères définie lors de notre configuration. Ceci pour simplifier notre code. Ce même principe est d'ailleurs utilisé pour différencier les modèles 'custom' :

```
# Load model, custom or Keras
try:
    model = getattr(models.Models(), modelName)(width=width,height=height,num_features=num_features,num_labels=num_labels)
except:
    model = models.Models().load(modelName=modelName, width=width,height=height,num_labels=num_labels)

model.summary()

def load(self, modelName, width, height, num_labels):
    base_model = getattr(keras.applications, modelName)(include_top=False,
                                                         weights=None,
                                                         input_shape=(width, height, 1),
                                                         pooling="avg")

    emotion = Dense(units=num_labels, kernel_initializer="he_normal", use_bias=False,
                    activation="softmax", name="emotion")(base_model.output)
    model = Model(inputs=base_model.input, outputs=emotion)
    model.compile(loss=categorical_crossentropy,
                  optimizer=Adam(learning_rate=0.001, beta_1=0.9, beta_2=0.999, epsilon=1e-07),
                  metrics=['accuracy'])

    return model
```

Code 2 – Import du modèle CNN VGG19 préformé par Keras - Class models.py

2.3.4 - Définition du modèle CNN

Un modèle CNN, comme expliqué au chapitre 1.6.1 dépend d'une architecture où se succèdent des types de convolutions, Relu, Maxpooling et Flatten. Nous mettons en annexe 2 le modèle complet CNN Hyperparameter que nous avons utilisé. En Python avec la bibliothèque Keras, cela s'exprime dans la représentation du code 3 ci-dessous. Le code complet fait une cinquantaine de lignes. Nous ne mettons ici que le premier enchaînement, pour en comprendre le principe :

```
def cnn(self, width, height, num_features, num_labels):
    model = Sequential()

    model.add(Conv2D(num_features, kernel_size=(3, 3), activation='relu', input_shape=(width, height, 1),
                    data_format='channels_last', kernel_regularizer=L2(0.01)))
    model.add(Conv2D(num_features, kernel_size=(3, 3), activation='relu', padding='same'))
    model.add(BatchNormalization())
    model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
    model.add(Dropout(0.5))
```

Code 3 – Première convolution sur une donnée d'entrée

On voit ici la première étape du traitement, le nombre de filtres que l'on applique lors du balayage raster de l'image, la taille du filtre et la taille séquentielle de l'image. Le modèle est construit avec une forme d'entrée prédéfinie et une forme de sortie définie. L'activation de type RELU (Unité linéaire rectifiée) est ici activée. Sur d'autres tests notamment implémentés sur le modèle « DCNN » nous avons utilisé l'activation ELU (Unité linéaire exponentielle) qui s'est avérée plus efficace. Le modèle est ensuite compilé avant d'être entraîné, ce que l'on voit dans le code 4 ci-dessous :

```
model.add(Dense(num_labels, activation='softmax'))
# Compilation du modèle avec l'optimiseur Adam et la perte d'entropie croisée catégorique
model.compile(loss=categorical_crossentropy,
              optimizer=Adam(learning_rate=0.001, beta_1=0.9, beta_2=0.999, epsilon=1e-07),
              metrics=['accuracy'])

return model
```

Code 4 – Compilation du modèle avec l'optimiseur Adam et la perte d'entropie croisée catégorique

La perte d'entropie croisée est une métrique utilisée pour mesurer la performance d'un modèle CNN. La perte est mesurée par un nombre compris entre 0 et 1. Zéro correspondant à un modèle parfait. L'objectif est généralement d'obtenir un modèle aussi proche de 0 que possible. Pour cela, nous utilisons un algorithme d'optimisation, Adam ici, avec les paramètres préconisés par Tensorflow [71]. Lorsque le modèle est correctement configuré vient la phase d'entraînement. Pour cela nous passons par la méthode `fit()` qui consiste à itérer sur le jeu de données (Epochs). C'est là que rentre en jeu, les deux fichiers que nous avons préparés en amont, `fdataX` pour les données et `fdataY` pour les labels. La finalité est de diviser les données en ensemble de formations et de validations. Nous nous entraînerons sur les données d'entraînement et validerons notre modèle sur les données de validation. Nous représentons cette opération par le code 5 ci-dessous :

```
x = np.load('./preprocessing/fdataX.npy')
y = np.load('./preprocessing/flabels.npy')
x -= np.mean(x, axis=0)
x /= np.std(x, axis=0)
# splitting into training, validation and testing data
[x_train, x_test, y_train, y_test] = train_test_split(x, y, test_size=0.1, random_state=42)
[x_train, x_valid, y_train, y_valid] = train_test_split(x_train, y_train, test_size=0.1, random_state=41)

# training the model
model.fit(np.array(x_train), np.array(y_train),
        batch_size=batch_size,
        epochs=epochs,
        verbose=1,
        validation_data=(np.array(x_valid), np.array(y_valid)),
        shuffle=True)
```

Code 5 – Entraînement du modèle CNN sur le jeu de données

À la fin de l'entraînement, nous obtenons plusieurs résultats, que nous sauvegardons dans 3 fichiers pour réutilisation dans l'application DataTrainX afin d'obtenir une prédiction. Le fichier *model.json* qui correspond à la définition du modèle utilisé. Il sera utilisé notamment pour des besoins de conversion pour le chargement de ce modèle avec TensorflowJS. Le fichier *best.h5* qui comprend l'ensemble de nos résultats, l'architecture du modèle, les poids du modèle et l'état de l'optimiseur. Ce fichier permet aussi de réinstancier le modèle et de reprendre l'entraînement exactement là où nous l'avons laissé. Cela permet de sauvegarder l'intégralité de l'état d'un modèle dans un seul fichier. Enfin, le fichier *weight.h5* utilisé pour nos prédictions contient les caractéristiques trouvées (feature/layer) et une liste de chaînes (noms ordonnés des couches du modèle). Pour chaque couche, un groupe nommé `layer.name`. Pour chaque groupe de couches, un attribut de groupe `weight_names` et une liste de chaînes (noms ordonnés des tenseurs de poids de la couche). Enfin, pour chaque poids de la couche, un ensemble de données stockant la valeur de poids, nommé d'après le tenseur de poids. Nous sauvegardons aussi les valeurs x_{test} et y_{test} car associées avec les fichiers *model.json* et *best.h5*. Cela nous permet à court terme de générer une matrice de confusion afin d'obtenir des statistiques de prévision. C'est ce que nous verrons dans la prochaine section.

2.3.5 - Précision et matrice de confusion

Lorsque l'entraînement est réalisé, nous souhaitons avoir une analyse des résultats obtenus. Pour cela, nous avons deux options. La première option est d'utiliser Tensorboard pour obtenir une courbe d'apprentissage de la progression de notre entraînement. Cela nous permet d'observer si l'entraînement se déroule logiquement ou si la courbe d'entraînement est stagnante ou en progression, après un certain nombre d'Epochs. Exemple figure 29.

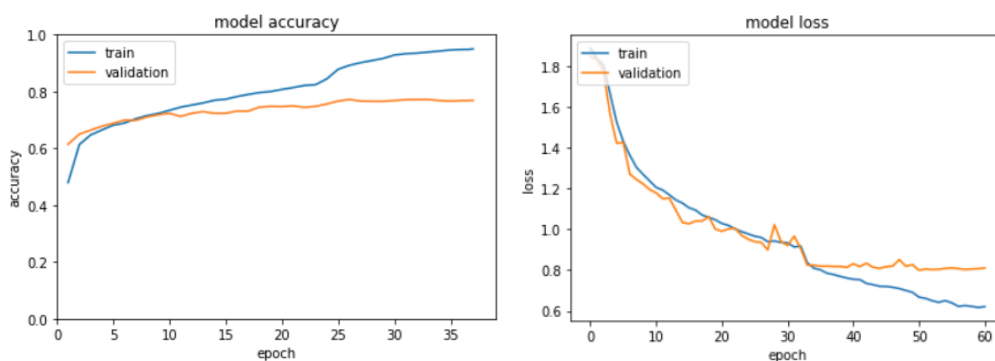


Figure 29 – Exemple de représentation d'un entraînement avec Tensorboard

La seconde option est d'orienter nos restitutions de résultats avec la génération d'une matrice de confusion. C'est l'un des évaluateurs les plus utilisés pour la classification multiclasse. Cela nous donne un bon aperçu des performances du modèle sur toutes les classes. Nous réalisons cette étape par une analyse des résultats en testant l'ensemble de notre fichier en fonction des poids obtenus avec l'aide des fichiers *model.json* et *best.h5*. C'est ce que nous représentons dans le code 6 ci-dessous :

```

yhat = loaded_model.predict(x)
yh = yhat.tolist()
yt = y.tolist()
count = 0

for i in range(len(y)):
    yy = max(yh[i])
    yyt = max(yt[i])
    predy.append(yh[i].index(yy))
    truey.append(yt[i].index(yyt))
    if yh[i].index(yy) == yt[i].index(yyt):
        count += 1

acc = (count / len(y)) * 100

```

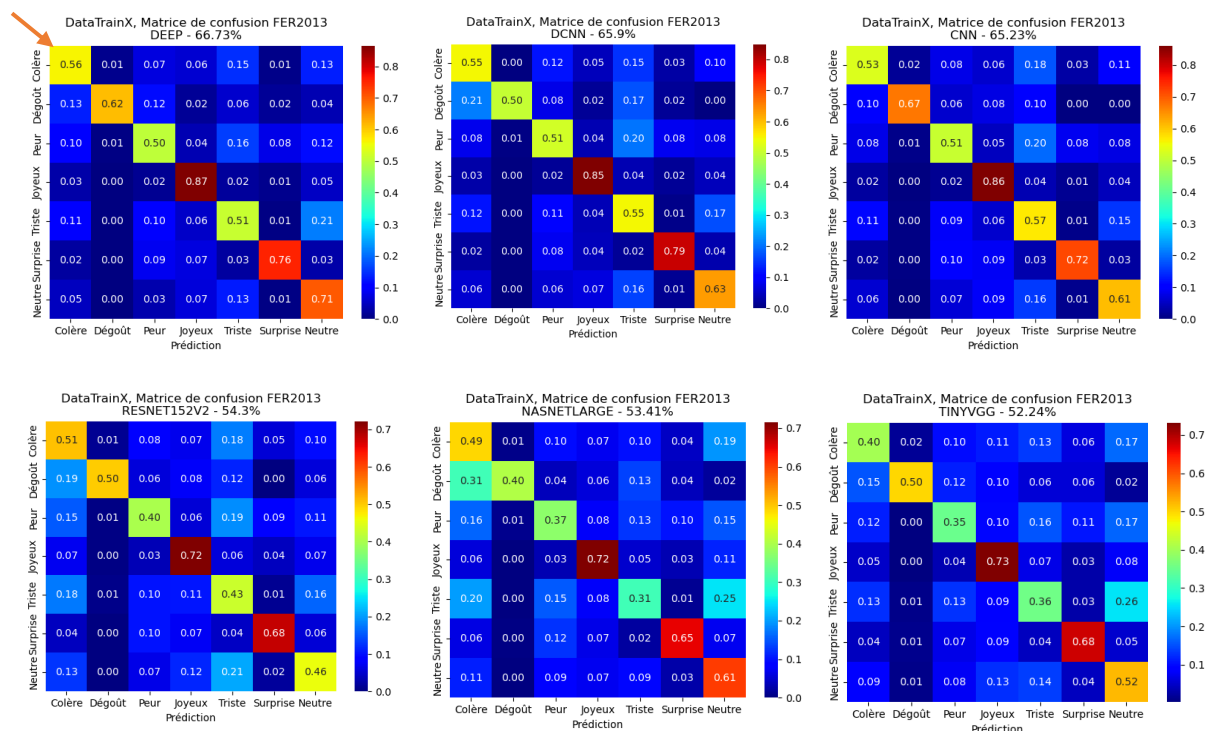
Code 6 – Génération d'une matrice de confusion en utilisant le modèle sur les données tests

Suite au chargement de ces deux fichiers, exprimé dans le code par *load_model* nous déterminons une prédiction par rapport aux valeurs *x_test* (exprimée *x*) et *y_test* (exprimée *y*), ce qui nous permet de générer la matrice avec la bibliothèque *sklearn*. En figure 30 nous représentons les différentes matrices de confusion que nous avons générées suivant les modèles que nous avons testés. Les matrices de confusions permettent de représenter sous forme de graphique les résultats de prévision du modèle entre les données réelles sur l'axe *y* et les données de prédiction sur l'axe *x*. Par exemple, sur le modèle Deep (en haut à gauche de la figure 29), nous affichons un résultat moyen de 66.73 % de prévision. Comme illustré dans le tableau 6, ce chiffre est une moyenne entre le nombre d'images trouvées (2395) par rapport au nombre d'images tests (3789).

	Colère	Dégout	Peur	Joie	Triste	Surprise	Neutre
Colère	280	6	37	31	73	5	66
Dégout	7	32	6	1	3	1	2
Peur	54	3	273	21	87	43	64
Joie	28	1	14	763	19	11	45
Triste	63	0	61	33	302	3	126
Surprise	10	0	36	30	11	313	14
Neutre	31	2	21	42	79	4	432

Tableau 5 – Résultats trouvés sur le modèle Deep sur une base de 3789 images tests.

Dans cet exemple, 280 images ont bien été déterminées « colère » sur les 498 que compte le jeu de test, soit un ratio de réussite de 56% (flèche rouge).



Nous n’avons pas toujours réussi à implémenter les modèles les plus performants, la raison principale c’est qu’ils sont utilisés dans une configuration matérielle et dans un environnement de développement dédié à cela, ce qui n’était pas notre cas pour des raisons d’agilité. Par ailleurs, FER2013 n’est pas ce qui se fait de mieux, des dataset plus performants sont disponibles, mais souvent réservés dans le cadre de recherches. Néanmoins, nous n’avons pas négligé cette étape qui nous a permis d’acquérir un apprentissage et des résultats significatifs pour la poursuite de notre application. Des modèles déjà entraînés sous format h5 avec de meilleures précisions sont disponibles et nous les utiliserons.

Nous retiendrons des axes d’améliorations concernant l’algorithme et le développement de notre code d’apprentissage. Nous aurions souhaité, par exemple, utiliser un prétraitement sur les images pour gérer par exemple la luminosité, la rotation, les effets de cisaillement qui auraient certainement amélioré la performance du modèle. Les prochaines étapes consistent aux développements de notre application où nous verrons comment nous utiliserons les modèles pour obtenir une prévision. Préalablement, nous devons maquetter notre application et déterminer nos choix de base de données, ce qui clora ce chapitre 2 de conception.

2.4 - Maquettage de l'application

Avant la phase de développement de l'application, nous avons schématisé nos différentes interfaces. Pour des raisons d'agilité, nous l'avons réalisée en même temps que la création de notre arborescence. Cela nous a permis de visualiser une première approche de notre application et la cohérence de nos zones de contenus, notamment pour respecter une norme W3C (header, main, footer, h1, article, sections, etc.). L'idée principale est de faire une « single page » interactive en gardant à l'esprit que nous travaillerons à terme avec NodeJS. En effet, certaines parties seront rechargées pour l'affichage des différents contenus. C'est ce que nous indiquons avec la notion de « switch » avec notre arborescence dans le tableau 6 ci-dessous :

Tableau 6 – Arborescence prévisionnelle de DataTrainX

Template	Description	Balise W3C	Switch
App	Loader	---	
_Header	Entête de page	Header	
_DatatrainX	Squelette du site	Main	
_Presentation	Page d'accueil	section, article, h2	✓
_Cnn	Pédagogie interactive CNN	section	✓
_CnnExplainer	14 templates repris du projet	Svg	
_FaceApi	Capture vidéo émotion	video, canvas	✓
_OpenData	Données disponibles au format JSON	JSON-LD	✓
_TrainX	Cœur de l'application, Tests	(layout)	
_ChartStream	Lecture des émotions sur graphique	canvas	
_Token	Attribution token	article, button	✓
_Kolb	Questionnaire Kolb	radio, button	✓
_Resultats	Pages de résultats, graphiques	article, canvas	✓
_Article	Détails de l'application	section, article, h1	
_Footer	Pied de page	footer	

L'objectif est de permettre à l'utilisateur de réaliser le test simplement, mais en axant sur l'information pour que ce soit pédagogique et rassurant sur le traitement des données. Nous souhaitons aussi que le site soit responsive, c'est-à-dire compatible sur un PC mais aussi sur une tablette ou un smartphone. Nous ne cherchons pas une entière compatibilité de l'application, mais que celle-ci soit présentable et lisible pour la partie article notamment. Pour cela, nous envisageons de nous aider de bibliothèques légères comme BULMA [72]. Les templates seront gérés avec la bibliothèque SVELTE [73] qui permet la création d'interfaces utilisateurs pour NodeJS. Les besoins et technologies étant posés, nous réalisons la première

phase de maquettage appelée zoning et mockup. Ces dernières représentent nos interfaces en figure 31. Nous proposons de découper notre interface en 4 parties structure HTML5. Avec un « header », un « main » qui comptera deux « sections » et un « footer ». Le header sera dédié au logo et accueillera un menu, il sera en position css « fixed » afin d’être continuellement accessible. Le « main » accueillera le cœur de l’application, une première section dynamique avec le principe de switch. L’objectif est que cette partie prenne la taille de l’écran client. Une deuxième section accessible en « scrolling » dédiée à l’explication de DataTrainX.

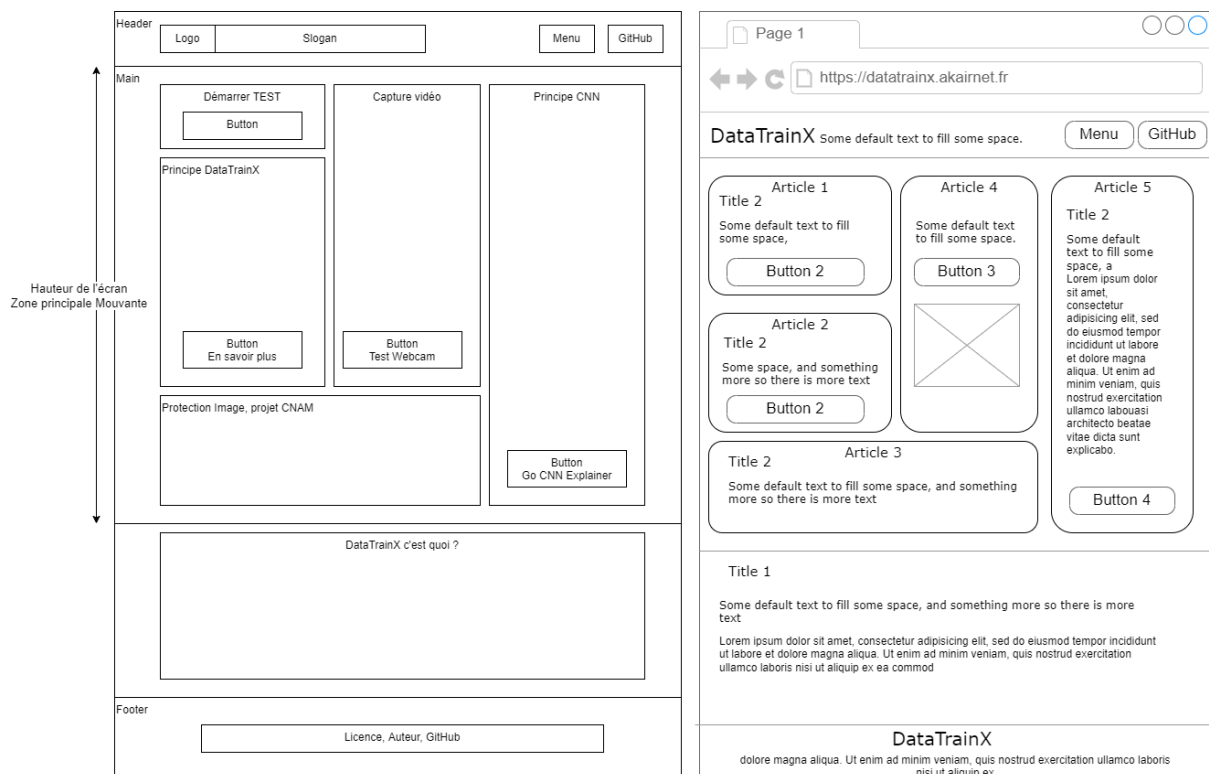
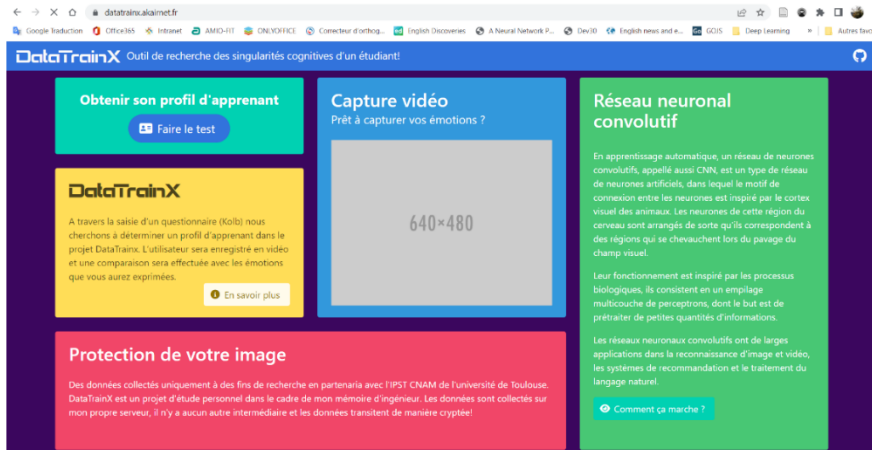


Figure 31 – A gauche le zoning (zones d’informations grossières), à droite le mockup avec découpage des 4 parties principales de l’application.

