EPITA Lyon INFOSPE

EyeContact
Cahier des charges

Auteur :
Rémi BARAZA
Lucas FAVRE
Patrick GASSELIN
Simon Zakhey

9 avril 2021

Table des matières

Chapitre 1

Présentation des membres du groupe

1.1 Patrick Gasselin

Patrick est né le 30 mars 2001 à Turin. Il est intéressé par l'informatique depuis qu'il était tout petit, depuis qu'il a reçus sa première PlayStation. Quand il jouait aux jeux, il s'intéressait toujours aux mécanismes cachés derrière. Même s'il n'est pas très fort en code, il reste tout de même émerveillé face aux grands développeurs qui réussissent à accomplir de grandes choses, notamment le créateur de Facebook qui grâce à un seul pc a réussis l'impossible : Relier des personnes qui ne sont pas dans la même localisation géographique. Patrick suit des cours en informatique dans une école d'ingénieur, EPITA. A ces débuts il avait peur de ne pas réussir ce cursus car il n'avait jamais codé auparavant et le fait de se retrouver dans une classe d'élèves qui codent depuis des années lui mettait une pression en plus sur les épaules. Mais ayant passé sa première année et entamant son quatrième semestre, il se sent de plus en plus à l'aise et veut accomplir de grande chose. Il n'a pas eu l'occasion de toucher au réseau de neurones dans son ancien projet (OCR) mais il compte se rattraper avec ce nouveau projet qu'il pense va lui permettre de prendre pleinement connaissance avec l'intelligence artificielle qui me passionne depuis longtemps. Il a hâte de commencer à comprendre les mathématiques cachées derrière tout ça et il a hâte que son réseau de neurone soit parfaitement performant pour leur projet.

1.2 Lucas Favre

Lucas FAVRE est né en Haute-Savoie et aime le rappeler. Il aime l'informatique et adore toucher à tout. Que ce soit du développement de site web ou d'algorithmes en C, Lucas est curieux et aime se documenter pour trouver des solutions à des problèmes. De plus, Lucas aime beaucoup les mathématiques et adore apprendre. Dans son temps libre, il aime faire du sport (vélo, ski, voile), sortir avec ses amis (quand il n'y a pas la Covid-19), jouer aux jeux vidéo et lire. Lucas est élève à EPITA depuis septembre 2018. A EPITA, excepté au cours du premier semestre, chaque élève doit réaliser des projets de programmation. Lors du projet du S2, Lucas à développer les menus, design et interfaces d'un jeu vidéo de type RPG, ainsi qu'un site web. Lors du projet de S3 qui avait pour thème de faire un OCR, Lucas à implémenter un réseau de neurones afin de reconnaitre les caractères d'un texte donné. Pour ce projet de S4, Lucas est motivé et compte bien faire de EyeContact un projet clair et complet.

1.3 Simon Zakhey

Simon Zakhey est étudiant en ingénierie informatique en 2ème année. Il est également très myope, et a régulièrement visité le cabinet d'ophtalmologie dans sa vie. Afin d'éviter cette inconvenance aux générations futures, Simon souhaite développer des moyens plus simples, et plus efficaces face à la crise sanitaire de tester les différentes facettes de son acuité visuelle.

1.4 Rémi Baraza

Je suis Rémi Baraza, élève en deuxieme année de l'Epita. Ayant deja réaliser deux projets informatiques pendant mon cursus, je suis particulièrement enthousiaste à l'idée d'en realiser un nouveau. La réalisation des deux derniers projets m'a permis de me familariser avec le travail d'equipe et d'améliorer ma rigueur dans mon travail. De plus,ayant approché le domaine de l'intelligence artificielle grâce au projet OCR, ce projet du S4 en est une nouvelle occasion.De plus, le côté "libre" apporte une dimension plus "ludique" que les deux projets précedents. En outre, j'ai toujours été intéressé par le milieu medical et par la dualité entre ce domaine là et celui de l'informatique, ce qui me motive d'avantage pour la réalisation du projet EyeContact