Le mockup est une image d’interface qui a été transformée en page HTML, dynamique et navigable. Ce nouveau format autorise l’insertion de liens vers des pages notamment. Il permet, aussi, de rendre un formulaire fonctionnel afin d’effectuer des simulations. Assez rapidement, cela nous a permis de finaliser notre prototype et de proposer un visuel semi-définitif que nous représentons en figure 32. La partie présentation devra prévoir notamment une information synthétique et précise des attentes de l’application. Il est donc prévu 5 articles. Un article « faire le test » pour lancer le cœur de l’application, un article résumé de

Datatraining afin d'expliquer à quoi sert l'application, un article sur la protection des données et l'open data afin de sécuriser l'utilisateur sur ce qui sera enregistré. Enfin un dernier article sur le principe d'un réseau neuronal convolutif. Ce dernier article proposera un lien vers le CnnExplainer de DataTrainX afin de visualiser le fonctionnement d'un réseau de convolution.



DataTrainX c'est quoi ?

Outil de recherche des singularités cognitives d'un étudiant!

Dans l'apprentissage, une logique de la restitution qui prévaut encore sur une logique de la compréhension serait à l'origine de nombreux échecs de l'apprenant. Pour se comprendre, comprendre le monde et autrui, tout apprenant produit et met en œuvre des ressources métacognitives.

On observera que cela fait appel à de nombreuses disciplines, comme la psychologie, la pédagogie, les neurosciences mais aussi le management, qui demandera à l'équipe enseignante une formation continue pour s'appuyer sur de tels dispositifs.

DataTrainX est un prototype de reconnaissance faciale des émotions (REF) qui a pour objectif principal de s'intéresser à l'analyse des comportements de l'apprenant dans un but de neuropédagogie.

Typologie d'apprentissage et émotions

Il y aura donc une phase de saisie d'un questionnaire afin de déterminer un profil d'apprenant de type Kolb dans le projet DataTrainX. L'utilisateur sera enregistré en vidéo et une comparaison sera effectuée entre la typologie trouvée par le questionnaire et les émotions qu'il aura exprimées. Comme l'illustre la figure ci-dessous, l'objectif est de trouver une corrélation entre la typologie d'apprentissage et l'émotion de l'utilisateur.

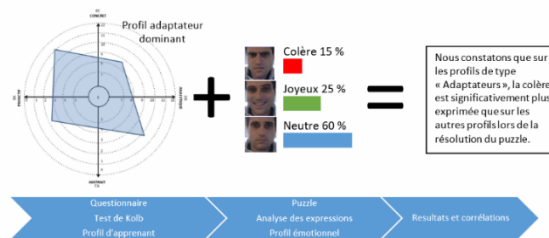


Figure 3. Exemple de restitution des résultats de l'application DataTrainX. Obtenir un profil d'apprenant, analyser l'émotion dans une mise en situation, trouver une corrélation.

Dans l'hypothèse où nous aurions des résultats significatifs entre expression et typologie, nous pourrions considérer que le profil émotionnel même de l'émotion détermine la typologie d'apprentissage et donc envisager, par exemple, une orientation du logiciel vers l'adaptative learning, c'est-à-dire une interaction entre l'IA et l'utilisateur dans la proposition des supports pédagogique.

L'objet de ce projet, pour orientation de se focaliser concrètement sur les moyens fonctionnels informatique à mettre en œuvre pour ce type d'application. Nous n'approfondirons pas l'aspect psychologique, qui se limitera au test de Kolb et à la résolution d'un Puzzle, l'objectif étant de démontrer comment mettre en place l'architecture fonctionnelle, logicielle et matérielle pour arriver à un résultat applicatif de la reconnaissance faciale de l'émotion.

Les résultats d'ordre psychologique à prendre en considération sur l'objectif de cette application auront donc une part importante de subjectivité clairement assumé.

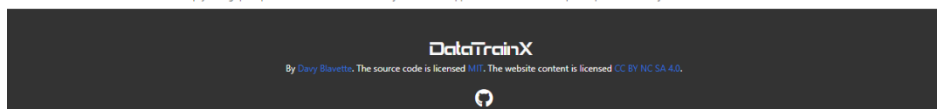


Figure 32 – Maquette prototype DataTrainX Home Page

Nous ne prévoyons pas de réaliser d'autres maquettes, comme la représentation des données ou la saisie du questionnaire. Elles reprendront la charte graphique principale que nous souhaitons « coloré » pour le côté ludique de l'application, nous les verrons dans le chapitre 3 développement. Nous terminons cette partie de conception avec le prochain chapitre sur le système de gestion des données.

2.5 - Système de gestion des données

Les données que nous souhaitons enregistrer sont relativement simples et ne demandent pas une quantité de tables et des relations importantes. Néanmoins, chaque utilisateur contient un nombre important de données. Cela est lié au fait d'un enregistrement constant d'un flux de données en rapport avec les émotions qu'ils exprimeront lors de la réalisation du test. Nous prévoyons donc de manipuler un large volume de données, mais dont les relations entre celles-ci ne sont pas particulièrement importantes. Par ailleurs, dans une logique d'évolution, nous nous sommes posé la question de la scalabilité, c'est-à-dire la faculté de DataTrainX à s'adapter aux fluctuations de la demande en conservant ses différentes fonctionnalités. L'idéal étant la scalabilité horizontale, à savoir l'augmentation du nombre de serveurs par une base de données distribuée au lieu d'une augmentation des capacités du serveur existant. Enfin, nos données étant rattachées à l'utilisateur que l'on nommera « trainer » par la suite, on privilégiera l'accès rapide aux données du « trainer » au lieu de la possibilité de réaliser des requêtes complexes. C'est en fonction de ces différentes stratégies que nous avons choisi de nous orienter sur du NoSQL.

2.5.1 - NoSQL et MongoDB

Le NoSQL est un terme désignant les bases de données qui ne sont pas relationnelles, elles sont apparues dans les années 2000 avec l'explosion du volume de données à traiter avec Internet et les réseaux sociaux. Le principal atout du NoSQL réside dans la simplicité à scaler par la division des données sur plusieurs instances. Sur MongoDB, il est aussi possible en fonction de paramétrages non standards [74], de préserver si besoin, les propriétés d'atomicité, de cohérence, d'isolation et de durabilité (propriétés ACID), que l'on retrouve sur les SGBD traditionnelles. Disponible en open source, MongoDB fait partie de ces bases de données NoSQL dans la famille « orientée document ». C'est-à-dire dans un format de stockage des données sur le modèle de documents JSON. Chaque document contient des paires de champs / valeurs. Les valeurs peuvent être des chaînes de caractères, des nombres, des booléens, des tableaux et même des objets. Ce format nous intéresse particulièrement, car notre application est développée en Javascript et le JSON est un format de données textuel dérivé de la notation des objets du langage JavaScript. Par ailleurs, MongoDB offre une très

grande flexibilité, il est facile d'ajouter ou retirer des champs au fur et à mesure que les besoins de l'application changent. Sur le principe de la réalisation de nos tests, nous avons pour objectif d'imbriquer des documents dans d'autres documents. L'architecture MongoDB s'avère plus performante pour des requêtes en lecture et en écriture qu'une base SQL. Il n'y a en effet pas plusieurs tables à consulter. Pour compléter, sur le choix de MongoDB dans la multitude de bases de données qu'il existe (SQL et NoSQL), nous pouvons aussi nous appuyer sur le Théorème de Brewer [75]. Ce théorème indique qu'il est impossible de garantir en même temps les contraintes de cohérence, de disponibilité ou de tolérance au partitionnement : « Qu'un système de calcul/stockage distribué ne peut garantir à un instant t que deux de ces contraintes, mais pas les trois ». Ce théorème permet de classer les bases de données selon les besoins prioritaires que nous représentons dans la figure 33, repris d'un travail sur openclassrooms [76] :

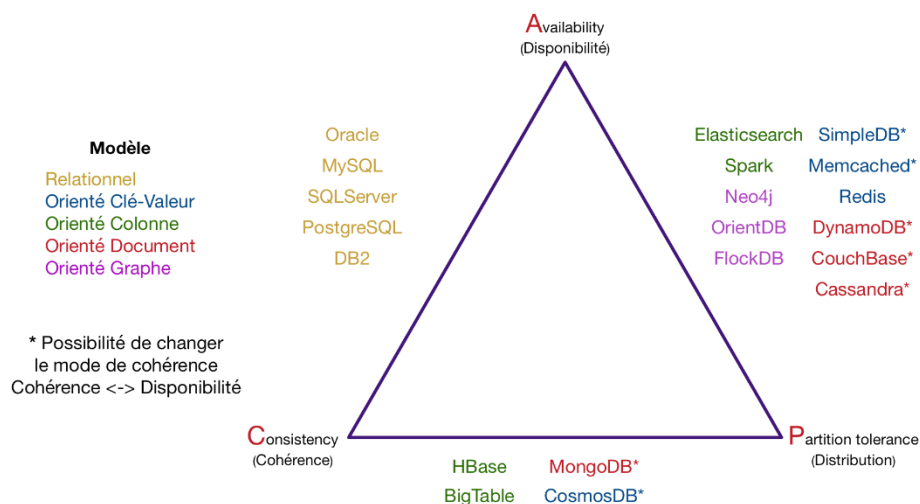


Figure 33 – Triangle réalisé à partir du théorème de Brewer, positionnant MongoDB sur les contraintes prioritaires de distribution et de cohérence [76].

Nous n'avons pas besoin d'une disponibilité importante de la base de données, car nous travaillerons de manière asynchrone avec des temps de sauvegarde ponctuels ou encore les données de résultats se feront sur des données déjà stockées par le client. On observe que dans des bases de données CP (cohérence + tolérance au partitionnement), la cohérence est maintenue en ayant une instance responsable de l'écriture et de la lecture. Les autres instances répliquent les données pour en assurer leur durabilité. Si l'instance leader est en panne, les autres instances vont élire un nouveau leader et la base de données sera indisponible pendant ce processus. La garantie que toute requête aura une réponse n'est donc

pas assurée dans cette configuration « standard », ce qui ne devrait pas être un problème pour nous. Notons qu’il est tout à fait possible de manuellement configurer MongoDB en mode AP, c’est-à-dire en privilégiant la disponibilité sur la cohérence.

2.5.2 - Modèle conceptuel et modèle logique de données

Le modèle conceptuel et le modèle logique de données sont des étapes essentielles dans la construction de bases relationnelles, mais ils ne sont pas si évidents pour les bases NoSQL. Sur MongoDB, le fait que des documents d’une même collection puissent avoir des champs différents et qu’il est très simple d’ajouter et/ou supprimer des champs à l’intégralité d’une collection nous permet d’orienter l’effort sur le développement. Comme à la manière agile, dont l’un des préceptes est de « se concentrer sur l’adaptation au changement davantage que sur le suivi et le respect d’un plan ». Nous ferons une rétrospection plus complète sur ce domaine, dans le chapitre 3.3 Data Mining et classification des données. On se contentera à ce stade de la définition sur l’ensemble des objets (documents) de la collection « trainer » (en anglais, nom de notre collection) à partir du tableau 7 JSON ci-dessous. L’objectif étant d’avoir une vision correcte des données que nous prévoyons d’obtenir afin d’orienter notre développement.

Tableau 7 – Collection envisagée sur le sujet « trainer » JSON MongoDB

Collection document datatrains.trainer	JSON
<pre> _id: ObjectId('6295cedc5ccdc296b7276c8') token: "08HlxbZsGqcW4A8Q" dataExpression: Array 0: Object FaceDetection: Object score: 0.06 box: Object imageDims: Object height: 480 width: 640 image: "0x0x死吾狸類型r管製∞林噪燥:臨匠括è" FaceExpression: Object colere: 0.28 degout: 0 peur: 0 joie: 0.24 triste: 0.08 surprise: 0.03 neutre: 0.34 created: 1653984971331 dataCondition: Array 0: Object ref: 0 dataProfil: Object theorist: 0 date: 1653983251388 </pre>	<pre> { "_id": { "oid": "6295cedc5ccdc296b7276c8" }, "token": "08HlxbZsGqcW4A8Q", "dataExpression": [{ "FaceDetection": { "score": 0.06, "box": { "height": 95.79, "width": 147.84, "x": 250.6, "y": 383.86 }, "imageDims": { "height": 480, "width": 640 }, "image": "0x0x死吾狸類型r管製∞林噪燥:臨匠括è" }, "FaceExpression": { "colere": 0.28, "degout": 0, "peur": 0, "joie": 0.24, "triste": 0.08, "surprise": 0.03, "neutre": 0.34 }, "created": 1653984971331 }, "dataCondition": [{ "ref": 0, "dataProfil": {}, "date": 1653983251388 }] } } </pre>

A ce stade nous prévoyons donc de créer une base de données DataTrainX avec seulement une collection « trainers ». Celle-ci intégrera un document par trainer lui-même identifié par un token. La taille de limitation du champ que l'on a l'habitude de noter en SQL n'est pas définissable en NoSQL. Le type attendu peut être cependant modélisé. Ce document est un ensemble de données au format JSON formalisé qui nous permet de réaliser le dictionnaire des données ci-dessous en tableau 8.

Tableau 8 – Dictionnaire des données DataTrainX

Nom / Clé	Type	Description
_id	ObjectId	Identifiant unique généré par MongoDB, token
token	String	Identifiant unique généré frontend
dataExpression	Array	Tableau regroupant la capture du visage et des expressions
__FaceDetection	Object	Ensemble des éléments en rapport avec le visage
__score	Double	Score de précision en pourcentage du visage détecté
__box	Object	Ensemble des éléments de capture du visage sur la webcam
__height	Double	Hauteur de la fenêtre visage
__width	Double	Largeur de la fenêtre visage
__x	Double	Position X du pixel horizontale
__y	Double	Position y du pixel verticale
__imageDims	Object	Information taille d'origine image webcam
__height	Int32	Hauteur de résolution de l'image
__width	Int32	Largeur de résolution de l'image
__image	String	Capture de l'image encodée en base64 et compression LZW [77]
__FaceExpression	Object	Ensemble des expressions émises
__colere	Double	Score de précision en pourcentage de l'expression colère détectée
__degout	Double	Score de précision en pourcentage de l'expression dégoût détectée
__peur	Double	Score de précision en pourcentage de l'expression peur détectée
__joie	Double	Score de précision en pourcentage de l'expression joie détectée
__triste	Double	Score de précision en pourcentage de l'expression triste détectée
__surprise	Double	Score de précision en pourcentage de l'expression surprise détectée
__neutre	Double	Score de précision en pourcentage de l'expression neutre détectée
__created	Double	Date, nombre de millisecondes écoulées depuis le 1er Janvier 1970
dataCondition	Array	Tableau regroupant la mise en situation du trainer
__ref	Int32	Référence numéro de question Kolb
__dataProfil	Object	Une option parmi un ensemble du score du profil (d'accord / pas d'accord)
__activist	Int32	Valeur 1 ou 0 si question fait référence à activist
__theorist	Int32	Valeur 1 ou 0 si question fait référence à theorist
__reflector	Int32	Valeur 1 ou 0 si question fait référence à reflector
__pragmatist	Int32	Valeur 1 ou 0 si question fait référence à pragmatist
__created	Double	Date, nombre de millisecondes écoulées depuis le 1er Janvier 1970
created	Double	Date, nombre de millisecondes écoulées depuis le 1er Janvier 1970

La visualisation des données se fera avec le logiciel MongoDB Compass et sur la page OpenData de l'application.

2.5.3 - Compass, CRUD et API REST

MongoDB Compass est une application permettant de visualiser facilement les données. Elle permet également de manipuler les données en faisant des opérations CRUD (create, read,

update, delete) sur un document. Enfin, elle permet de déboguer et d'optimiser ses requêtes, en analysant la performance des requêtes et l'utilisation des index. Cela nous a permis en phase de conception de réaliser nos maquettes de documents sur la collection « trainer ». Tel que nous l'avons envisagé sur la figure 24 (Architecture hybride MVC), nous prévoyons une couche modèle/contrôleur pour assurer les échanges avec la base de données. Il est prévu que la vue communique avec le contrôleur via une API REST. Aussi, c'est finalement sur le modèle (backend) et le service factory (frontend), que nous concentrerons l'effort de structuration de la base de données. Le NoSQL est un autre paradigme comparativement aux bases de données relationnelles, dans le sens où c'est bien le client qui structure les données et non l'inverse. Sur MongoDB en l'occurrence, en dehors d'un formalisme JSON, il est possible d'injecter n'importe quel type de données en base NoSQL. C'est le principe clé/valeur qui détermine réciproquement le champ et la donnée, et non une donnée accessible dans un champ typé comme en SQL. Nous envisageons donc la construction d'une API de type REST qui interagit avec la base de données en communication avec une architecture de type MVC, sur laquelle les applications peuvent communiquer comme l'expose la figure 34 ci-dessous.

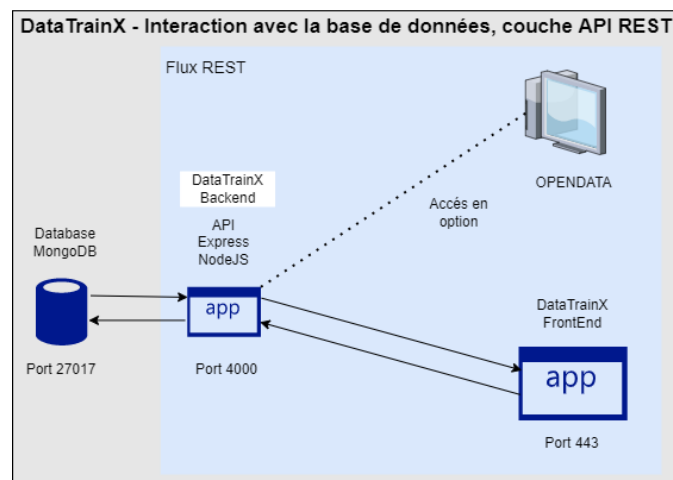


Figure 34 – Architecture de l'API REST sur DataTrainX

L'accès direct aux données en OpenData est une option envisageable, l'architecture doit le prévoir, mais les contraintes fonctionnelles liées à la sécurité et/ou la charge serveur, classe cette orientation en option secondaire, raison pour laquelle nous l'avons mis en pointillés dans la figure 34.

Cette partie clôt la phase de conception. Nous allons maintenant nous orienter sur le développement de l'application dans la troisième et dernière partie de ce mémoire.

Chapitre 3 - Développement du projet

Ce dernier chapitre est consacré aux développements et aux résultats de l'application. Nous verrons les différentes technologies et bibliothèques mises en œuvre. Le choix de ces orientations ainsi que les langages informatiques utilisés. Les stratégies algorithmiques que nous avons mis en pratique afin d'obtenir une application fluide et rapide. Nous prendrons en considération la sécurité de l'application et sa capacité de monter en charge à travers différents tests. Enfin, nous concluons ce chapitre sur les résultats obtenus et nous ferons une rétrospection sur les améliorations que nous aurions pu apporter.

3.1 - Solution Node.JS

Le choix de nous orienter sur une application web s'explique par le côté portatif de l'application. En effet, le principal objectif est de proposer la solution au plus grand nombre, afin d'obtenir un nombre de résultats importants nous permettant de légitimer une cohérence. De surcroît, la solution se veut fluide par le nombre d'interactions prévues, notamment dans l'enregistrement interactif des émotions du candidat. Le développement client est donc une composante importante de l'application qui justifie des échanges façon « streaming » avec le serveur. Javascript est un langage particulièrement efficace dans ces domaines. Il existe de nombreuses bibliothèques JavaScript disponibles sur le marché et NodeJS en fait partie avec la particularité d'être un serveur en JS. NodeJS peut être utilisé pour construire différents types d'applications web, mais il est très spécialisé pour des applications interactives. Un développeur JavaScript peut travailler sur l'ensemble de l'application web au lieu de plusieurs développeurs se spécialisant sur le client (frontend) ou le serveur (backend). Le code JS est partagé entre le client et le serveur ce qui nous permet de réduire significativement les temps et coûts de développement de l'application. Les tâches courantes de DataTrainX, comme la lecture et l'écriture dans les connexions réseau, la lecture ou l'écriture dans la base de données, la lecture ou l'écriture dans le système de fichiers, peuvent être exécutées rapidement à l'aide de Node.js. Ce dernier utilise la version 8 du moteur

développé par Google qui compile le JavaScript en code machine natif ce qui améliore grandement le temps de réponse de l'application. Enfin, Node.js c'est aussi et surtout une communauté qui partage et s'échange des modules facilement réutilisables d'un projet à l'autre avec le gestionnaire de paquets NPM. Nous l'avons beaucoup utilisé pour DataTrainX, ce que nous détaillerons dans le prochain chapitre.

3.1.1 - NPM et paquets utilisés

NPM est le gestionnaire de paquets couramment utilisé avec Node.js. Il permet de gérer les différentes dépendances entre les modules parmi un dépôt qui en comporte presque 50000. Il permet d'intégrer une liste de bibliothèques / composants dans le projet, en respectant la contrainte de version définie. Cela nous permet de réutiliser des codes et de les mettre à jour avec une extrême facilité. Nous n'avons pas prévu de proposer DataTrainX en paquet NPM public, nous aurions pu, car notre projet suit convenablement les configurations prérequis liées au fichier package.json. Cependant, le projet étant disponible publiquement sur GitHub cela nous semblait suffisant. Dans le tableau 9 ci-dessous, nous indiquons les paquets utilisés dans notre projet ainsi qu'une description sommaire. Nous travaillons à deux niveaux, des paquets sont utilisés uniquement en mode développement, d'autres pour la production.

Tableau 9 – Composants utilisés en frontend avec NodeJS pour DataTrainX

NodeJS v16.13.1	Description
Développement	
rollup v2.72.0 @rollup/plugin-commonjs v22.0.0 @rollup/plugin-json v4.1.0 @rollup/plugin-node-resolve v13.3.0 @rollup/plugin-replace v2.3.2 rollup-plugin-svelte v7.1.0 rollup-plugin-terser v5.1.3 rollup-plugin-css-only v3.1.0 tslib v2.3.1 svelte v3.48.0	<p>Rollup est un « bundler » Javascript il compile l'ensemble des modules importés en un seul fichier unique bundle.js et bundle.css. Des plugins complémentaires sont utilisés pour la minification (forme d'obfuscation afin de réduire les temps de chargement) et la prise en charge des dépendances en utilisant un ensemble de normes pour unifier ECMAScript ou TypeScript (tslib).</p> <p>La configuration de ces éléments se fait dans le fichier racine rollup.config.js.</p> <p>Svelte est un composant important dans notre développement. C'est une bibliothèque JavaScript que nous utilisons pour la réalisation de nos interfaces.</p>

Production	
<p> @tensorflow/tfjs v3.15.0 @tensorflow/tfjs-node v3.16.0 Canvas v2.9.1 chart.js v3.7.1 chartjs-adaptor-luxon v1.1.0 chartjs-plugin-streaming v2.0.0 node-fetch v3.2.3 sirv-cli v2.0.2 </p> <p><i>Paquage mineur</i></p> <p> svelte-scrollto v0.2.0 svelte-json-tree v1.0.0 lz-string v1.4.4 </p>	<p>En production, nous avons fait le choix d'utiliser TensorflowJS en configuration browser. Tensorflow n'est donc pas géré en backend et compilé par NodeJS, nous expliquerons ce choix stratégique dans la section suivante. Canvas est utilisé pour manipuler la balise html canvas, notamment pour la capture vidéo dans le cadre de la REF ainsi qu'avec chartjs pour la représentation graphique des résultats ou du streaming des émotions en complément de FaceApi et node-fetch. Sirv-cli est un paquet pour gérer le démarrage de notre serveur (dev vs prod), c'est notamment ici qu'on gère le port, le protocole https et le certificat SSL généré par Let's Encrypt.</p>
<pre>"workspaces": ["packages\\datatrainx"]</pre> <p> Face-api v0.22.2 CnnExplainer build b9c1759 </p>	<p>Nous avons créé un workspace spécifique pour inclure deux projets importants que nous avons adaptés, FaceApi et CnnExplainer. Ceci en effet pour garder la main sur la customisation de ces deux paquets et éviter leurs mises à jour sur la branche officielle. Nous détaillons le pourquoi dans un chapitre dédié plus bas.</p>

Nous détaillons plus spécifiquement certains paquets considérés comme majeurs pour notre développement, dans les prochains chapitres.

3.1.2 - TensorflowJS

Nous avons utilisé TensorflowJS 3.18 car nous sommes dans un environnement Javascript, contrairement à la bibliothèque principale que nous avons utilisée avec Python. Dans cette configuration nous avons plusieurs possibilités : utiliser Tensorflow en backend et le compiler avec NodeJs ou utiliser la bibliothèque compilée disponible sur un serveur distant (balise script). Nous avons testé les deux options. La première approche a un intérêt réel si nous souhaitons utiliser l'accélération matérielle disponible de la machine, comme CUDA, chose

que nous avons réalisée lors de la construction des modèles FER2013. Par exemple, pour notre applicatif, nous aurions pu récupérer l'image de la webcam, sous-traiter sur notre serveur la reconnaissance du visage et l'analyse de l'émotion, puis renvoyer le résultat au client. C'était d'ailleurs notre première approche, mais les contraintes d'upload de notre serveur sur un réseau domestique nous ont orientées vers une autre solution. En effet, nous devons limiter les retours serveur, les temps de réponse étaient trop lents pour une application qui se devait de répondre « façon streaming ». Nous avons donc fait le choix de tensorflowJS exécuté côté client, avec l'option que ce dernier soit hébergé sur un serveur dédié. C'est ce que nous exprimons par le code 7 de la balise script ci-dessous de fichier html.

```
<script src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs@3.18.0/dist/tf.min.js" integrity="sha256-Yl8X8Ug0tUNf21v4ls+0vnBjPIWv3xbDs9Y0zE+9HwM=" crossorigin="anonymous"></script>
```

Code 7 – Chargement de la bibliothèque TensorflowJS en balise script

Cela veut dire que ce sont les caractéristiques machine du client qui analysent le visage puis l'émotion. Notre application marchera donc plus lentement ou de manière très fluide selon la configuration machine de la personne qui testera l'application. Nous pourrions cependant revenir à la première option assez facilement si nous trouvons, par la suite, un hébergement de l'application plus solide qu'actuellement. À titre d'information nous mettons ici en figure 35 les performances du serveur réalisées sur nperf.com :

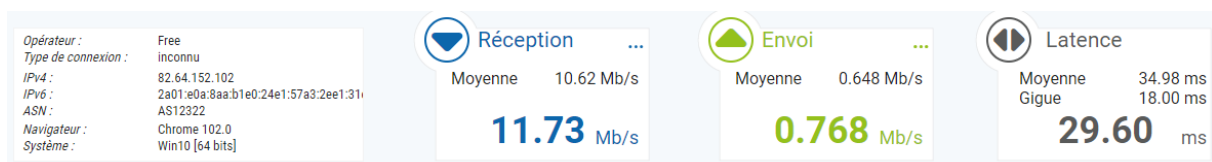


Figure 35 – Performances en upload de 0.7 Mb/s du serveur, test nperf.com

On est très bas par rapport à un Datacenter. L'avantage c'est que cela offre des conditions idéales pour chercher les optimisations.

C'est donc dans la configuration script que l'on utilisera TensorflowJS pour la reconnaissance du visage et des émotions avec FaceApi ou lors du chargement du modèle Fer2013 avec CnnExplainer.

3.1.3 - Svelte, routage et moteur de template

Svelte [73] est une bibliothèque frontend JavaScript mais à la différence des dernières bibliothèques du même type (jQuery, Angular, React), où la plupart du travail est directement exécuté au niveau du navigateur Web, Svelte est compilé en backoffice afin de générer la logique de manipulation du DOM. Cette particularité offre l'avantage de performances accrues, car il y a moins de code à exécuter lors d'un changement d'état au sein d'un composant. Mais aussi, elle minimise le poids du code final qui dépend du nombre de fonctionnalités utilisées dans le code. Par exemple, nos templates sont gérés par svelte, sous la forme de trois zones de code (JS, CSS, HTML). Si ce composant n'est pas appelé, il n'y a pas de surcharge et de code inutilisé côté client. Pour réaliser le chargement de nos templates, nous avons utilisé ce principe de composant « Slot ». Un composant est une brique logique isolée dans le but d'être réutilisée. C'est ce qu'on appelle aussi l'état d'un composant. La notion d'état est implicite comme pour les propriétés, l'état est donc une propriété susceptible de changer à l'intérieur du composant. Le changement de valeur d'une propriété provoque la mise à jour du composant. Il existe le slot balise `<slot />` qui est un container et qui joue le rôle d'une propriété contenant les enfants d'un composant donné. Dans notre fichier `Layout.svelte` nous utilisons ce principe par cet appel avec le fichier parent `Datatrains.svelte` que nous représentons dans le code 8 ci-dessous :

Datatrains.svelte

```
<main>
  {#if viewportComponent == views[layoutValue]}
  <section id='main' class="" on:outroend={updateViewportComponent} transition:fly={{ y: 200, duration: 1000 }}>
    <Layout>
      <svelte:component this={viewportComponent}/>
    </Layout>
  </section>
  {/if}
```

Layout.svelte

```
<div class="layout start jumbotron text-center hero is-fullheight-with-navbar">
  <slot></slot>
</div>
```

La transition permet un effet visuel au changement de composant. Le `<slot>` indique la zone réservée.

Code 8 – Utilisation de svelte et du slot pour le chargement dynamique des pages

La variable `layoutValue` est gérée dans le « store » de svelte qui est un objet avec une méthode d'abonnement et qui permet aux composants abonnés d'être avertis chaque fois que la valeur du magasin change. Dans un autre template nous changeons cette valeur et par abonnement le template `Layout.svelte` change et prend la valeur du composant appelé, c'est ce que nous

indiquons dans le code 9 ci-dessous sur le click du logo. Header.svelte qui charge le template Presentation.svelte.

```
<header id="header">
  <div id="logo">
    <div id="logo-text" on:click={() => layoutStore.setLayout("presentation")}>
      DataTrainX
    </div>
  </div>
```

Utilisation du magasin, ici paramétrage de la vue.

Code 9 – Principe de chargement des templates sur DataTrainX avec le slot de Svelte

Le magasin « store.js » centralise le setLayout sous cette forme, code 10 ci-dessous :

```
function createLayout() {
  const { subscribe, set } = writable( value: "presentation");
  return {
    subscribe,
    setLayout: (view) => set(view)
  };
}
export const layoutStore = createLayout();
```

Un composant abonné, reçoit l'information et charge le composant « presentation ». Ce que nous faisons avec notre fonction createLayout() donnant accès au paramétrage setLayout().

Code 10 – Magasin "store" de svelte permettant aux composants abonnés d'être avertis chaque fois que la valeur du magasin change

Nous avons utilisé d'autres fonctionnalités sur Svelte, mais cette démonstration nous paraissait assez complète sur son utilisation dans les grandes lignes pour ne pas approfondir le reste.

3.1.4 - Chart.js

Chart.js [78] est une bibliothèque JavaScript open source gratuite pour la visualisation de données. Nous l'utilisons à deux endroits. D'une part pour présenter l'enregistrement des données façon streaming (figure 36) de le REF et d'autre part sur la restitution des résultats pour afficher le profil d'un apprenant.

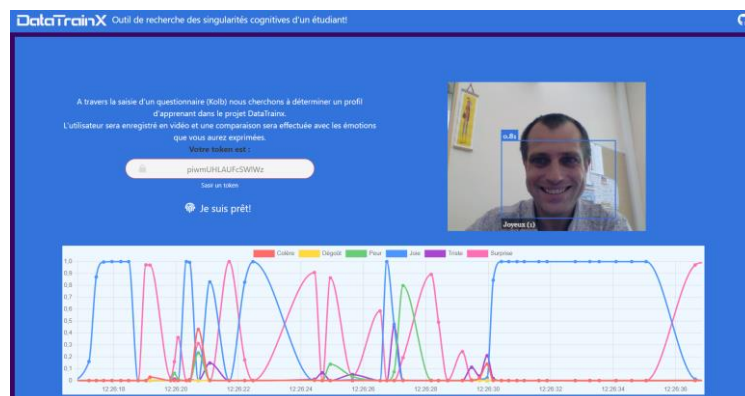


Figure 36 – Chart.js utilisé en mode streaming pour afficher les émotions détectées sur un graphique

Chart.js est utilisé avec le plugin complémentaire chartjs-plugin-streaming pour l'animation et la mise à jour interactives des informations de la REF. La configuration et l'affichage de ces données sont centralisés sur le fichier ChartStream.svelte. Les dataset sont stockées et ajoutées dans un tableau JavaScript présent dans le fichier data.js. Ils sont mis à jour via le fichier Faceapi.svelte lorsque les émotions sont trouvées. Ces données sont utilisées à différents endroits de notre application, ici par exemple dans l'affichage d'un diagramme. Une représentation de cette opération est visible dans le code 11 ci-dessous :

data.js (service factory)

```
export let dataExpression = {
  colere:[],
  degout:[],
  peur:[],
  joie:[],
  triste:[],
  surprise:[]
};
```

Nous manipulons des données préformatées, celles-ci sont regroupées dans un dossier appelé « service factory »

Faceapi.svelte

```
const result = await faceapi.detectSingleFace(videoSource, options).withFaceExpressions();

if (result) {
  const now = Date.now();
  dataExpression.peur.push({
    x: now,
    y: Object.values(result)[1]["Peur"],
  });
}
```

Ce paterne nous permet de modéliser l'insertion de nos données.

ChartStream.svelte

```
const dataChart = new Chart(ctx, {
  type: "Line",
  data: data,
  options: {
    scales: {
      x: {
        type: "realtime",
        realtime: {
          duration: 20000,
          refresh: 1000,
          delay: 2000,
          //onRefresh: onRefresh,
        },
      },
    },
  },
  label: "Peur",
  backgroundColor: "#6BCB77",
  borderColor: "#6BCB77",
  cubicInterpolationMode: "monotone",
  data: dataExpression.peur,
});
```

Code 11 – Mise à jour interactive du graphique sur la REF

3.1.5 - FaceApi

FaceApi est un projet open source réalisé par Vincent Mühler et une vingtaine de contributeurs en 2020. C'est une API de reconnaissance faciale JavaScript pour le navigateur et NodeJS, implémentés à partir du noyau tensorflowJS. C'est ce projet que nous utilisons pour reconnaître les émotions à partir de la webcam. Pour ce faire, nous chargeons deux modèles. Le premier sert à reconnaître le visage, basé sur la bibliothèque OpenCV et les éléments pseudo-Harr (chapitre 1.4.2), ce fichier est tiny_face_detector_model-shard1, il est très léger

et ne fait que 189 ko. Le détecteur de visage a été entraîné sur un ensemble de données personnalisées d'environ 14 000 images étiquetées avec des boîtes englobantes. De plus, le modèle a été formé pour prédire les boîtes englobantes qui couvrent entièrement les points des caractéristiques faciales ainsi que des unités d'action (AUs) (chapitre 1.5.2). Le deuxième modèle chargé est `face_expression_model-shard1`, il fait 322 ko et il est dédié à la reconnaissance de l'émotion. Il est beaucoup plus efficace que le modèle que nous avons entraîné avec FER2013, raison pour laquelle nous l'utilisons. Nous avons fait le choix de créer un paquet spécifique pour l'intégration de FaceApi. En effet ce dernier n'était pas compatible avec nos différents modules. Les versions de Tensorflow utilisées étaient par exemple obsolètes. Nous centralisons l'utilisation de cette API ainsi que la webcam sur Faceapi.svelte. L'ensemble est géré en trois parties. Une première partie frontend qui gère l'affichage, une deuxième partie « videoCam » qui lance la vidéo et enfin une troisième partie qui détecte le visage et les émotions. L'ensemble est géré avec le magasin de svelte pour communiquer avec les autres templates.

3.1.6 - CnnExplainer

CnnExplainer est un projet de Zijie Wang et al. [79] dans le cadre d'une collaboration de recherche entre Georgia Tech et l'État de l'Oregon. Ce projet a pour but de rendre accessible la compréhension de l'application de l'apprentissage en profondeur par le développement d'un outil de visualisation interactif conçu pour les non-experts. CnnExplainer permet d'examiner le principe d'un réseau de neurones convolutif (CNN), grâce à des transitions fluides entre les niveaux d'abstraction. L'outil permet aux utilisateurs d'inspecter l'interaction entre les opérations mathématiques de bas niveau et les structures de modèle de haut niveau.

Nous avons souhaité intégrer CnnExplainer au projet DataTrainX d'un point de vue pédagogique afin de faciliter la compréhension de l'application. Nous y avons apporté quelques modifications pour que ce dernier soit compatible avec notre modèle d'entraînement TinyVGG généré sur FER2013. Il est léger, ne fait que 51 ko et il respecte l'architecture attendue de CnnExplainer. Néanmoins, cela a impliqué notamment de passer l'ensemble de l'application sur l'analyse d'un seul canal « gray » au lieu de RGB et d'adapter la configuration sur un nombre de neurones de sortie calibré sur un modèle de reconnaissance des expressions. Nous avons poussé notre contribution sur le GitHub du projet, ce qui nous a

valu quelques félicitations de l’auteur. Nous centralisons l’utilisation de CnnExplainer sur notre fichier Cnn.svelte. CnnExplainer permet de visualiser le fonctionnement d’un réseau de convolution de manière interactive. Comme l’illustre la figure 37 chaque étape de convolution est cliquable et permet l’accès à une fenêtre explicative en 5 étapes pour arriver aux résultats du modèle (Convolution, ReLu, Pooling, Flatten, Output).

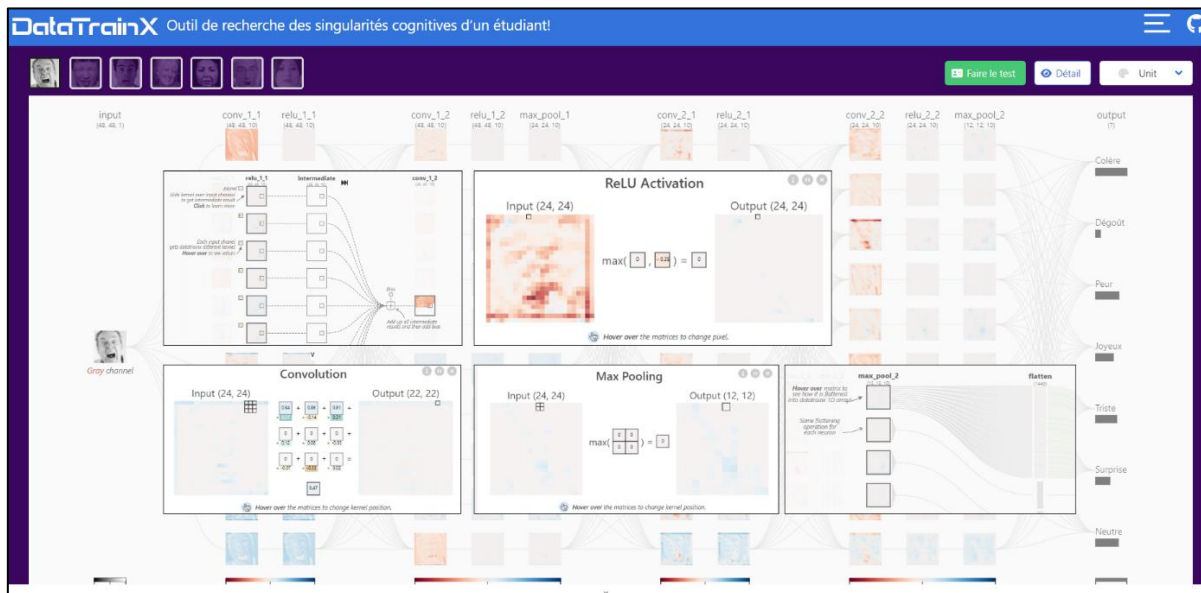


Figure 37 – Principe de CnnExplainer intégré à DataTrainX avec notre modèle TinyVgg FER2013

La démonstration se limite à ces 5 étapes. Le modèle TinyVGG est donc construit à partir de ces différentes étapes afin d’être exploité avec CnnExplainer. D’autres modèles plus complexes comme « deep » avec lesquels nous avons eu de meilleurs résultats ne peuvent pas être chargés sur CnnExplainer car il ne prend pas en considération d’autres étapes comme le BatchNormalisation() [80] ou le Dropout() [81]. Puis globalement, ici nous ne représentons que 4 étapes de convolution (conv1_1, conv1_2, conv2_1, conv2_2), le modèle « deep » par exemple en compte 6, ce qui aurait rendu l’affichage d’autant plus complexe.