Chapitre 2

Présentation du projet

2.1 De quoi va traiter notre projet ?

Les troubles de la vison sont un problème qui touche une très large partie de la population. Ce sont les atteintes sensorielles les plus fréquentes.Ils concernent trois personnes sur quatre agées de plus de 20 ans et 97 pourcent des plus de 60 ans. De plus, nous savons tous qu'obtenir un rendez-medical dans de brefs delais n'est pas une mince affaire. L'équipe s'est fait la remarque qu'il manquait d'outils permettant à l'utilisateur de controler son état de santé de manière autonome, en tout cas, que ces applications n'etait pas très popluaire. Les troubles de la vision touchant beaucoup de personne, l'equipe s'est donc rapidement penché vers le domaine de l'ophtamologie. L'intelligence artificielle etait un domaine de l'informatique devenu incoutournable et indispensable dans dans la realisation de projets informatiques avec une telle utilité. Ainsi, l'equipe a eu pour idée d'integrer une intelligence artificielle realisant plusieurs actions parmis lesquelles la realisation autonome de differents tests visuels mais aussi la generation automatique de tests.En effet, de nombreux tests visuels existent destinées aux differents troubles de la vision et l'equipe s'est rendu compte que certains tests etaient complementement implementable en C. Ainsi, l'equipe s'est penchée vers la creation d'une intelligence artificielle qui realiserait plusieurs taches selon les tests : en effet, notre logiciel sera capable de génerer un test, en particulier le test d'acuité (E de snellen) qui s'adapte aux réponses du patient au fur et à mesure du test et fournit du prédiction du patient avant même qu'il l'ai finit. Ce qui nous pousse à réaliser une telle tâche est le fait que, en rendez vous medical chez l'ophtalmologue, il arrive souvent que des erreurs apparaissent, par exemple,le patient dit à voix haute la lettre sensée devinée, cette dernière est correcte, par hasard, alors qu'il ne l'a pas véritablement devinée.L'objectif sera de réaliser une intelligence artificielle capable de distinguer ce type d'erreur, s'adapter pour la suite, fournir une prédiction s'approchant au maximum du résultat final du test. Ce principe serait aussi utilisé pour le test de daltonisme. Nous allons alors vous présenter plus en detail les différents points de notre projet.

2.2 Répartition des tâches

Voici donc le tableau de la répartition des tâches :

Si l'une des tâches s'avère être plus difficile que prévu, cette répartition des tâches en susceptible de changer.

Partie	**Responsable**
SiteWeb	Lucas
Implémentation réseaux de neurone E de snellen	Patrick et Rémi
Implémentation réseaux de neurone daltonisme	Lucas et Simon
Interface graphique	Patrick Rémi et Simon

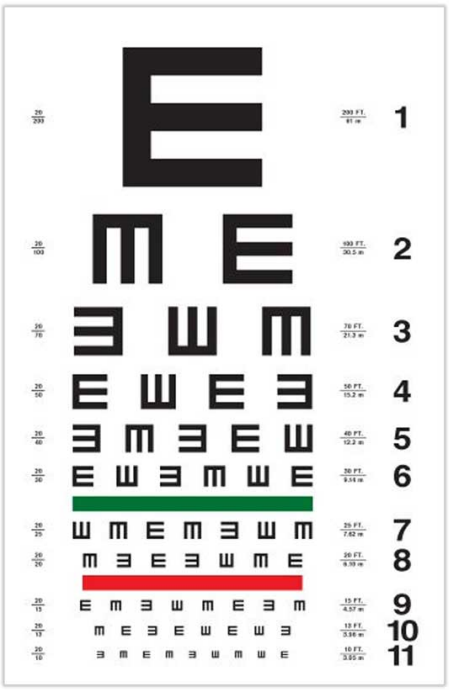
FIGURE 2.1 – Tableau de répartition des tâches