Ceci clôt notre chapitre sur l’adaptation et l’intégration des différents modules sur notre projet. Dans le prochain chapitre, nous détaillerons les modules propres à notre application distincts en deux sections, le développement frontend et le développement backend.

3.2 - Développement Frontend

Dans ce chapitre nous détaillerons les principales fonctionnalités que nous avons développées. Cela comprend, l'interface en général, le chargement de la webcam, la saisie du questionnaire, la mise en situation émotionnelle ainsi que l'enregistrement des données au fur et à mesure du test.

3.2.1 - Organisation et structure des dossiers

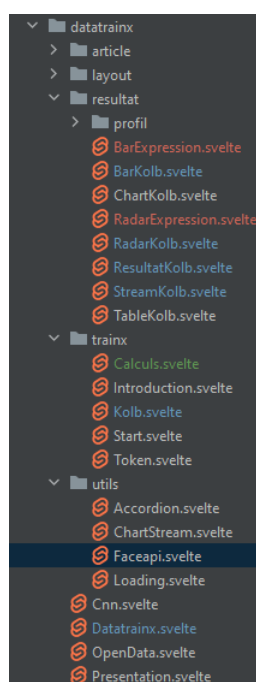


Figure 38 – Organisation du développement frontend en 7 dossiers

Le Frontend est la partie la plus significative de notre développement, elle concentre la majorité des fonctionnalités de notre application dont les principaux objectifs sont la fluidité, la fiabilité et le responsive. Pour cela, nous nous sommes appuyés sur le framework CSS Bulma et Svelte. Le découpage de notre code fut important afin que ce dernier puisse être le plus portable et réutilisable. Cela nous a orientés vers un découpage de 50 fichiers réparti en 8 dossiers que nous indiquons dans la figure 38 ci-contre. Le dossier « datatrainx » permet de bien différencier notre application frontend des autres éléments que nous avons intégrés, comme CnnExplainer, qui regroupe elle-même son organisation propre. Ce dossier principal regroupe les principaux contenus de l'application. Le dossier « layout » qui inclut le menu, le header, le footer, les principes de slot pour switch des contenues. Le dossier « utils » qui inclut des processus clés de notre applicatif et réutilisés à différents endroits, notamment le plus important qui est Faceapi.svelte. Le dossier « Trainx » est réservé à la partie mise en condition des utilisateurs (tests, questions, token, calculs etc.). Le dossier, résultat, regroupe les éléments de restitution des tests (graphiques, profil, score).

3.2.2 - IHM et loader

DataTrainX étant une application node.js, le principe est de charger un unique fichier JavaScript appelé bundle.js regroupant l'ensemble des bibliothèques et composants pour qu'ensuite l'application soit uniquement dépendante du client. Cela permet d'avoir une application interactive et fluide, mais avec l'inconvénient d'avoir un temps de chargement

plus long au lancement de l'application. En mode production, c'est-à-dire après l'utilitaire terser de minification du fichier, ce dernier fait 645 ko. Si l'on rajoute TensorflowJS on arrive déjà à 1Mo avec juste le chargement de ces deux fichiers, ce qui est relativement beaucoup pour une page web. Le reste de la page, quelques images, des icônes, du CSS on arrive à un total de 1,7 Mo pour un temps d'affichage, en corrélation à l'upload du serveur, d'une dizaine de secondes. Les statistiques moyennes d'attente d'un internaute devant une application web sont de 3 secondes, avant qu'il parte. Par ailleurs à ce stade nous ne chargeons toujours pas les deux modèles entraînés de reconnaissance du visage et des expressions qui font respectivement, 5.3Mo et 330 ko. Malgré ces mauvais paramètres, nous n'avons pas un score si catastrophique sur PageSpeed Insights de google [82], voir figure 39 ci-dessous :



Figure 39 – Test de performance de l'affichage de DatatrainX sur PageSpeed de google

Nous améliorions notre temps d'attente et d'interactivité par l'affichage d'une page de chargement en CSS animé (keyframe) et html. Ce temps de chargement est accompagné de quelques lignes en JavaScript qui injecte dans le dom l'appel de Bundle.js et détermine la suppression de l'élément html/CSS loader. Le code Javascript crée une balise de script HTML d'appel du fichier bundle.js, ce qui nous permet de savoir quand ce dernier est chargé. En attendant, nous avons une page d'animation très légère que peut exécuter le client et qui améliore significativement notre score, figure 39.

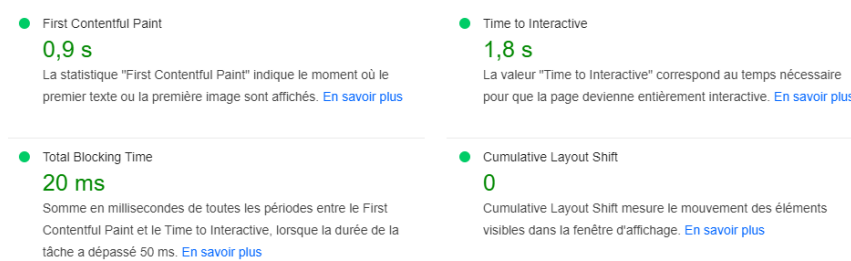


Figure 40 – Score PageSpeed en augmentation grâce à un loader et un chargement déporté de Bundle.js

Le loader html/CSS montre une animation sur le logo DataTrainX, qui lui-même utilise la balise svg. L'ensemble est très léger et se charge en moins d'une seconde (First Contenu Paint). Nous

avons aussi un pourcentage de chargement, qui est calé sur une dizaine de secondes avec un timeout. Le fond change aussi de couleurs, l'ensemble nous donne un CLS (Cumulative Layout Shift) à zéro qui est la meilleure note indiquant une métrique centrée utilisateur qui permet d'évaluer la stabilité visuelle d'une page web. La fin du chargement du bundle permet de découvrir l'IHM de l'application telle que désignée en partie conception.

3.2.3 - Header, Menu

Le header et le menu, sont gérés en partie CSS, « fixed » permettant à ce dernier d'être toujours disponible même si l'on scrolle l'application vers le bas. Le logo présent sur le header, comme d'ailleurs tous les logos présents sur l'application (loader, header, présentation, footer) sont gérés en balise svg et centralisée sur un unique fichier, code 12. Ceci pour alléger le code en upload, nous n'avons pas de fontes ou d'images spécifiques à télécharger.

```
<svg width="180.3" height="22.5" viewBox="0 0 180.3 22.5" xmlns="http://www.w3.org/2000/svg">
  <g id="svgGroup" stroke-linecap="round" fill-rule="evenodd" font-size="9pt" stroke="#fff" stroke-width="0" fill="#{color}" style="...">
    <path d="M 0 22.5 L 0 7.74 L 4.38 7.74 L 4.38 18.51 L 17.64 18.51 Q 18.69 18.51 19.065 17.85 A 2.666 2.666 0 0 0 19.334 17.152 Q 19.44
  </g>
</svg>
```

Code 12 – fichier Logo.svelte, appelé à différent endroit de l'application. La couleur est paramétrable.

Le header centralise aussi un menu, permettant l'accessibilité à l'ensemble des pages que compose l'application, figure 41.

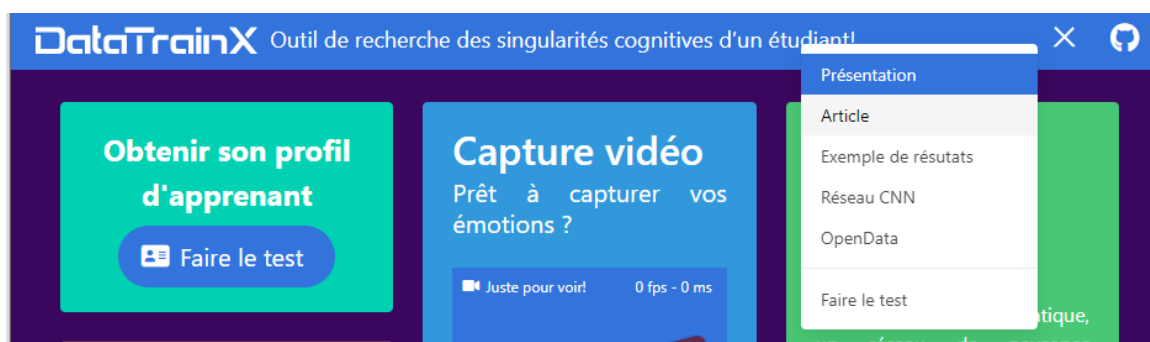


Figure 41 – Menu DataTrainX. Balise "active" qui indique où l'on se trouve dans l'application

Ce menu est découpé en différents fichiers « svelte », gère les effets SVG type « hamburger », indique la page courante sur fond bleu, apparaît et disparaît avec un effet visuel de type « fly, fade ».

Ce menu permet le switch de la page principale avec les autres contenus du site sur le principe du <slot> expliqué précédemment.

3.2.4 - Capture Webcam

Sur le portail « présentation » nous proposons une capture vidéo afin de « tester » l'application sans pour autant se lancer dans le test complet. Cet applicatif est centralisé sur le fichier *Faceapi.svelte*, le même applicatif que nous utiliserons lors de la phase de tests utilisateurs. C'est l'un des fichiers que nous avons le plus travaillé, il est découpé en trois parties (script, style et html) comme d'ailleurs la majorité de nos fichiers svelte. La partie script fait 240 lignes de code et utilise 7 bibliothèques que nous détaillons dans le tableau 10 ci-dessous :

Tableau 10 – Bibliothèques utilisées pour la capture webcam

Bibliothèque et méthodes utilisées	Description
<pre>import * as faceapi from 'datatrainX' faceapi.SsdMobilenetv1Options() faceapi.nets.ssdMobilenetv1.load() faceapi.loadFaceExpressionModel() faceapi.detectSingleFace().withFaceExpressions() faceapi.matchDimensions() faceapi.resizeResults() faceapi.draw.drawFaceExpressions()</pre>	Chargement des modèles de reconnaissance des visages et des expressions. Résultats de détection du visage et des expressions. Affichage des résultats sur canvas.
<pre>import {layoutStore, videoStore, timeStore, fpsStore, loadingStore, infoLoadStore} from ".././stores";</pre>	Chargement, souscription et setting des variables utilisées dans d'autres composants
<pre>import {FaceExpression, FaceDetection, streamExpression} from ".././service-factory/data"; import {updatePush} from ".././service-factory/update"; import LZString from 'lz-string';</pre>	Variables et fonctions de mises à jour en lien avec MongoDB. Compressions des images. Enregistrement des données.
<pre>import {fade} from 'svelte/transition'; import Loading from './Loading.svelte';</pre>	Composant frontend. Affichage d'un loader selon conditions et effets de transition.

Le script comporte 7 fonctions qui gèrent l'ensemble du processus. La fonction asynchrone **playCam()** qui charge les deux modèles, *ssdMobilenetv1* (5,3 mo) pour la reconnaissance du visage et *face_expression* (300 ko) pour la reconnaissance des expressions. Initialement nous avons utilisé un modèle beaucoup plus léger pour la reconnaissance du visage *tiny_face_detector* (190 ko), mais ce dernier n'était pas assez efficace pour reconnaître un visage rapidement, les imprécisions étaient trop importantes dans une configuration type streaming. Nous avons donc privilégié un chargement initial plus long pour des résultats plus rapides et plus précis ensuite, d'où la nécessité d'avoir mis en place un loader représenté en figure 42. *PlayCam()* lance ensuite la webcam et attend la détection du visage et de la reconnaissance des expressions. Vient ensuite le lancement en boucle avec un *setTimeout* de

la fonction **onPlay()**. Cette dernière lance les méthodes de reconnaissance du visage puis des émotions. Nous exprimons ces résultats par une étape intermédiaire d'affichage du temps pour avoir déterminé le visage et l'émotion en milliseconde et le nombre d'images par seconde (fps) traité dans ce laps de temps, ce que fait la fonction **updateTimeStats()**.



Figure 42 – Les trois étapes du loader sur la webcam. Les deux premières se font au chargement des modèles, la troisième se lance si le visage et l'expression ne sont pas détectés.

Lorsque nous avons les résultats, nous les intégrons dans un tableau qui sera repris par un graphique de type streaming puis nous faisons appel à la fonction **takePicture()**. **takePicture()**, est chargé de dessiner sur un canvas html le rendu du visage et les informations d'expressions, c'est en effet cette dernière que nous affichons à l'écran et non la vidéo de la webcam. À cette étape nous avons toutes les données disponibles sur l'utilisateur que nous enregistrons sur mongoDB, ce que nous faisons à la cinquième étape avec la fonction **pushData()**, figure 43.

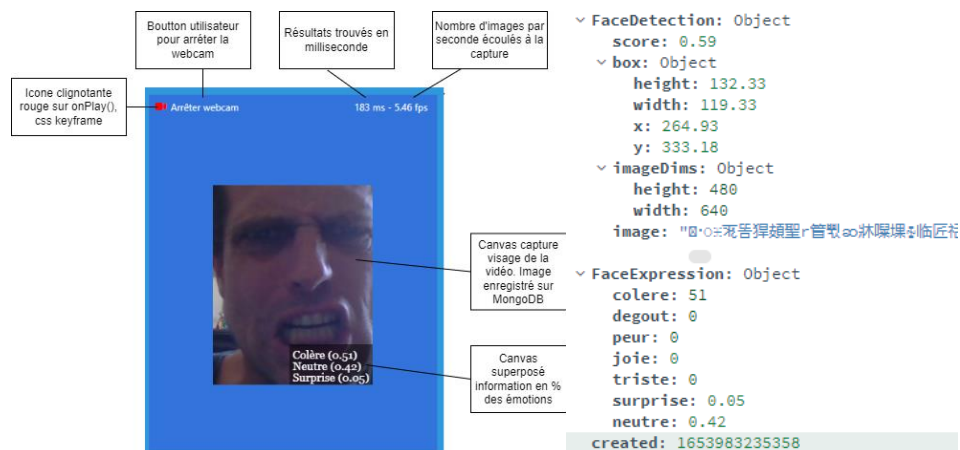


Figure 43 – Description de l'interface capture webcam et l'équivalent data envoyée par **pushData()**

Nous émettons néanmoins quelques conditions pour l'enregistrement afin d'éviter une saturation du serveur backend, il faut qu'une expression différente de « neutre » et supérieure à 20% soit émise. Les deux dernières fonctions sont **stateVideo()** et **stopCam()**. La première permet de déterminer si la webcam est active et de lancer selon les conditions

playCam() ou stopCam() qui comme son nom l'indique met fin à la webcam et au setTimeout de la fonction onPlay(). Cette dernière option se lance par choix utilisateur (arrêter webcam) ou à la détection d'un changement de page provoqué par l'utilisateur ou l'application.

3.2.5 - Test de Kolb

Le test est le cœur de l'application, c'est ici que nous mettons en condition l'utilisateur afin de capter ses émotions en fonction de différents tests qu'il doit réaliser. Cet applicatif fait intervenir 5 fichiers de type svelte que nous retrouvons sur l'interface en figure 44 ci-dessous avec une description.



Figure 44 – Interface de la phase de test découpée en trois zones

Le fichier Token.svelte crée le premier enregistrement sur MongoDB, c'est ce qui nous permet de récupérer l'ID de MongoDB que nous affichons en Token pour l'utilisateur. Ce dernier à la possibilité de le garder pour éventuellement le réutiliser par la suite pour ne pas redémarrer de nouveaux tests, mais les continuer ou revoir ses résultats si ceux-ci sont terminés. Il faut savoir que dès que le token est attribué, les résultats s'enregistrent sur MongoDB de manière continue comme l'illustre la figure 45.

```

/trainer/dataExpression/629765575ccdc296b7279f0 200 12.579 ms - 40
/trainer/dataExpression/629765575ccdc296b7279f0 200 16.091 ms - 40
/trainer/dataExpression/629765575ccdc296b7279f0 200 14.057 ms - 40
/trainer/dataExpression/629765575ccdc296b7279f0 200 17.841 ms - 40
/trainer/dataExpression/629765575ccdc296b7279f0 200 21.709 ms - 40
/trainer/dataExpression/629765575ccdc296b7279f0 200 19.931 ms - 40
/trainer/dataExpression/629765575ccdc296b7279f0 200 16.717 ms - 40
/trainer/dataExpression/629765575ccdc296b7279f0 200 20.633 ms - 40
/trainer/dataExpression/629765575ccdc296b7279f0 200 20.979 ms - 40
/trainer/dataExpression/629765575ccdc296b7279f0 200 18.363 ms - 40

```

629765575ccdc296b7279f0	200	fetch	update:js:15	307 B	28 ms
629765575ccdc296b7279f0	200	fetch	update:js:15	307 B	26 ms
629765575ccdc296b7279f0	200	fetch	update:js:15	307 B	31 ms
629765575ccdc296b7279f0	204	preflight	Preflight	0 B	9 ms
629765575ccdc296b7279f0	200	fetch	update:js:15	307 B	39 ms
629765575ccdc296b7279f0	200	fetch	update:js:15	307 B	32 ms
629765575ccdc296b7279f0	200	fetch	update:js:15	307 B	28 ms
629765575ccdc296b7279f0	200	fetch	update:js:15	307 B	34 ms
629765575ccdc296b7279f0	200	fetch	update:js:15	307 B	32 ms
629765575ccdc296b7279f0	200	fetch	update:js:15	307 B	32 ms

Figure 45 – À gauche les logs MongoDB, à droite les requêtes network du client

Seuls l'arrêt de la webcam ou le changement de page autre que le test peuvent arrêter l'enregistrement. Sinon le flux est continu comme on peut le constater sur les logs de

MongoDB où les requêtes prennent un temps de réponse de 30 ms en retour. Si l'utilisateur arrête sa webcam ou l'obstrue, la reconnaissance faciale et donc des émotions ne se fait plus, ce qui fait basculer le loader « Détection du visage » ainsi que l'obstruction des boutons ne permettant pas la poursuite des tests, figure 46.

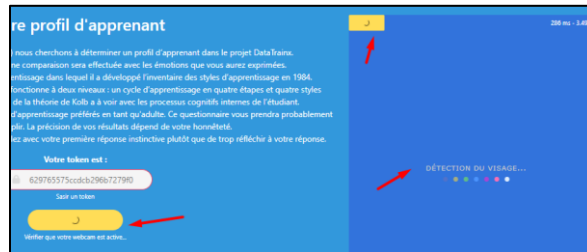


Figure 46 – L'obstruction de la webcam ou son arrêt ne permet pas à l'utilisateur de poursuivre les tests

Cette opération est gérée par le store de Svelte que nous avons détaillé plus haut. Enfin, le bouton « je suis prêt » permet la bascule slot de Token.svelte à Kolb.svelte qui permet de gérer le questionnaire. Le test de Kolb est composé de 80 questions avec deux réponses possibles, « Je suis d'accord » ou « Je ne suis pas d'accord » nous avons repris le questionnaire adapter de Alain Chapman [83] qui nous semblait le plus pertinent au vu du nombre de questions et de l'exploitation des résultats. Nous avons intégré les questions avec leur référence et leur typologie dans un fichier kolb2.js au format JSON. Cela nous donne un objet facilement exploitable par la suite et donc directement intégré au client. Le fichier Kolb.svelte importe cette variable objet questions afin de l'afficher à l'utilisateur. Le composant est le même pour les 80 questions, c'est toujours le magasin de svelte qui assure le rafraîchissement de la page avec divers effets pour passer d'une question à l'autre. Des contrôles sont réalisés si l'utilisateur ne coche rien, il ne peut pas passer à la question suivante. Chaque validation de question entraîne un appel backend sur le même principe que Faceapi.svelte pour l'envoi des émotions, ici nous envoyons les réponses saisies par l'utilisateur avec l'appel de la fonction updatePush, code 13.

```

updatePush('dataCondition', {
  ref:question,
  dataProfil:{
    [type]:parseInt(checked)
  },
  created:Date.now()
});

```

```

dataCondition: Array
  0: Object
    ref: 0
    dataProfil: Object
      theorist: 0
      date: 1653984250768

```

Code 13 – Phase intermédiaire d'envoi des données sur mongoDB avec appel de la fonction updatePush()

updatePush() est une fonction frontend dans un ensemble de « service factory » qui centralise les appels de type REST avec le backend. Nous reprendrons ces notions dans le chapitre suivant dédié au Backend. La fin du questionnaire permet d'afficher le résultat du profil de l'apprenant et une représentation visuelle des données que nous avons enregistrées pour tous utilisateurs. Cela fait l'objet de deux pages frontend, Resultats.svelte et OpenData.svelte, nous proposons un chapitre dédié plus bas à ce sujet sur l'exploitation des données en général, nous reprendrons donc ces sujets dans la section 3.4 Data Mining et classification des données de ce mémoire.

3.2.6 - Responsive

Enfin nous souhaitons terminer ce chapitre par notre volonté que toutes nos pages soient compatibles format mobile ou tablette. Suite aux différents tests de l'application, il paraissait évident qu'il y avait un intérêt de passer les tests du questionnaire dans un bus ou un canapé. Certes actuellement, nous serions techniquement limités si plusieurs personnes passent le test en même temps, les réponses serveur ne suivraient pas au vu de notre configuration réseau actuelle, cependant nous avons souhaité l'envisagé. Pour cela, nous nous sommes appuyés sur le framework CSS Bulma et de notre volonté d'avoir un développement « mobile first ». Nous proposons ci-dessous en figure 47 quelques visuelles de l'application au format mobile.

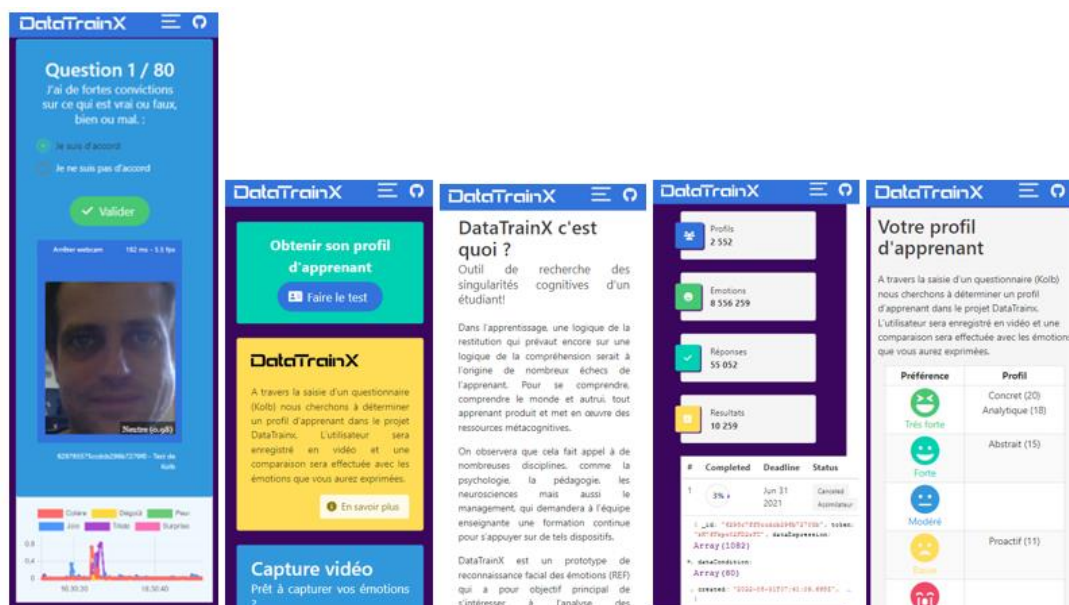


Figure 47 – Format mobile, page "Tests", "Présentation", "Article", "OpenData", "Résultats"

3.3 - Développement Backend

Dans ce chapitre nous détaillerons le principe d'échanges des données entre le frontend et le backend développé sur un principe d'échange de type REST. Ce dernier assure les échanges avec la base de données MongoDB. Nous verrons le principe d'une architecture MVC et le système de routage mis en place.

3.3.1 - Bibliothèques utilisées

Comparativement au développement frontend, le développement Backend est moins significatif en termes de temps. Comme déjà expliqués, nous aurions pu traiter ici la reconnaissance de l'image et de l'émotion, mais finalement pour d'autres raisons stratégiques nous cantonnons le développement backend en gardien des échanges avec la base de données. Bien que disponible dans le même projet, c'est donc une nouvelle application node.js que nous avons développée. Comme sur le frontend, bien que plus minimaliste, elle dispose de sa configuration propre. Nous détaillons ci-dessous les bibliothèques que nous utilisons dans le tableau 11 :

Tableau 11 – Composants utilisés en backend avec NodeJS pour DataTrainX

NodeJS v16.13.1	Description
Développement	
nodemon v2.0.16	Permet de relancer le serveur en cas de modification des fichiers backend.
Production	
express v4.18.1 body-parser v1.20.0 cors v2.8.5 jsonwebtoken 8.5.1 mongoose v6.3.3 morgan v1.10.0	Express est le framework standard pour node.js en version backend. Il nous permet de gérer le démarrage du serveur et le système de routage que nous utiliserons sous forme d'API REST. Dans cette optique nous utilisons body-parser pour lire les données contenues dans une requête http. Cors et jsonwebtoken assurent la sécurité des appels XHR et nous permettent de nous protéger des échanges avec l'extérieur. Mongoose, nous permet de simplifier les échanges avec mongoDB. Enfin morgan nous permet de gérer les logs serveur.

3.3.2 - API REST et Architecture MVC

Le principe d'une API REST (Representational State Transfer Application Program Interface) est de permettre aux logiciels de communiquer entre eux, le plus souvent on utilise des API REST pour créer des services web. En utilisant le protocole http le client peut demander des ressources avec un langage que le serveur comprend et le serveur renvoie la ressource avec un langage que le client accepte. Nous utiliserons un langage commun au format JSON pour l'envoi client et la réponse du serveur. Les appels http se feront en méthode POST pour la création d'un « trainer », en méthode PUT pour les mises à jour du « trainer » et en méthode GET pour récupérer le contenu d'un « trainer ». Les échanges se font selon le diagramme de séquence ci-dessous, en figure 48.

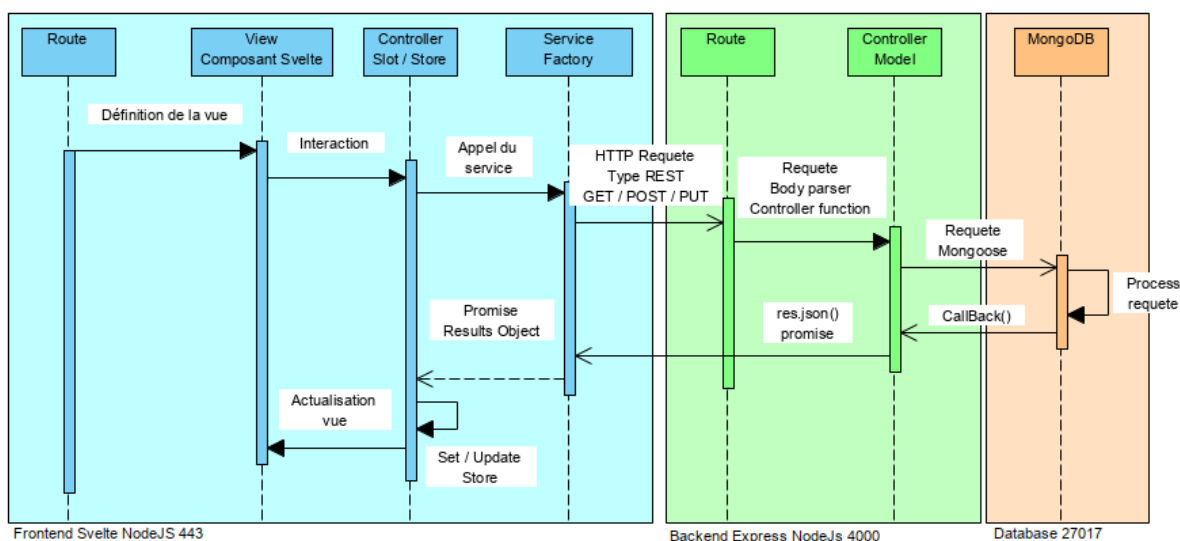


Figure 48 – Diagramme de séquence API REST

Nous utilisons un service factory coté frontend pour formaliser les données selon le modèle attendu coté backend. Les deux ressources ont donc des logiques similaires, en frontend le service factory reprend un modèle objet de base, en backend le modèle reprend un schéma plus précis avec Mongoose sur le typage des données avant d'assurer l'injection en base de données. L'application backend est donc relativement simplifiée, l'effort étant porté surtout sur le modèle attendu qui garantit la cohérence des données que nous injectons en base. La structure backend de nos fichiers suit la configuration ci-contre, en

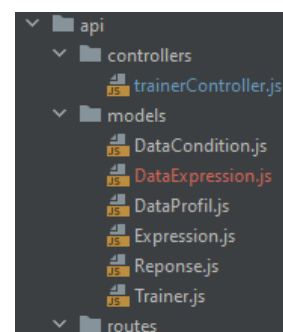


Figure 49 – Structure des dossiers coté backend de type API Rest

figure 49. Le système de routage contient 6 routes incluant la gestion de la sécurité sous forme JWT [84] que nous aborderons dans un prochain chapitre. Comme l'illustre le code 14 ci-dessous, nous avons une route de type POST pour l'enregistrement du trainer et la récupération de son Id. Trois routes de type GET en relation avec la récupération des données. Enfin, trois routes de type PUT, pour les mises à jour et insertion du trainer correspondant aux différentes étapes frontend.

```
router.post( path: "/", trainerController.assTrainer);
router.get( path: "/", trainerController.demoTrainer);
router.get( path: "/dataProfil/:trainerId", auth, trainerController.resultTrainer);
router.put( path: "/dataExpression/:trainerId", auth, trainerController.pushExpressionTrainer);
router.put( path: "/dataCondition/:trainerId", auth, trainerController.pushConditionTrainer);
router.put( path: "/dataProfil/:trainerId", auth, trainerController.updateProfilTrainer);
```

Code 14 – Système de routage Backend API REST

L'ensemble fait appel aux différentes méthodes du contrôleur. Les requêtes sont relativement simplifiées. Nous exposons par exemple le principe de push des données sur `pushExpressionTrainer` avec l'injection des données dans un tableau dans le code 15 ci-dessous. L'objet Trainer faisant référence à la construction du modèle.

```
exports.pushExpressionTrainer = async (req, res) => {
  try {
    const id = req.params.trainerId;
    await Trainer.findByIdAndUpdate(id, { $push: { dataExpression:req.body } });
    res.status(200).json({response:"pushExpressionTrainer, ok"});
  } catch (err) {
    res.status(500).json(err);
  }
};
```

Code 15 – Principe du contrôleur et envoi des données en base des données

Le frontEnd et le backend étant relié et les données injectées à cette étape, il nous reste plus qu'à les exploiter. C'est ce que nous verrons dans le prochain chapitre. Comment restituer les données à l'utilisateur afin d'obtenir la finalité de l'application à savoir obtenir un profil d'apprenant corrélé à ses expressions.

3.4 - Data Mining et classification des données

Dans ce chapitre nous nous focaliserons sur les données enregistrées. Cela comprend, la restitution du résultat à l'utilisateur, le panel représentatif et la fourniture des données en OpenData. Nous concluons sur l'analyse de ces données avec les différentes requêtes qui nous paraissent pertinentes.

3.4.1 - Les résultats d'un test

Au stade de ce mémoire notre application enregistre les données sur une durée estimée à 10 minutes, le temps de la saisie des 80 questions de Kolb. Nous avons à la base souhaité augmenter la mise en situation de l'utilisateur par une phase de casting et de résolution de puzzles. Par manque de temps, nous n'avons pas développé ces options, nous ferons donc une rétrospection sur nos données actuelles. Un utilisateur fournit sur la phase de test une moyenne de 400 expressions en 10 minutes, sachant qu'il n'y a pas vraiment de mise en situation et qu'une expression est un panel en pourcentage des 6 expressions de bases. A savoir que nous enregistrons une expression sous la forme d'un ensemble d'expressions, l'expression étant reconnue sur un panel de probabilités, par exemple, colère 10%, joie 80%, surprise 20%. Dans cette logique ce chiffre de 400 expressions peut contenir en réalité 2400 expressions. A contrario, si un utilisateur reste neutre pendant 10 minutes, il n'y aura strictement aucun enregistrement sur l'émotion car nous n'enregistrons pas l'émotion neutre dominante, hormis le fait qu'elle soit accompagnée d'une autre expression supérieure à 20%. De plus, nous avons les réponses aux 80 questions de Kolb, cela nous donne aussi un panel de concaténation de réponses relativement importantes. Nous avons donc, d'après ces données, cherché à obtenir une classification afin de rendre les résultats les plus compréhensibles possibles. Raison pour laquelle nous avons intitulé ce chapitre Data Mining, certes nous ne sommes pas dans des données de type Business Intelligence (BI), avec des tickets de caisses par millions non classifiés, le terme est un peu galvaudé, mais nous avons essayé au mieux de chercher une classification pertinente. Cette classification se fait par une phase de calcul à la fin des 80 questions nous permettant d'obtenir un dataProfil. Cette classification est composée d'un profil d'apprenant comportant 18 valeurs et d'un profil des émotions comportant 32 valeurs.

Nous reportons cette classification dans le tableau 12 ci-dessous :

Tableau 12 – Données dataProfil classifiées pour analyse

JSON	Exemple	Description
dataProfil	Object	Regroupe l'ensemble du profil
__apprenant	Object	Regroupe le profil apprenant
__dim1	Object	Regroupe les scores de dimension 1
__activist	Object	Regroupe le profil activist
__scoreRacine	13	(Int32) – Les points des questions « activist »
__score	5	(Int32) – Mise à l'échelle du score sur 1 (faible) à 5 (fort)
__tab	[0,80]	(Array) – Mise en forme du score type bar
__reflector	Object	Regroupe le profil reflector
__(...)	(idem activist)	(idem activist)
__theorist	Object	Regroupe le profil theorist
__(...)	(idem activist)	(idem activist)
__pragmatist	Object	Regroupe le profil pragmatist
__(...)	(idem activist)	(idem activist)
__dim2	Object	Regroupe les scores de dimension 2
__adaptateur	5	(Int32) – Moyenne score activist vs pragmatist
__divergeur	3	(Int32) – Moyenne score reflector vs theorist
__assimilateur	3	(Int32) – Moyenne score theorist vs reflector
__convergeur	4	(Int32) – Moyenne score pragmatist vs theorist
__profil	ADAPTATEUR	(String) – Profil dominant (peut être concaténé)
__score	4	(Int32) – Moyenne des scores (force du profil)
__expression	Object	Regroupe l'ensemble des expressions
__resultatExpressions	Object	Regroupe l'ensemble des resultats expressions
__colere	Object	Regroupe les résultats de colère
__count	16	(Int32) - Nombre des colères exprimés
__best	0.99	(Decimal) - Plus fort score en colère
__total	3.87	(Decimal) - Somme total des colères
__ratio	4	(Int32) - Ratio en % sur l'ensemble des émotions
__image	image/jpeg;base64	(String) - Image de la colère de score best
__degout	Object	Regroupe les résultats dégoût
__(...)	(idem colere)	(idem colere)
__peur	Object	Regroupe les résultats de dégoût
__(...)	(idem colere)	(idem colere)
__joie	Object	Regroupe les résultats de joie
__(...)	(idem colere)	(idem colere)
__triste	Object	Regroupe les résultats de triste
__(...)	(idem colere)	(idem colere)
__surprise	Object	Regroupe les résultats de surprise
__(...)	(idem colere)	(idem colere)
__score	16	(Int32) - Ratio en % des émotions par rapport à Neutre

Ces données sont donc classifiées, puis mises à jour sur le « trainer » à l'aide d'algorithmes qui sont regroupés dans le fichier Calculs.svelte, ce que nous détaillerons dans le prochain chapitre.

3.4.2 - Le calcul des données de classification

La phase de calcul est certainement le point le plus algorithmique de l'application. Nous procédons en 6 étapes pour classifier nos données, elles se déroulent après la saisie de la question 80 de Kolb où nous lançons l'affichage de `Calculs.svelte`. Cette dernière lance un loading tant que l'ensemble des calculs et mises à jour ne sont pas terminés. À la fin, elle propose un bouton « voir les résultats ». La première étape consiste à récupérer les données du « trainer » sur MongoDB via l'API REST, ensuite de lancer 4 fonctions de calculs, puis enfin de mettre à jour les données du « trainer » en 6^e et dernière étape. Les 3 premières fonctions sont dédiées à l'apprenant, seule la dernière fonction gère les expressions. La première fonction `preference()` est peut-être la plus complexe à appréhender, c'est elle qui détermine les différents scores en fonction des scores racines (activist, reflector, theorist, pragmatist) dont nous mettons en annexe 3 la grille de calcul sur laquelle nous nous sommes appuyés. Nous proposons de mettre en code 16 commenté de cette partie :

```
//On boucle les objets du profil dim1 récupéré sur la collection trainer
for (const [profil, value] of Object.entries(data.dataProfil.apprenant.dim1)) {
  try {
    //on se refere au serviceFactory de preferenceProfil
    Object.entries(preferenceProfil[profil]).forEach(([type, forceValue], index) => {
      if (value.scoreRacine >= forceValue) {
        //on se refere au serviceFactory de scoreTab
        dataProfil.apprenant.dim1[profil].tab = scoreTab[index].tab;
        score = scoreTab[index].score;
        totalScore += score;
        //stop la boucle en levant exception
        throw Break;
      }
    });
  } catch (e) {
    //on ajoute au serviceFactory dataProfil les valeurs de score
    dataProfil.apprenant.dim1[profil].scoreRacine = value.scoreRacine;
    dataProfil.apprenant.dim1[profil].score = score;
  }
}
//score global
dataProfil.apprenant.score = Math.round(totalScore / Object.keys(data.dataProfil.apprenant.dim1).length);
```

Code 16 – fonction `preference()` calcul des scores du profil

À partir de ces scores, on peut déterminer facilement le profil et son score le plus haut, ce que nous faisons dans la deuxième et troisième fonction. La dernière fonction permet de déterminer une classification des expressions, là aussi nous proposons le code 17 commenté ci-dessous :

```

//boucle sur les expressions streaming de la collection trainer
data.dataExpression.forEach(function (expressions) {
  //boucle sur le panel d'expression de la collection trainer
  for (const [expression, value] of Object.entries(expressions.FaceExpression)) {
    //on ne souhaite pas faire de statistiques sur l'expression neutre
    if (value > 0 && expression != 'neutre') {
      //on ajoute au serviceFactory de dataProfil
      resultatExpressions[expression].count += 1;
      resultatExpressions[expression].total += value;
      if (resultatExpressions[expression].best < value) {
        resultatExpressions[expression].best = value;
        resultatExpressions[expression].image = expressions.FaceDetection.image;
        resultatExpressions[expression].key = counter;
      }
      //serviceFactory temporaire utiliser pour l'affichage des resultats
      streamExpression[expression].push({
        x: expressions.created,
        y: value,
      });
      counter++;
    }
    ;
    totalCounter++;
  }
});
//ratio
for (const [expression, value] of Object.entries(resultatExpressions)) {
  resultatExpressions[expression].ratio =
  Math.round(resultatExpressions[expression].count / counter * 100);
};
dataProfil.expressions.resultatExpressions = resultatExpressions;
dataProfil.expressions.score = Math.round(counter / totalCounter * 100);
}

```

Code 17 – fonction emotions() classification des expressions

Enfin, nous actualisons les données de dataProfil sur la collection « trainer » puis nous proposons à l'utilisateur d'afficher les résultats.

3.4.3 - Affichage du profil

L'affichage du profil exploite l'ensemble des données classifiées. De plus cette partie comporte principalement des restitutions sous forme de texte afin de proposer un contenu d'analyse pertinent dont les sources ont été obtenues d'après plusieurs variantes sur Kolb (Chapman[83], Rivard[85]). Nous incluons ainsi des analyses sur les 4 principaux profils (Adaptateur, Convergeur, Assimilateur, Divergeur) et leurs 4 sous profils (Activist, Pragmatist, Theorist, Reflector) à travers 12 fichiers. Une page « exemple de résultats » est disponible sur l'application afin de se faire une idée du processus sans pour autant souhaiter réaliser le test. La première partie de l'affichage présente une synthèse des résultats de l'apprenant, figure 50. Il y figure un radar reprenant les scores « racine ». C'est à partir de ces scores que l'on peut en déduire le reste (les scores, les forces, les textes).

Votre profil d'apprenant est : ADAPTATEUR/CONVERGEUR

L'adaptateur

Actions orientées vers la mobilisation et l'animation des collectivités

- Bonnes habiletés à l'application et à l'action.
- Prédilection à agir rapidement et à prendre des risques.
- Résolution intuitive des problèmes.



Score fort

Le divergeur

Réflexions orientées vers les personnes et les situations concrètes

- Bonnes habiletés à imaginer des solutions.
- Intérêt pour les personnes et pour les différences.
- Idéation et observation sous des angles différents.