Chapitre 3

Explication scientifique

3.1 Présentation du test d'acuité

Nous vivons dans un monde où 2,2 milliards de personnes souffrent de troubles de la vision. Les tests chez les opticiens se multiplient pour trouver de façon efficace, rapide et fiable le trouble de la vision chez une personne. Nous avons tous types de tests allant du test de lecture au test de la pression oculaire. Etant donnée que nous sommes dans une école d'informatique, il était inévitable pour nous d'implémenter une intelligence artificielle qui sache résoudre des tests optiques. Patrick et Rémi vont donc coder une intelligence artificielle sachant réaliser le test d'acuité de manière la plus fiable possible. Le test d'acuité est un test qui est très simple. Nous allons afficher une forme (dans notre cas ça sera la lettre E), cette forme sera affichée selon les flèches directionnelle. Par exemple, si le E est affiché de tel sorte, cela voudra dire que le E est tourné vers la droite. Le test d'acuité visuelle est un test qui permet de mesurer la précision de la vue d'un individu et plus précisément c'est un test qui permet de voir si la vision de l'individus arrive à capter les plus petits détails. Il est dépendant de plusieurs facteurs neurologiques et optiques tels que la santé et le fonctionnement de la rétine ou bien de la capacité d'interprétation de notre cerveau. Une faible acuité visuelle est souvent dû à une maladie appelée amétropie. Cette maladie cause une déformation dans la rétine ce qui implique une mauvaise perception de la lumière par le cerveau et donc on mauvaise vue. Maintenant que nous savons un peu plus de quoi s'agit l'acuité visuelle, il est temp de nous intéresser au test qui permet de vérifier cette acuité visuelle chez l'individus. Parmi de nombreux tests, nous allons travailler sur le E de Snellen. Le E de Snellen est un test qui utilise un optotype orienté soit à gauche, à droite, en haut ou en bas. Un optotype est une forme spéciale qui permet de mesurer l'acuité visuelle d'un individu. Dans notre cas nous allons utiliser l'optotype E.

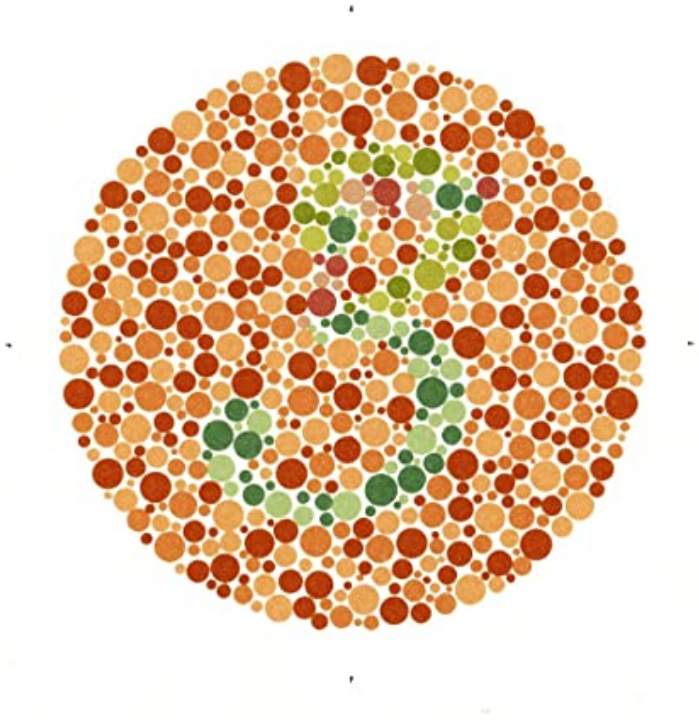


Le test est très simple. Nous allons défiler cet optotype à des positions différentes et nous allons aussi changer sa taille. Nous allons faire cela 20 fois et à la fin, en fonction des réponses de notre patient, nous pourront déterminer si ce dernier est affecté par un trouble visuel qui lui empêche d'avoir une bonne acuité visuelle et nous pourront alors lui prescrire un traitement et des exercices pour améliorer cela.