Score fort

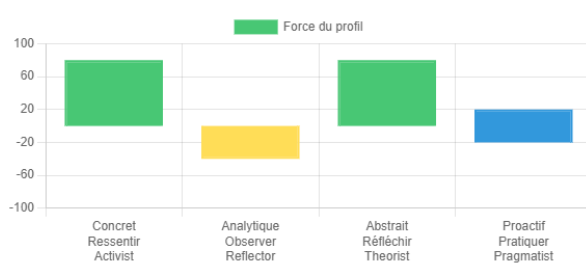
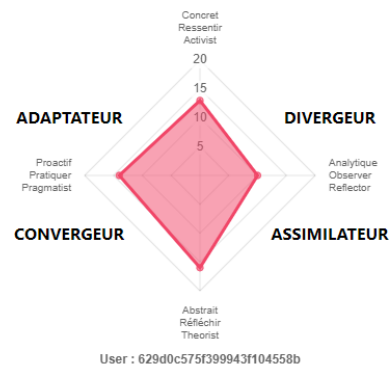


Figure 50 – Affichage des résultats, synthèse du profil de l'apprenant

On affiche ensuite le contenu, pour plus de précision sur le profil et les 8 scores obtenus. L'affichage se fait sous forme d'accordéon pour gagner en lisibilité. La dernière partie reprend le profil émotionnel que nous représentons en figure 51 ci-dessous.



Figure 51 – Affichage des résultats, profil émotionnel

Le profil émotionnel est composé de trois parties. Une partie vignettes qui reprend les meilleurs moments du test en affichant « la photo » de l'expression la plus forte dans chacune des catégories. Elle indique l'intensité de l'émotion du profil couplé à une comparaison avec un panel de « trainer » ayant le même profil d'apprenant et sur l'ensemble des « trainers ». Une deuxième partie représente le streaming des émotions émises sous forme d'une ligne de vie. La dernière partie représente deux graphiques de comparaison avec les résultats du panel

apprenant et du panel global. Un radar sur les ratios des expressions émises et un graphique en barres représentent le pourcentage global d'expressions émises par rapport à l'expression neutre. En effet, grâce à la classification effectuée sur dataProfil il est assez facile d'afficher les résultats de synthèse sur la collection « trainers ». Dans l'éventualité que nous aurions à terme des millions de « trainer », nous envisagerons d'externaliser les synthèses « panel apprenant » et « panel global » sur un nouveau document. En dernière partie dans la gestion des données, nous verrons la page OpenData.

3.4.4 - Les statistiques OpenData

La page OpenData a pour objectif de montrer à tout utilisateur l'ensemble des données recueillies lors d'un test. L'objectif de cette page est de solliciter d'éventuelles demandes de partage des données. Dans ce cas, nous développerons l'ouverture de l'API vers l'extérieur. Cette page en figure 52 représente les données des panels et une représentation du « trainer » « userDemo » que nous reprenons sur la page « exemple de résultat ».

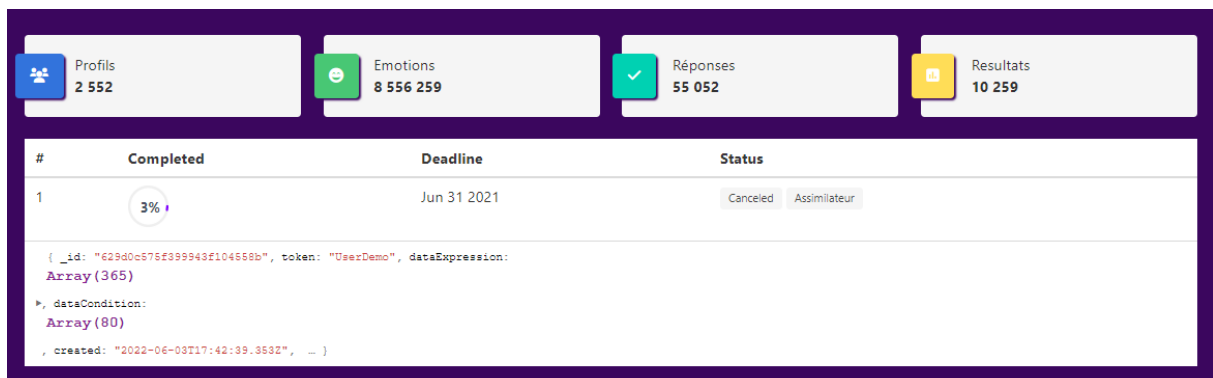


Figure 52 – Page OpenData avec visibilité des données JSON d'un Trainer

L'affichage JSON est interactif et permet de plonger sur toutes les données collectées. L'affichage des données statistiques de type count est ici fictif, car nous ne disposons pas actuellement de l'analyse de 2552 profils, mais ces informations permettent de donner un sens sur ce que nous souhaitons afficher à terme.

Cette page clôt notre application dans ses fonctionnalités. Nous verrons dans le prochain chapitre les efforts réalisés pour sécuriser notre application.

3.5 - Sécurité et charge serveur

Ce chapitre est dédié à la sécurité de notre application, nous verrons ici les principales failles de sécurité d'une application NodeJs/MongoDB et les stratégies que nous avons appliquées pour les renforcer.

3.5.1 - SSH / TLS et HTTPS

La première démarche que nous avons réalisée est la création d'un certificat SSL sur le domaine `datatrainx.akairnet.fr`, url de notre application. Ce certificat SSL (Secure Sockets Layer) permet d'établir avec certitude le lien entre le site Internet et son propriétaire, mais aussi de sécuriser les échanges entre le frontend et le backend. Il permet ainsi le chiffrement TLS [86] des informations et garantir ainsi la confidentialité des échanges. Techniquement, les identifications numériques permettent d'associer une clé publique au propriétaire du domaine. La clé publique permet l'échange avec une clé privée stockée sur le serveur. Celle-ci va chiffrer les informations transmises entre le client et le site Internet. Ceci nous permet de proposer une application sur le port 443 dédié en HTTPS. Nous l'associons avec un module de contrôle d'intégrité afin de vérifier que le message n'a pas été modifié au cours de son passage sur Internet. Le certificat est déposé chez Let's Encrypt qui est une autorité de certification qui fournit gratuitement des certificats. Le certificat a une durée de vie de 6 mois, il est généré et/ou renouvelé par le logiciel Certbot [87]. Ce dernier est installé sur notre serveur et effectue le processus de validation du domaine. Le principe de fonctionnement est assez simple, on effectue l'ensemble des opérations avec une simple ligne de commande comme l'illustre la figure 53. Ceci nous permet de générer deux fichiers, la clé publique `cert.pem` et la clé privée `privkey.pem`, qui sont respectivement utilisés sur nos deux applications frontend et backend.

```
-[0m - Congratulations! Your certificate and chain have been saved at:
C:\Certbot\live\datatrainx.akairnet.fr\fullchain.pem
Your key file has been saved at:
C:\Certbot\live\datatrainx.akairnet.fr\privkey.pem
Your cert will expire on 2021-04-04. To obtain a new or tweaked
version of this certificate in the future, simply run certbot
again. To non-interactively renew *all* of your certificates, run
"certbot renew"
- If you like Certbot, please consider supporting our work by:

Donating to ISRG / Let's Encrypt: https://letsencrypt.org/donate
Donating to EFF: https://eff.org/donate-le

C:\Windows\system32>
```

Figure 53 – Renouvellement du certificat avec la commande `certbot renew`

La lecture des certificats est assurée par le composant sirv-cli pour l'application frontend sur le port 443 et le composant natif https de NodeJS pour le côté backend sur le port 4000.

La deuxième étape consiste à sécuriser l'API, c'est ce que nous verrons dans le prochain chapitre avec la création de tokens d'authentification.

3.5.2 - JWT

Afin de sécuriser notre API, il est important d'identifier les utilisateurs/applications qui font des appels et déterminer leurs autorisations. Sinon dans l'absolu, tout le monde pourrait utiliser notre API et utiliser nos services CRUD compromettant ainsi l'intégrité de notre base de données. Pour cela nous nous appuyons sur le standard RFC7519 [88] JSON Web Token (JWT). Il permet l'échange sécurisé de jetons (tokens) entre plusieurs parties. C'est un composant supplémentaire coté backend à nodeJS (jsonwebtoken). Il est utilisé dans notre contrôleur aux différentes étapes des appels. Le principe est d'attribuer un token d'une durée de vie de 24 h 00 lors de la création d'un « trainer », cette étape se fait à la création du token_id au démarrage du test en frontend. Ce token contient l'ID de l'utilisateur et une chaîne secrète en tant que payload (les données encodées dans le token). La chaîne secrète est la clé privée générée par certbot (privkey.pem). Un middleware assure la vérification du token, il est appelé sur chacune de nos routes afin de vérifier leur autorisation d'exécution. Chaque appel frontend est complété d'une authentification de ce token généré coté backend, figure 54.

```
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VySWQiOiI2MmEyMDA2MTI2MGNiZGF1OWI0MTZinz1XQiojE2NTQ3ODQwOTcsImV4cCI6MTY1NDg3MDQ5N30.gclJJDsI_BixX0ACnuGhR8w0xgQOT0qo5GFk7U65euI
Connection: keep-alive
Content-Length: 8679
Content-Type: application/json
Host: datatrainx.akairnet.fr:4000
Origin: https://datatrainx.akairnet.fr
Referer: https://datatrainx.akairnet.fr/
```

Figure 54 – Entête Http complété du token coté frontend

Des tests ont été réalisés avec des applications tiers comme Postman et nous avons bien un retour de type 401 Unauthorized si nous ne sommes pas authentifiés.

Nous verrons dans la prochaine section l'exploitation du dictionnaire CVE relatif aux failles de sécurité des applications web.

3.5.3 - Les principales failles d'une application web

L'organisme à but non lucratif MITRE [89] soutenu par le département de la Sécurité intérieure des États-Unis maintient un dictionnaire des principales failles de sécurité des applications web, appelé CVE. On y recense par exemple les principales vulnérabilités connues d'une application NodeJS. La faille principale est dans l'installation d'un composant via NPM. Dans ce sens, nous avons toujours pris les dernières versions et si ces dernières ont des dépendances obsolètes nous les avons mises à jour. NPM est relié à CVE et nous indique si un composant est compromis avec différents niveaux d'alertes. La commande « npm audit » nous indique « found 0 vulnerabilities ». Avec l'authentification API c'est pour nous les deux principales failles de sécurité à vérifier. Nous pensons donc être à l'abri des principales failles de sécurité que nous référençons dans le tableau 13 ci-dessous avec une description.

Tableau 13 – Failles de sécurité et stratégies appliquées

Failles de sécurité	Stratégies
Faille XSS Injecter du contenu dans une page	L'idée d'injecter du code JS sur mongoDB pour ensuite l'exécuter côté client nous paraît compliquée au vu de la modélisation des données attendues, que ce soit côté service factory ou modèle/contrôleur.
File Inclusion Code execution	Nous n'avons pas d'upload de fichiers sur notre serveur avec notre applicatif, donc difficile d'exécuter du code arbitraire.
Injection SQL	Il est possible de corrompre les données, cela peut impacter le panel représentatif. Mise en place de vérification des données cohérentes. Cependant il n'est pas possible d'exécuter une requête non voulue de par le principe modèle/contrôleur.
CSRF/CRLF	Pas d'authentification type formulaire.
Force brute / DDoS	Oui actuellement notre réseau n'est pas adapté, il est possible de faire tomber notre serveur en cas de fortes sollicitations (même non voulues). Vérifier si un token injecte trop de contenu (phase de test interminable).

Overflow	Mises à jour des bibliothèques, de NodeJS.
ByPass	Pas de session admin sur l'application.
Directory traversal	Route minimaliste actuellement sur notre application.

Il existe des composants complémentaires pour NodeJS dans le domaine de la sécurisation de l'application, certaines sont intéressantes, mais au stade de la rédaction de ce mémoire nous ne les avons pas installés/testés. Nous citerons cependant NjSolid [90] qui permet de lancer une plateforme de remplacement en cas de problème sur NodeJS avec vérification des alertes CVE. Helmet [91], adapté côté API Express pour sécuriser les en-têtes http ou encore RateLimitFlexible pour les attaques DDoS [92]. Ce panel non exhaustif de composants clôt notre chapitre sur la sécurité et nous permet d'aborder la dernière section de notre mémoire liée à l'organisation, la planification et l'environnement en général que nous avons suivie pour mener à bien ce projet.

3.6 - Environnement de développement

Dans ce dernier chapitre, nous indiquerons les principes généraux que nous avons suivis dans l'organisation de notre projet. Cela comprend les licences de développements, les règles et conventions que nous nous sommes imposées dans notre développement. Les tests et intégrations continus, la gestion de nos versions et enfin la planification et temps allouée dans chacune des catégories de ce mémoire.

3.6.1 - Licences, conventions et documentation

Nous proposons la solution DataTrainX sur le dépôt GitHub sous licence MIT. Cela veut dire que le logiciel est libre et open source. Cette licence de logiciel est permissive et implique très peu de limitations sur la réutilisation du code, elle est ainsi compatible avec de nombreuses autres licences. Nous l'avons complétée sur la partie « website » de la licence créative Commons CC BY NC SA 4.0. Dans le cas de la réutilisation des données en OpenData par exemple, cela stipule qu'elles ne peuvent pas être exploitées commercialement. Ces deux licences indiquent une grande liberté d'utilisation et modification du logiciel sans pour autant s'en attribuer la propriété intellectuelle. Ces informations se trouvent en pied de page du site, figure 55 et sur un fichier licence du code racine.

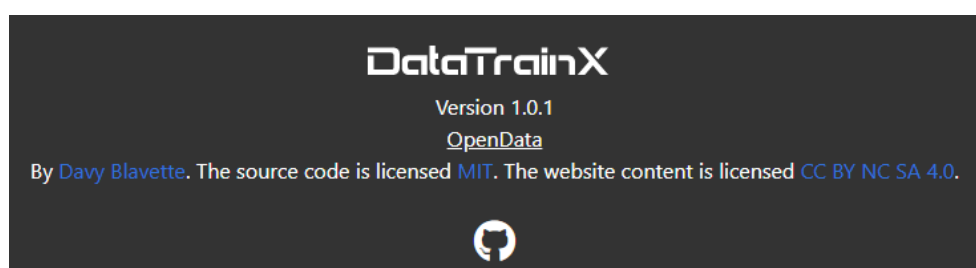


Figure 55 – Informations des licences en pied de page

Pour le développement du code, nous nous sommes appuyés sur les conventions de nommages habituelles en Javascript, à savoir le Camel case pour la déclaration des variables et fonctions, ces notions se retrouvent dans la plupart des codes que nous avons partagés dans ce mémoire. Enfin, dernier point, concernant la documentation, nous reconnaissons un effort modeste sur le code en général. Ce qui est plutôt dommage, car JavaScript est plutôt performant dans ce domaine avec JSDoc.

JSDoc est un langage de balisage utilisé pour documenter les codes sources Javascript. En utilisant des commentaires qui contiennent des informations pour JSDoc, il est possible de générer une documentation par différents formats, tels que l'HTML par exemple. A notre décharge, ce sont des pratiques que l'on utilise surtout avec du code backend conséquent, ce qui n'était pas vraiment notre cas. Nous avons préféré orienter l'effort sur la documentation GitHub via trois fichiers Readme. Le premier est côté frontend, il présente l'application, l'installation, la démonstration et les informations clés sous forme de badge, figure 56.



Figure 56 – Badges utilisés sur « readme » en frontend, afin de visualiser les technologies clé de DataTrainX

Le deuxième se trouve sur la partie backend où l'on explique le principe de l'API. Enfin le troisième fichier se trouve sur dataset/fer2013 où l'on retrouve le code Python pour générer les modèles sur Fer2013. Comme ce n'est pas un projet NodeJS avec NPM, on explique les différentes bibliothèques à installer avec PIP, la documentation sur ce projet initial est relativement conséquente. L'ensemble de nos sources sont développées selon une des méthodes agiles d'intégration continue, que nous aborderons dans le prochain chapitre.

3.6.2 - Tests & intégration continue

L'intégration continue (IC) est le processus long que nous avons réalisé depuis le démarrage du projet. Ce sont des notions que nous avons abordées en phase d'analyse et que nous avons su mettre en pratique. Nous pensons d'ailleurs que cela se traduit et peut être se répercute dans la forme et l'écrit de ce mémoire. Chacun de nos modules a été développé et intégré successivement formant à la fin notre application. Cela nous a permis de corriger très tôt les bugs successifs et de compartimenter chaque module. Les tests et les intégrations se sont donc réalisés continuellement au fil de l'eau comme illustré en figure 57.

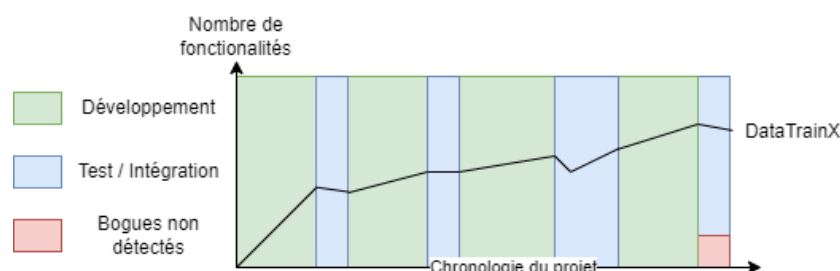


Figure 57 – Intégration continue sur DataTrainX

L'objectif était donc, quand nous en avons la disponibilité de fabriquer « souvent » (build), de tester « souvent » (test) et d'intégrer « souvent » (intégrate). L'objectif de l'IC est de vérifier, de façon automatique et à chaque modification de code source, que le résultat des modifications ne produit pas de régression de l'application en cours de développement. Le « build » correspond à un ensemble de composants livrables, pleinement testés et versionnés. Il est défini par un numéro de version que nous indiquons sur la partie « footer » du site et la section releases de GitHub, figure 58.



Figure 58 – Principe de badge, dernière release sur GitHub

Les tests sont réalisés sur une page dédiée à ces processus appelés DebugTest accessible à partir du menu et activés selon les besoins. En cas de régression, ce qui peut se produire malgré les tests, nous avons la possibilité de revenir rapidement aux versions précédentes du code, ce qui sous-entend d'utiliser un outil de gestion de versions du code. Le fait de pousser régulièrement les modifications quand elles sont jugées « stable » offre aussi l'avantage de ne pas reprendre de nombreux points en cas de régression et de résoudre les builds ratés rapidement. C'est ce que nous verrons dans le prochain chapitre sur ces outils, Travis CI, TortoiseGit et GitHub que nous avons utilisés.

3.6.3 - Planification & gestion des versions

Pour gérer l'ensemble de nos releases nous avons utilisé trois outils. Travis CI qui fournit un service en ligne utilisé pour compiler, tester et déployer le code source des logiciels développés dont nous avons donné les accès sur notre dépôt GitHub. Sa configuration s'effectue en YAML. TortoiseGit qui est un logiciel client de gestion de version Git, et enfin GitHub qui est un service web d'hébergement et de gestion de développement de logiciels. L'ensemble de ces outils permettent de nous donner de nombreuses statistiques sur le déroulement du projet. On peut constater par exemple que nous avons débuté DataTrainX le 12 décembre 2021, comme l'illustre la figure 59. Qu'une longue phase d'analyse sans contribution s'est réalisée entre janvier et mi-mars 2022. En réalité, sur le premier trimestre de l'année, nous étions en phase d'analyse et sur l'implémentation du modèle CNN avec

Fer2013. Notre première release d'ailleurs s'est faite fin mars et correspond à la version 1.0.0 de notre application Python/Tensorflow de construction des modèles sur le dataset Fer2013. C'était une phase de R&D avec des installations hardware avec CUDA par exemple et des phases de compréhension et de tests de fonctionnement de Tensorflow, donc peu de code à commiter. Cela correspond aussi à la phase de conception de notre mémoire, l'analyse ayant débuté en octobre 2021.

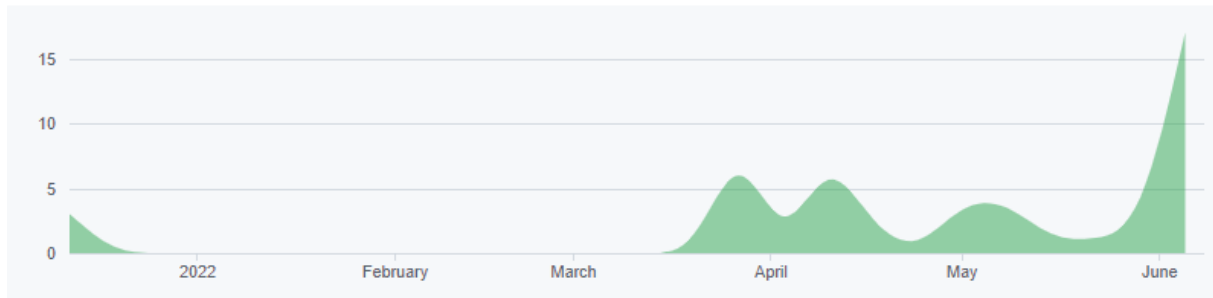


Figure 59 – Contributions de type commit sur DataTrainX, source GitHub

Le développement du projet DataTrainX a donc débuté en avril avec une forte accélération dès la fin mai afin de se donner l'objectif de présenter ce mémoire en juillet 2022. L'ensemble comporte 51 commits représentant un peu plus de 50 000 lignes de code en 1500 ajouts repartis sur 63 fichiers. Cela inclut les projets tiers comme CnnExplainer et Face-api. Les composants du projet (paquets NPM ou bibliothèques) ne sont pas compris dans ces statistiques car ils s'installent directement via NPM ou PIP tel qu'expliqué sur les fichiers Readme de nos trois applications.

À titre d'information, les 25 paquets NPM utilisés représentent à eux seuls 2 millions de lignes de code car eux-mêmes utilisent d'autres paquets (350 au total).

Ce chapitre clôt notre mémoire, nous proposons un bilan du projet dans la phase de conclusion.

CONCLUSION & PERSPECTIVES

J'avais initialement l'ambition de m'orienter sur un projet en rapport avec mon métier de formateur technique en informatique. J'avais plusieurs objectifs. D'une part, me donner des clés sur la pédagogie et l'adaptation de mes cours selon les différents profils de stagiaires. D'autre part, faire un travail de veille technologique sur des domaines de compétences que je ne connaissais pas. Par ailleurs, c'est toujours le leitmotiv que je me suis fixé dans ma carrière d'informaticien, à savoir que tout nouveau projet doit amener à la technologie la mieux adaptée et non à celle que l'on maîtrise le mieux.

Pour ce projet DataTrainX, j'ai particulièrement fait du « zèle » dans ce domaine, car en octobre 2021, date à laquelle je me suis lancé sur ce mémoire, je n'avais que peu de connaissances sur les technologies utilisées dans DataTrainX. Peu de connaissances sur le deep-learning et les réseaux CNN. Je n'avais jamais développé sur NodeJS et sur le moteur frontend svelte qui m'était inconnu. Le framework CSS Bulma a été une découverte ainsi que le NoSQL et MongoDB que je n'avais jamais utilisés auparavant. La décision de passer sur NoSQL ne s'est faite que tardivement. Initialement j'avais dans l'idée de faire une base de données de type MySQL, puis au fur et à mesure du développement, le NoSQL paraissait comme une évidence et la technologie la mieux adaptée pour ce projet. Alors certes, je n'ai pas la prétention, chose que je n'aurai d'ailleurs jamais sans doute, d'être un expert dans ces domaines, néanmoins il me semble avoir acquis une culture et des connaissances indéniables dans ces technologies. Pour ma part, je pense donc humblement avoir su remplir mon deuxième objectif.

Concernant la pédagogie différenciée, la première étape était d'obtenir un profil d'apprenant selon les modèles dominants. Dans ce domaine, les résultats sur Kolb sont intéressants et je regrette toujours que ces outils soient si peu considérés. Pour avoir pu tester personnellement mon application, je dois dire que mon profil, d'ailleurs « mis en pâture » sur la page « exemple de résultats », est assez significatif. Dans mon métier, je constate malheureusement trop souvent des postures figées dans l'incapacité à compenser les échecs d'un stagiaire et je n'arrive pas à me résigner au fait qu'il n'existe pas de solutions. Je pense sincèrement que Kolb est un outil parmi d'autres dans une logique de recherche d'améliorations dans la transmission du savoir.

Dans ce projet je voulais acquérir des connaissances en deep-learning et c'est donc naturellement que j'ai fait un rapprochement entre émotion et apprentissage. L'émotion, en soi, était un prétexte pour utiliser ce type de technologie, car j'ai bien conscience que même si le rapprochement n'est pas dénoué de toute logique, je savais que ce serait une tâche bien trop complexe pour arriver à des conclusions significatives. J'ai essayé tout de même d'arriver à des résultats au stade de ce mémoire. Ils ne sont cependant pas significatifs.

Il est prévu par la suite d'intégrer des tests complémentaires de mises en situations émotionnelles de l'utilisateur. Ce que l'on avait appelé par les phases castings et puzzles par exemple. Le choix du puzzle est compliqué, car la corrélation même de la pertinence de ces données corrélées au profil d'apprentissage est du domaine de la psychologie. Il faudrait, dans ce domaine, trouver un partenariat avec un chercheur en psychologie afin de donner du sens et une direction aux résultats dans le choix de la mise en situation. La prochaine étape importante est de proposer l'application aux étudiants de l'université de Rodez. Le panel de test reste une étape essentielle sur le retour d'expérience utilisateur et l'enrichissement des données. À ce stade, l'application DataTrainX est opérationnelle et de qualité suffisante pour se lancer sur ces prochaines étapes. Ce prototype était destiné à devenir un système informatique de la REF axé sur le profil d'un apprenant, dans ses capacités fonctionnelles, je pense qu'il a atteint son objectif.

Pour en revenir au deep-learning, j'ai conscience que depuis la dernière décennie l'informatique s'engage dans des bouleversements profonds, remettant en cause de nombreux acquis. Le deep-learning, le NoSQL, la conteneurisation, NodeJS ou même les méthodologies agiles en sont des exemples. Et j'éviterai par ailleurs de m'engager sur l'informatique quantique qui peut présager l'émergence d'un nouveau paradigme dans ce domaine, d'ici 10 ans selon IDC [93]. L'informatique est donc une science en éternelle mouvance qui demande aux informaticiens beaucoup d'exigence dans leurs capacités à s'adapter.

C'est ce que j'ai essayé de démontrer comme compétence dans ce projet à travers une multitude d'expériences acquises à travers ces 20 années dans ce domaine.

BIBLIOGRAPHIE

- [1] « Ce que la recherche nous dit sur les styles d'apprentissage (ou retour sur un mythe tenace) | Édupass ». <https://edupass.hypotheses.org/1049> (consulté le 25 février 2020).
- [2] « E. Couzon et F. Dorn : Les émotions : développer son intelligence émotionnelle. Issyles-Moulineaux : ESF éditeur, 2009. »
- [3] « David A. Kolb », *Wikipedia*. 14 août 2019. Consulté le: 1 février 2020. [En ligne]. Disponible sur: https://en.wikipedia.org/w/index.php?title=David_A._Kolb&oldid=910789064
- [4] « Réseau neuronal convolutif », *Wikipédia*. 27 janvier 2020. Consulté le: 29 janvier 2020. [En ligne]. Disponible sur: https://fr.wikipedia.org/w/index.php?title=R%C3%A9seau_neuronal_convolutif&oldid=166794123
- [5] « Paul Ekman », *Wikipédia*. 5 janvier 2020. Consulté le: 23 janvier 2020. [En ligne]. Disponible sur: https://fr.wikipedia.org/w/index.php?title=Paul_Ekman&oldid=166068024
- [6] J. Houssaye, *La Pédagogie : une encyclopédie pour aujourd'hui*. ESF éditeur, 2018.
- [7] « A. Mehrabian : Communication without words. *Psychology Today*, 2.4:53–56, 1968. ».
- [8] « RAVDESS | SMART Lab ». <https://smartlaboratory.org/ravdess/> (consulté le 22 janvier 2020).
- [9] « WIDER FACE: A Face Detection Benchmark ». <http://shuoyang1213.me/WIDERFACE/> (consulté le 30 novembre 2021).
- [10] « Nielsen, J. (1994) *Usability Engineering*, AP Professional, Cambridge. ».
- [11] « Développement rapide d'applications », *Wikipédia*. 6 décembre 2019. Consulté le: 18 novembre 2021. [En ligne]. Disponible sur:

https://fr.wikipedia.org/w/index.php?title=D%C3%A9veloppement_rapide_d%27applications&oldid=165182808

[12] « “Deep learning” : les dessous d’une technologie de rupture », *Futuribles*. <https://www.futuribles.com/fr/document/deep-learning-les-dessous-dune-technologie-de-rupt/> (consulté le 2 novembre 2021).

[13] « Michael Grimm, D Dastidar, and K Kroschel. Recognizing emotions in spontaneous facial expressions. In International Conference on Intelligent Systems And Computing (ISYC), 2006. »

[14] « Machine de Boltzmann restreinte », *Wikipédia*. 20 octobre 2021. Consulté le: 2 novembre 2021. [En ligne]. Disponible sur: https://fr.wikipedia.org/w/index.php?title=Machine_de_Boltzmann_restreinte&oldid=187302440

[15] « Apprentissage profond », *Wikipédia*. 20 octobre 2021. Consulté le: 2 novembre 2021. [En ligne]. Disponible sur: https://fr.wikipedia.org/w/index.php?title=Apprentissage_profond&oldid=187302384

[16] « Carroll Izard », *Wikipedia*. 4 février 2021. Consulté le: 22 septembre 2021. [En ligne]. Disponible sur: https://en.wikipedia.org/w/index.php?title=Carroll_Izard&oldid=1004816203

[17] « Silvan Tomkins », *Wikipedia*. 11 janvier 2021. Consulté le: 22 septembre 2021. [En ligne]. Disponible sur: https://en.wikipedia.org/w/index.php?title=Silvan_Tomkins&oldid=999642309

[18] « Figure 2: Sample images from the Cohn-Kanade DFAT-504 facial expression... », *ResearchGate*. https://www.researchgate.net/figure/Sample-images-from-the-Cohn-Kanade-DFAT-504-facial-expression-database-Shown-are-the-six_fig1_221318664 (consulté le 2 novembre 2021).

[19] « Jean Piaget », *Wikipédia*. 31 octobre 2021. Consulté le: 2 novembre 2021. [En ligne]. Disponible sur: https://fr.wikipedia.org/w/index.php?title=Jean_Piaget&oldid=187585172

- [20] « Gérard Vergnaud », *Wikipédia*. 17 juillet 2021. Consulté le: 2 novembre 2021. [En ligne]. Disponible sur: https://fr.wikipedia.org/w/index.php?title=G%C3%A9rard_Vergnaud&oldid=184734507
- [21] « Qu'est-ce que la neuropédagogie ? », *XOS*, 8 février 2018. <https://www.xos-learning.fr/blog/qu-est-ce-que-la-neuropedagogie/> (consulté le 23 septembre 2021).
- [22] « Séminaire "Inclusion des élèves en situation de handicap dans le cadre des enseignements de sciences expérimentales" Paris, 26 mai 2018 - Christian Sarralié ».
- [23] R. Dunn, S. A. Griggs, J. Olson, M. Beasley, et B. S. Gorman, « A Meta-Analytic Validation of the Dunn and Dunn Model of Learning-Style Preferences », *The Journal of Educational Research*, vol. 88, n° 6, p. 353-362, 1995.
- [24] R. Riding et I. Cheema, « Cognitive Styles—an overview and integration », *Educational Psychology*, vol. 11, n° 3-4, p. 193-215, janv. 1991, doi: 10.1080/0144341910110301.
- [25] *Motivational styles in everyday life: A guide to reversal theory*. Washington, DC, US: American Psychological Association, 2001, p. xiv, 373. doi: 10.1037/10427-000.
- [26] « Myers Briggs Type Indicator », *Wikipédia*. 11 mai 2020. Consulté le: 3 juin 2020. [En ligne]. Disponible sur: https://fr.wikipedia.org/w/index.php?title=Myers_Briggs_Type_Indicator&oldid=170746746
- [27] C. W. Allinson et J. Hayes, « The Cognitive Style Index: A Measure of Intuition-Analysis For Organizational Research », *J Management Studies*, vol. 33, n° 1, p. 119-135, janv. 1996, doi: 10.1111/j.1467-6486.1996.tb00801.x.
- [28] « Test : Mon style d'apprentissage selon Kolb ». Consulté le: 22 janvier 2020. [En ligne]. Disponible sur: https://www.fondationbombardier.ca/wp-content/uploads/2019/09/Test_mon_style_dapprentissage_selon_kolb.pdf
- [29] N. Entwistle, « Approaches to learning and forms of understanding », janv. 1998.
- [30] J. Vermunt, « Metacognitive, cognitive and affective aspects of learning styles and strategies: A phenomenographic analysis », *Higher Education*, vol. 31, p. 25-50, janv. 1996, doi: 10.1007/BF00129106.

[31] « [Coffield et al. 2004] (en) F. Coffield, D. Moseley, E. Hall et K. Ecclestone, « Learning styles and pedagogy in post-16 learning : A systematic and critical review », Learning and Skills Research Centre, Londres, 2004 ».

[32] « L'utilité des « styles d'apprentissage » VAK (visuel, auditif, kinesthésique) en éducation : entre l'hypothèse de recherche et le mythe scientifique - Luc Rousseau, Yvon Gauthier et Julie Caron ».

[33] H. Pashler, M. McDaniel, D. Rohrer, et R. Bjork, « Learning Styles: Concepts and Evidence », *Psychol Sci Public Interest*, vol. 9, n° 3, p. 105-119, déc. 2008, doi: 10.1111/j.1539-6053.2009.01038.x.

[34] « Méthode de Viola et Jones », *Wikipédia*. 12 octobre 2020. Consulté le: 22 octobre 2021. [En ligne]. Disponible sur: https://fr.wikipedia.org/w/index.php?title=M%C3%A9thode_de_Viola_et_Jones&oldid=175497864

[35] « OpenCV », *SourceForge*. <https://sourceforge.net/projects/opencvlibrary/> (consulté le 22 octobre 2021).

[36] « Caractéristiques pseudo-Haar », *Wikipédia*. 28 décembre 2018. Consulté le: 22 octobre 2021. [En ligne]. Disponible sur: https://fr.wikipedia.org/w/index.php?title=Caract%C3%A9ristiques_pseudo-Haar&oldid=155206932

[37] « Image intégrale », *Wikipédia*. 9 juillet 2013. Consulté le: 22 octobre 2021. [En ligne]. Disponible sur: https://fr.wikipedia.org/w/index.php?title=Image_int%C3%A9grale&oldid=94817551

[38] « Paul Viola et Michael Jones, « Robust Real-time Object Detection » ».

[39] « #017 Face detection algorithms comparison », *Master Data Science*, 1 juin 2020. <https://datahacker.rs/017-face-detection-algorithms-comparison/> (consulté le 9 novembre 2021).

[40] « Artificial Intelligence for Human Computing: ICMI 2006 and IJCAI 2007 International Workshops, Banff, Canada, November 3, 2006, Hyderabad, India, January 6, 2007, Revised

Selected and Invited Papers | Jeffrey F. Cohn (auth.), Thomas S. Huang, Anton Nijholt, Maja Pantic, Alex Pentland (eds.) | download ». <https://fr1lib.org/book/2317881/08d05f?id=2317881&secret=08d05f> (consulté le 10 novembre 2021).

[41] « *Facial action coding system* », *Wikipédia*. 17 juin 2021. Consulté le: 10 novembre 2021. [En ligne]. Disponible sur: https://fr.wikipedia.org/w/index.php?title=Facial_action_coding_system&oldid=183888407

[42] U. Hess, R. Adams, et R. Kleck, « Facial Appearance, Gender, and Emotion Expression. », *Emotion (Washington, D.C.)*, vol. 4, p. 378-88, janv. 2005, doi: 10.1037/1528-3542.4.4.378.

[43] D. A. Sauter et A. H. Fischer, « Can perceivers recognise emotions from spontaneous expressions? », *Cognition and Emotion*, vol. 32, n° 3, p. 504-515, 2018, doi: 10.1080/02699931.2017.1320978.

[44] Zhang et al, « Joint Face Detection and Alignment using Multi-task Cascaded Convolutional Networks ».

[45] *CNN Explainer*. Polo Club of Data Science, 2021. Consulté le: 7 décembre 2021. [En ligne]. Disponible sur: <https://github.com/poloclub/cnn-explainer>

[46] « TensorFlow », *Wikipédia*. 6 octobre 2021. Consulté le: 7 décembre 2021. [En ligne]. Disponible sur: <https://fr.wikipedia.org/w/index.php?title=TensorFlow&oldid=186919162>

[47] « Compute Unified Device Architecture », *Wikipédia*. 6 juillet 2021. Consulté le: 7 décembre 2021. [En ligne]. Disponible sur: https://fr.wikipedia.org/w/index.php?title=Compute_Unified_Device_Architecture&oldid=184400927

[48] « AffectNet – Mohammad H. Mahoor, PhD ». <http://mohammadmahoor.com/affectnet/> (consulté le 7 décembre 2021).

[49] « Papers with Code - FER2013 Dataset ». <https://paperswithcode.com/dataset/fer2013> (consulté le 7 décembre 2021).

- [50] « GEMEP Corpus - Swiss Center For Affective Sciences - UNIGE », 29 juin 2016. <https://www.unige.ch/cisa/gemep> (consulté le 18 novembre 2021).
- [51] « Resources – Jeffrey Cohn ». <http://www.jeffcohn.net/resources/> (consulté le 7 décembre 2021).
- [52] « Papers with Code - MMI Dataset ». <https://paperswithcode.com/dataset/mmi> (consulté le 7 décembre 2021).
- [53] « The MUG Facial Expression Database | Multimedia Understanding Group ». <https://mug.ee.auth.gr/fed/> (consulté le 7 décembre 2021).
- [54] « Yale Face Database | vision.ucsd.edu ». <http://vision.ucsd.edu/content/yale-face-database> (consulté le 11 décembre 2021).
- [55] « Hadoop », *Wikipédia*. 23 décembre 2020. Consulté le: 29 mars 2022. [En ligne]. Disponible sur: <https://fr.wikipedia.org/w/index.php?title=Hadoop&oldid=177941828>
- [56] J. L. H. Patterson David A., « A New Golden Age for Computer Architecture ». <https://cacm.acm.org/magazines/2019/2/234352-a-new-golden-age-for-computer-architecture/fulltext> (consulté le 11 décembre 2021).
- [57] « NVIDIA cuDNN », *NVIDIA Developer*, 2 septembre 2014. <https://developer.nvidia.com/cudnn> (consulté le 11 décembre 2021).
- [58] « Intel® Optimization for TensorFlow* Installation Guide », *Intel*. <https://www.intel.com/content/www/us/en/developer/articles/guide/optimization-for-tensorflow-installation-guide.html> (consulté le 11 décembre 2021).
- [59] « GitHub - google/XNNPACK: High-efficiency floating-point neural network inference operators for mobile, server, and Web », *GitHub*. <https://github.com/google/XNNPACK> (consulté le 11 décembre 2021).
- [60] « Advanced Vector Extensions », *Wikipédia*. 13 octobre 2021. Consulté le: 11 décembre 2021. [En ligne]. Disponible sur: https://fr.wikipedia.org/w/index.php?title=Advanced_Vector_Extensions&oldid=187109317
- [61] « Let's Encrypt - Certificats SSL/TLS gratuits ». <https://letsencrypt.org/fr/> (consulté le 3 mars 2022).

[62] « Conda (package manager) », *Wikipedia*. 9 février 2022. Consulté le: 3 mars 2022. [En ligne]. Disponible sur:

[https://en.wikipedia.org/w/index.php?title=Conda_\(package_manager\)&oldid=1070868910](https://en.wikipedia.org/w/index.php?title=Conda_(package_manager)&oldid=1070868910)

[63] « npm », *Wikipédia*. 11 janvier 2022. Consulté le: 3 mars 2022. [En ligne]. Disponible sur: <https://fr.wikipedia.org/w/index.php?title=Npm&oldid=189802929>

[64] *GitHub fer2013*. [En ligne]. Disponible sur: <https://github.com/davy-blavette/Datatraining/tree/main/dataset/fer2013>

[65] K. Team, « Keras documentation: Keras Applications ». <https://keras.io/api/applications/> (consulté le 11 avril 2022).

[66] « Stanford University CS231n: Deep Learning for Computer Vision ». <http://cs231n.stanford.edu/> (consulté le 15 avril 2022).

[67] « Papers with Code - Convolutional Neural Network Hyperparameters optimization for Facial Emotion Recognition ». <https://paperswithcode.com/paper/convolutional-neural-network-hyperparameters> (consulté le 13 avril 2022).

[68] G. Sharma, « Facial Emotion Recognition (FER) using Keras », *Analytics Vidhya*, 23 septembre 2020. <https://medium.com/analytics-vidhya/facial-emotion-recognition-fer-using-keras-763df7946a64> (consulté le 13 avril 2022).

[69] « Google Colaboratory ». https://colab.research.google.com/github/RodolfoFerro/psychopathology-fer-assistant/blob/master/model/Convolutional_model.ipynb (consulté le 13 avril 2022).

[70] « Papers with Code - Deep-Emotion: Facial Expression Recognition Using Attentional Convolutional Network ». <https://paperswithcode.com/paper/deep-emotion-facial-expression-recognition> (consulté le 13 avril 2022).

[71] « tf.keras.optimizers.Adam | TensorFlow Core v2.8.0 », *TensorFlow*. https://www.tensorflow.org/api_docs/python/tf/keras/optimizers/Adam (consulté le 12 avril 2022).

[72] « Bulma: Free, open source, and modern CSS framework based on Flexbox ». <https://bulma.io> (consulté le 20 avril 2022).