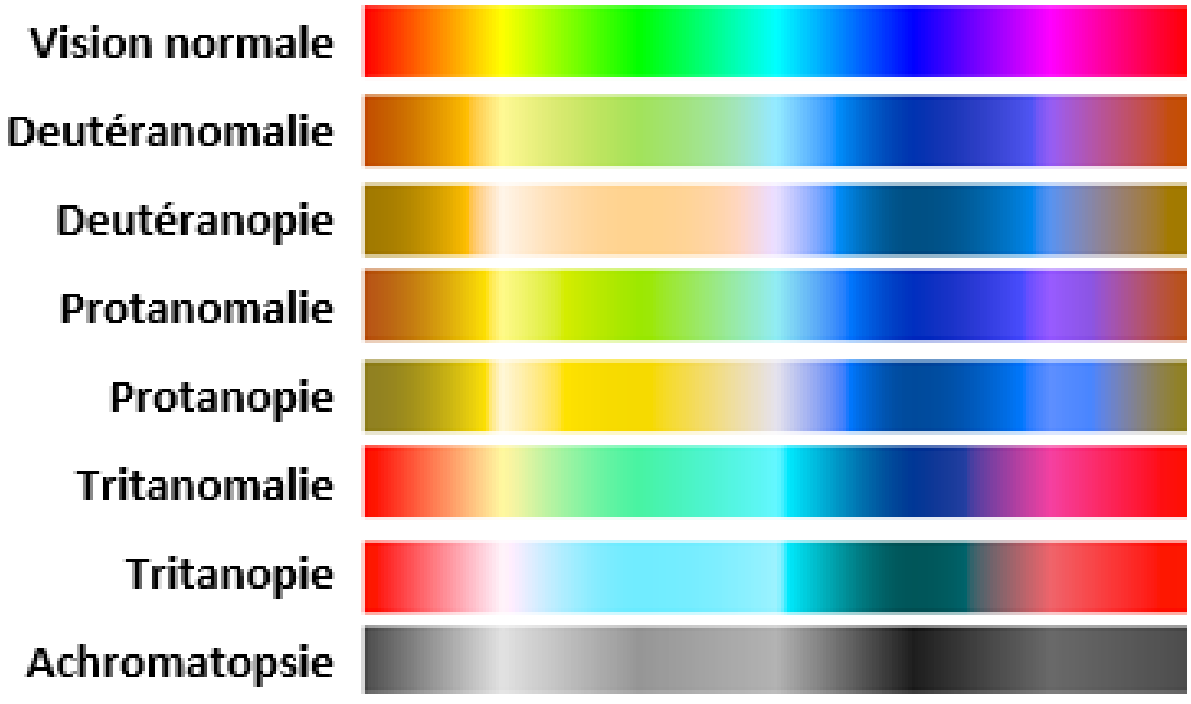
3.2 Présentation du dépistage du daltonisme

Le but de ce deuxième exercice est de créer, à l'aide d'algorithmes ou d'une intelligence artificielle, des tests afin de déceler le daltonisme chez l'Homme.

Pour se faire, nous allons nous inspirer de tests déjà existant : les tests de Ishihara ou plaques pseudo-isochromatiques. Ces plaques sont très célèbres et permettent de déceler facilement le daltonisme chez un large nombre de personne. Cette série de test est nommé selon le nom de leur créateur, le docteur Shinobu Ishihara. Ces tests seront publiés pour la première fois en 1917 et est composé de 38 planches colorées sur lesquelles un cercle constitué de points de différentes tailles et couleurs de teintes légèrement différentes et disposés de manière aléatoire. Si seul la planche numéro une est déchiffré correctement, le patient est surement atteint d'achromatopsie, soit l'incapacité à différencier les couleurs. Cette série de test permet de différencier les daltonismes les plus fréquents (seule la tritanopie qui est le fait de confondre le vert, le bleu et le violet, et la tritanomalie qui est une absence de perception des couleurs, ne sont pas détectés).



Comme dit précédemment, une planche du test est constituée de points de couleurs différentes, disposés aléatoirement, dans laquelle apparait une forme. Les couleurs sont choisies sur des « axes de confusion colorés » qui sont prédéterminés afin de déceler des types précis de daltonisme. Toutes les teintes apparaissent à plusieurs degrés de taille, saturations et de luminosité. Un ensemble de points reproduit une forme reconnaissable par la teinte mais est composée de plusieurs saturations et luminosités différentes de manière aléatoire. Un daltonien ne voyant pas une couleur ne pourra pas déchiffrer une forme. A l'inverse il existe des planches dans lesquelles l'homogénéité est utilisée pour « piéger » les personnes souffrant de daltonismes en leur faisant percevoir des formes qui ne sont pas existantes pour les sujets « normaux ».