[73] « Svelte • Cybernetically enhanced web apps ». <https://svelte.dev/> (consulté le 20 avril 2022).

[74] « ACID Transactions Basics », *MongoDB*. <https://www.mongodb.com/basics/acid-transactions> (consulté le 10 juin 2022).

[75] « Théorème CAP », *Wikipédia*. 13 avril 2022. Consulté le: 11 mai 2022. [En ligne]. Disponible sur: https://fr.wikipedia.org/w/index.php?title=Th%C3%A9or%C3%A8me_CAP&oldid=192822000

[76] « Maîtrisez le théorème de CAP », *OpenClassrooms*. <https://openclassrooms.com/fr/courses/4462426-maitrisez-les-bases-de-donnees-nosql/4462471-maitrisez-le-theoreme-de-cap> (consulté le 11 mai 2022).

[77] « Lempel-Ziv-Welch », *Wikipédia*. 30 mars 2022. Consulté le: 31 mai 2022. [En ligne]. Disponible sur: <https://fr.wikipedia.org/w/index.php?title=Lempel-Ziv-Welch&oldid=192389042>

[78] « Chart.js | Open source HTML5 Charts for your website ». <https://www.chartjs.org/> (consulté le 6 mai 2022).

[79] Z. J. Wang *et al.*, « CNN Explainer: Learning Convolutional Neural Networks with Interactive Visualization », *IEEE Trans. Visual. Comput. Graphics*, vol. 27, n° 2, p. 1396-1406, févr. 2021, doi: 10.1109/TVCG.2020.3030418.

[80] « Batch normalization », *Wikipedia*. 23 mai 2022. Consulté le: 1 juin 2022. [En ligne]. Disponible sur: https://en.wikipedia.org/w/index.php?title=Batch_normalization&oldid=1089368517

[81] « Dilution (neural networks) », *Wikipedia*. 13 mai 2022. Consulté le: 1 juin 2022. [En ligne]. Disponible sur: [https://en.wikipedia.org/w/index.php?title=Dilution_\(neural_networks\)&oldid=1087622447](https://en.wikipedia.org/w/index.php?title=Dilution_(neural_networks)&oldid=1087622447)

[82] « PageSpeed Insights ». https://pagespeed.web.dev/report?url=https%3A%2F%2Fdatatrainx.akairnet.fr%2F&hl=fr&form_factor=desktop (consulté le 31 mai 2022).

- [83] « Chapman. [2005]. Kolb's learning styles, 2005. Retrieved from <http://www.businessballs.com/kolblearningstyles.html>. »
- [84] « JSON Web Token », *Wikipédia*. 6 décembre 2021. Consulté le: 2 juin 2022. [En ligne]. Disponible sur: https://fr.wikipedia.org/w/index.php?title=JSON_Web_Token&oldid=188613073
- [85] P. Rivard et M. Lauzier, *La gestion de la formation et du développement des ressources humaines: pour préserver et accroître le capital compétence de l'organisation*. Presses de l'Université du Québec, 2013.
- [86] « Transport Layer Security », *Wikipédia*. 11 mars 2022. Consulté le: 7 juin 2022. [En ligne]. Disponible sur: https://fr.wikipedia.org/w/index.php?title=Transport_Layer_Security&oldid=191806531
- [87] « Certbot ». <https://certbot-prod.eff.org/> (consulté le 7 juin 2022).
- [88] M. Jones, J. Bradley, et N. Sakimura, « JSON Web Token (JWT) », Internet Engineering Task Force, Request for Comments RFC 7519, mai 2015. doi: 10.17487/RFC7519.
- [89] « MITRE », *Wikipédia*. 16 avril 2022. Consulté le: 7 juin 2022. [En ligne]. Disponible sur: <https://fr.wikipedia.org/w/index.php?title=MITRE&oldid=192915826>
- [90] « N|Solid — Node.js that's Secure, Reliable, and Extensible from NodeSource », *NodeSource*. <https://nodesource.com/products/nsolid> (consulté le 7 juin 2022).
- [91] « helmet », *npm*. <https://www.npmjs.com/package/helmet> (consulté le 7 juin 2022).
- [92] Roman, *animir/node-rate-limiter-flexible*. 2022. Consulté le: 7 juin 2022. [En ligne]. Disponible sur: <https://github.com/animir/node-rate-limiter-flexible>
- [93] « IDC Forecasts Worldwide Quantum Computing Market to Grow to \$8.6 Billion in 2027 », *IDC: The premier global market intelligence company*. <https://www.idc.com/getdoc.jsp?containerId=prUS48414121> (consulté le 10 juin 2022).

LISTE DES FIGURES, TABLEAUX ET CODES

Figure 1 – Principe de développement de DataTrainX orienté sur le prototype vertical et le test d'utilisabilité. L'utilisateur doit réaliser complètement un scénario typique d'utilisation du logiciel avant de passer à une autre fonctionnalité.	12
Figure 2 – Les 5 phases d'organisation de DataTrainX inspiré de la méthode RAD	13
Figure 3 – Architecture de construction, test d'utilisabilité	15
Figure 4 – Exemple de la description de Grimm et al. sous forme d'arbre de décision sur la région de la bouche.	17
Figure 5 – En deep-learning, l'éléphant ici n'est pas étiqueté, du moins il n'est pas nécessaire de définir de nombreux descripteurs (trompe, oreille, etc.) l'algorithme arrive par lui-même à classifier l'image [15].	18
Figure 6 – Exemples d'images étiquetées que l'on trouve à disposition. Les six émotions de base (Ekman 2006) sont représentées sur le côté gauche ainsi que l'affichage de l'ouverture de la bouche sur le côté droit [18]	19
Figure 7 – «Adapter c'est éviter de transformer une situation d'apprentissage en situation de handicap.» [22].	20
Figure 8 – Exemple de restitution des résultats de l'application DataTrainX. Obtenir un profil d'apprenant, analyser l'émotion dans une mise en situation, trouver une corrélation.	24
Figure 9 – L'extension des caractéristiques pseudo-Haar [36], à droite une caractéristique de ligne (5) trouvée par la variation de l'intensité de la lumière entre les yeux et le nez.	27
Figure 10 – Balayage de type raster	28
Figure 11 – L'image intégrale comprend la somme de l'intensité d'un pixel additionnée de toutes les intensités pixels au-dessus de lui et à sa gauche	28
Figure 12 – Calcul du vecteur sur un filtre pseudo-Haar à l'aide de l'image intégrale.	29
Figure 13 – Classification en cascade permettant une recherche rapide du visage.	30
Figure 14 – La détection des visages comprend les visages issus de l'identification et des photos. On observe la différence de résultats entre deux modèles entraînés sur des limites de décision différents.	31

Figure 15 – Schéma sur les influences de différents facteurs à prendre en considération sur la reconnaissance des expressions	32
Figure 16 – Représentation de la surprise avec le système FACS AU01+AU02+AU05B+AU026	33
Figure 17 – Les 4 étapes d’analyse pour une expression en mouvement.	35
Figure 18 – Principe d'une architecture CNN	36
Figure 19 – Principe de convolution et de l'activation ReLU, ici la comparaison entre les deux images permettent de trouver trois caractéristiques, l'une des caractéristiques (en vert) est nettoyée ensuite des valeurs négatives (ReLU).....	37
Figure 20 – Réseau CNN avec trois couches de convolution et 7 neurones de sorties obtenus avec un test local sur CNN Explainer [45] (dataset non représentatif).....	38
Figure 21 – Diagramme des cas d'utilisation de l’étude préliminaire.....	41
Figure 22 – Diagramme de séquence "passer les tests" dans le cas nominal	47
Figure 23 – Architecture des flux d’informations DataTrainX	50
Figure 24 – Architecture hybride 3 tiers avec design MVC sur une partie de l'applcatif	52
Figure 25 – Architecture technique DataTrainX	53
Figure 26 – Exemple d'images FER2013 après transformation des pixels en image	56
Figure 27 – Equation définissant l'apprentissage supervisé du modèle	56
Figure 28 – Visualisation du traitement d’un EPOCH en 27 secondes sur le modèle CNN Hyperparameter.....	58
Figure 29 – Exemple de représentation d'un entraînement avec Tensorboard	61
Figure 30 – Les 6 matrices de confusions que nous avons générées à partir des modèles entraînés	63
Figure 31 – A gauche le zoning (zones d’informations grossières), à droite le mockup avec découpage des 4 parties principales de l’application.	65
Figure 32 – Maquette prototype DataTrainX Home Page.....	66
Figure 33 – Triangle réalisé à partir du théorème de Brewer, positionnant MongoDB sur les contraintes prioritaires de distribution et de cohérence [76]	68
Figure 34 – Architecture de l’API REST sur DataTrainX	71
Figure 35 – Performances en upload de 0.7 Mb/s du serveur, test nperf.com	75
Figure 36 – Chart.js utilisé en mode streaming pour afficher les émotions détectées sur un graphique	77

Figure 37 – Principe de CnnExplainer intégré à DataTrainX avec notre modèle TinyVgg FER2013	80
Figure 38 – Organisation du développement frontend en 7 dossiers	81
Figure 39 – Test de performance de l'affichage de DatatrainX sur PageSpeed de google	82
Figure 40 – Score PageSpeed en augmentation grâce à un loader et un chargement déporté de Bundle.js.....	82
Figure 41 – Menu DataTrainX. Balise "active" qui indique où l'ons e trouve dans l'application	83
Figure 42 – Les trois étapes du loader sur la webcam. Les deux premières se font au chargement des modèles, la troisième se lance si le visage et l'expression ne sont pas détectés.	85
Figure 43 – Description de l'interface capture webcam et l'équivalent data envoyée par pushData().....	85
Figure 44 – Interface de la phase de test découpé en trois zones	86
Figure 45 – À gauche les logs MongoDB, à droite les requêtes network du client	86
Figure 46 – L'obstruction de la webcam ou son arrêt ne permet pas à l'utilisateur de poursuivre les tests	87
Figure 47 – Format mobile, page "Tests", "Présentation", "Article", "OpenData", "Résultats"	88
Figure 48 – Diagramme de séquence API REST.....	90
Figure 49 – Structure des dossiers coté backend de type API Rest	90
Figure 50 – Affichage des résultats, synthèse du profil de l'apprenant.....	96
Figure 51 – Affichage des résultats, profil émotionnel	96
Figure 52 – Page OpenData avec visibilité des données JSON d'un Trainer	97
Figure 53 – Renouvellement du certificat avec la commande certbot renew	98
Figure 54 – Entête Http complété du token coté frontend.....	99
Figure 55 – Informations des licences en pied de page	102
Figure 56 – Badges utilisés sur « readme » en frontend, afin de visualiser les technologies clé de DataTrainx	103
Figure 57 – Intégration continue sur DataTrainX.....	103
Figure 58 – Principe de badge, dernière release sur GitHub.....	104
Figure 59 – Contributions de type commit sur DataTrainX, source GitHub.....	105

Tableau 1 – Sources de dataset recherchées	40
Tableau 2 – Bibliothèques logicielles utilisées pour le CNN	49
Tableau 3 – Caractéristiques du fichier CSV FER2013	55
Tableau 4 – Benchmark mondial par modèle sur FER2013 et comparaison avec nos résultats sur 8 modèles	57
Tableau 5 – Résultats trouvés sur le modèle Deep sur une base de 3789 images tests.	62
Tableau 6 – Arborescence prévisionnelle de DataTrainX.....	64
Tableau 7 – Collection envisagée sur le sujet « trainer » JSON MongoDB	69
Tableau 8 – Dictionnaire des données DataTrainX	70
Tableau 9 – Composants utilisés en frontend avec NodeJS pour DataTrainX.....	73
Tableau 10 – Bibliothèques utilisées pour la capture webcam	84
Tableau 11 – Composants utilisés en backend avec NodeJS pour DataTrainX	89
Tableau 12 – Données dataProfil classifiées pour analyse.....	93
Tableau 13 – Failles de sécurité et stratégies appliquées	100
Code 1 – Préparation du jeu de données avant convolution, mappage des images avec les étiquettes en rouge	57
Code 2 – Import du modèle CNN VGG19 préformé par Keras - Class models.py	59
Code 3 – Première convolution sur une donnée d'entrée	59
Code 4 – Compilation du modèle avec l'optimiseur Adam et la perte d'entropie croisée catégorique	60
Code 5 – Entraînement du modèle CNN sur le jeu de données	60
Code 6 – Génération d'une matrice de confusion en utilisant le modèle sur les données tests	62
Code 7 – Chargement de la bibliothèque TensorflowJS en balise script	75
Code 8 – Utilisation de svelte et du slot pour le chargement dynamique des pages.....	76
Code 9 – Principe de chargement des templates sur DataTrainX avec le slot de Svelte	77
Code 10 – Magasin "store" de svelte permettant aux composants abonnés d'être avertis chaque fois que la valeur du magasin change	77
Code 11 – Mise à jour interactive du graphique sur la REF	78

Code 12 – fichier Logo.svelte, appelé à différent endroit de l'application. La couleur est paramétrable.....	83
Code 13 – Phase intermédiaire d'envoi des données sur mongoDB avec appel de la fonction updatePush().....	87
Code 14 – Système de routage Backend API REST.....	91
Code 15 – Principe du contrôleur et envoi des données en base des données.....	91
Code 16 – fonction preference() calcul des scores du profil.....	94
Code 17 – fonction emotions() classification des expressions.....	95

ANNEXES

1 – Etude de Coffield

Cohérence (l'instrument de mesure doit démontrer une cohérence interne ①), **Fiable** (même réponses sur au moins 2 tests ②), **Validité conceptuelle** (vérifier une validité théorique ③) et une **Validité prédictive** (permet de prévoir des faits à partir des éléments donnés ④).

Cette étude a ainsi pu déterminer la cohérence de ces tests et a permis de réaliser la classification suivante :

Facteurs génétiques	Caractéristiques cognitives	Élément de la personnalité stable	Préférences stables mais flexibles	Stratégies d'apprentissage
1	2	3	4	5
10 - Dunn et Dunn ① ② ③ ④ 9 - Gregorc ① ② ③ ④	12 - Riding ① ② ③ ④	3 - Apter ① ② ③ ④ 13 - Jackson ① ② ③ ④ 4 - Myers-Briggs ① ② ③ ④	1 - Allinsonet Hayes ① ② ③ ④ 5 - Herrmann ① ② ③ ④ 8 - Honeyet Mumford ① ② ③ ④ 7 - Kolb ① ② ③ ④ XX -Felderet Soloman	6 - Entwistle ① ② ③ ④ 11 - Sternberg ① ② ③ ④ 2 – Vermunt ① ② ③ ④
Pas d'éléments disponibles - Critère respecté - Critère non respecté				

2 – Modèle CNN Hyperparameter pour Fer2013

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 46, 46, 64)	640
conv2d_2 (Conv2D)	(None, 46, 46, 64)	36928
batch_normalization_1 (Batch Normalization)	(None, 46, 46, 64)	256
max_pooling2d_1 (MaxPooling2D)	(None, 23, 23, 64)	0
dropout_1 (Dropout)	(None, 23, 23, 64)	0
conv2d_3 (Conv2D)	(None, 23, 23, 128)	73856
batch_normalization_2 (Batch Normalization)	(None, 23, 23, 128)	512
conv2d_4 (Conv2D)	(None, 23, 23, 128)	147584
batch_normalization_3 (Batch Normalization)	(None, 23, 23, 128)	512
max_pooling2d_2 (MaxPooling2D)	(None, 11, 11, 128)	0
dropout_2 (Dropout)	(None, 11, 11, 128)	0
conv2d_5 (Conv2D)	(None, 11, 11, 256)	295168
batch_normalization_4 (Batch Normalization)	(None, 11, 11, 256)	1024
conv2d_6 (Conv2D)	(None, 11, 11, 256)	590080
batch_normalization_5 (Batch Normalization)	(None, 11, 11, 256)	1024
max_pooling2d_3 (MaxPooling2D)	(None, 5, 5, 256)	0
dropout_3 (Dropout)	(None, 5, 5, 256)	0
conv2d_7 (Conv2D)	(None, 5, 5, 512)	1180160
batch_normalization_6 (Batch Normalization)	(None, 5, 5, 512)	2048
conv2d_8 (Conv2D)	(None, 5, 5, 512)	2359808
batch_normalization_7 (Batch Normalization)	(None, 5, 5, 512)	2048
max_pooling2d_4 (MaxPooling2D)	(None, 2, 2, 512)	0
dropout_4 (Dropout)	(None, 2, 2, 512)	0
flatten_1 (Flatten)	(None, 2048)	0
dense_1 (Dense)	(None, 512)	1049088
dropout_5 (Dropout)	(None, 512)	0
dense_2 (Dense)	(None, 256)	131328
dropout_6 (Dropout)	(None, 256)	0
dense_3 (Dense)	(None, 128)	32896
dropout_7 (Dropout)	(None, 128)	0
dense_4 (Dense)	(None, 7)	903
Total params: 5,905,863		
Trainable params: 5,902,151		
Non-trainable params: 3,712		
Train on 29068 samples, validate on 3230 samples		

3 – Grille de scores Chapman, Kolb

	QUESTION NUMBER			
	2	7	1	5
4	13	3	9	
6	15	8	11	
10	16	12	19	
17	25	14	21	
23	28	18	27	
24	29	20	35	
32	31	22	37	
34	33	26	44	
38	36	30	49	
40	39	42	50	
43	41	47	53	
45	46	51	54	
48	52	57	56	
58	55	61	59	
64	60	63	65	
71	62	68	69	
72	66	75	70	
74	67	77	73	
79	76	78	80	
Totals:				
	Activist	Reflector	Theorist	Pragmatist

ACTIVIST	REFLECTOR	THEORIST	PRAGMATIST	
20	20	20	20	Very strong preference
19	19	19	19	
18	18	18	18	
17		17	17	
16		16		
15				Strong preference
12	17	15	16	
11	16	14	15	
	15			Moderate preference
10	14	13	14	
9	13	12	13	
8	12	11	12	
7				Low preference
6	11	10	11	
5	10	9	10	
4	9	8	9	Very low preference
3	8	7	8	
2	7	6	7	
1	6	5	6	
0	5	4	5	
	4	3	4	
	3	2	3	
	2	1	2	
	1	0	1	
	0		0	

RESUME

Dans l'apprentissage, une logique de la restitution qui prévaut encore sur une logique de la compréhension serait à l'origine de nombreux échecs de l'apprenant. Pour se comprendre, comprendre le monde et autrui, tout apprenant produit et met en œuvre des ressources métacognitives. On observera que cela fait appel à de nombreuses disciplines, comme la psychologie, la pédagogie ou les neurosciences. DataTrainX est un prototype de reconnaissance facial des émotions (REF) qui a pour objectif principal de s'intéresser à l'analyse des comportements de l'apprenant dans un but de neuropédagogie. Pour ce faire, nous utiliserons des algorithmes de deep-Learning et plus particulièrement les réseaux de neurones convolutifs (CNN) dans la reconnaissance des émotions de bases définies par le psychologue P. Ekman. L'idée est de se concentrer sur la démarche que privilégie chaque individu pour appréhender l'acte d'apprendre. Dans le projet DataTrainx l'utilisateur sera enregistré en vidéo et une comparaison sera effectuée entre la typologie trouvée par le questionnaire de Kolb et les émotions qu'il aura exprimées. L'objectif est de trouver une corrélation entre la typologie d'apprentissage et l'émotion de l'utilisateur et de démontrer comment mettre en place l'architecture fonctionnelle, logicielle et matérielle pour arriver à un résultat applicatif sur la REF axé sur un profil d'apprentissage.

Mots clés : CNN, REF, Deep learning, Tensorflow, Dataset, Convolution, NodeJS, apprentissage, neurosciences, neuropédagogie, Kolb, vidéo.

ABSTRACT

In learning, the logic of restitution which still prevails over the logic of comprehension would be at the origin of many failures of the learner. To understand themselves, the world and others, all learners produce and implement metacognitive resources. We can notice that this calls upon many disciplines, such as psychology, pedagogy or neurosciences. DataTrainX is a prototype of Facial Emotion Recognition (FER) which main objective is to focus on the analysis of any learner behavior for the purpose of neuropedagogy. In order to achieve this, we will use deep-Learning algorithms and more particularly convolutional neural networks (CNN) in recognition of basic emotions defined by the psychologist P. Ekman. The idea is to focus on the approach that each individual favors to apprehend the act of learning. In the DataTrainX project the user will be recorded on video and a comparison will be made between the typology found by the Kolb questionnaire and the emotions he has expressed. The objective is to find a correlation between the learning typology and the user's emotion. The purpose of this project is to focus concretely on the IT functional means to be implemented for this type of application and being to demonstrate how to set up the functional, software and hardware architecture to arrive at an application result of FER.

Key words : CNN, REF, Deep learning, Tensorflow, Dataset, Convolution, NodeJS, learning, neurosciences, neuropedagogy, Kolb, video.