Le daltonisme peut aussi être héréditaire et les confusions sont donc bien plus claires et

déterminées. Chaque groupe de planche doit être interprété de manière différente. En complément de ces planches, un appareil nommé la lanterne de Beyne peut être utilisé mais nous ne l'utiliserons pas dans notre projet. Dans cette partie, nous avons donc vu à quoi servaient ces tests et dans la partie suivante, nous expliquerons comment les utiliser dans notre projet.

Chapitre 4

Implémentation des intelligences artificielles

Notre projet se base sur deux intelligences artificielles qui vont être codé par deux équipes. Le but étant de réaliser, d'un côté le test de E de snellen avec une très fine marge d'erreur, et de l'autre côté générer des tests qui servent aux depistages du daltonismes pour pousser le despistage encore plus loins et dès lors être plus efficaces lors des consultations.

4.1 Implémentation de l'IA dans la résolution du test du E de snellen

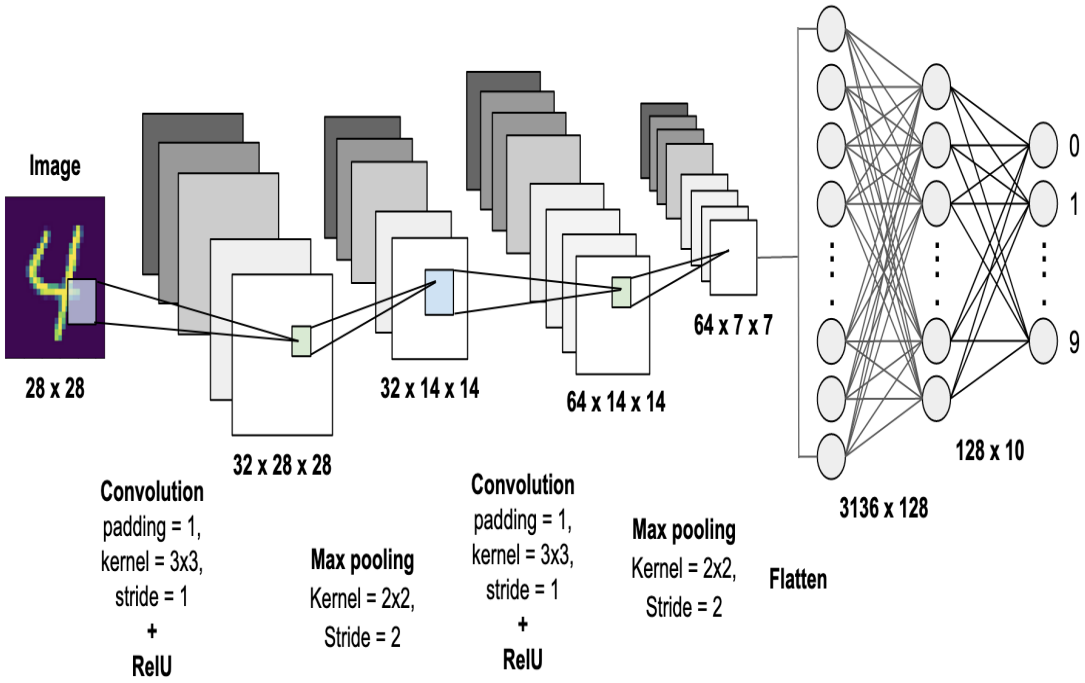
Notre exercice consiste en reproduire un test de Snellen mais plus efficace grâce à l'intelligence artificielle. Aujourd'hui, les tests d'acuité visuelle effectué chez un ophtalmologue ne sont pas du tout précis. En effet, quand les lettres deviennent petites et que le patient commence à recontrer des difficultés à lire la lettre, il aura tendance à deviner cette dernière. Or, le médecin ne peut pas savoir si le patient a eu bon car il a réussi à distinguer la lettre ou parcequ'il l'a deviné au hasard. Notre objectif avec Rémi est de réduire cette erreur humaine et de proposer un exercice plus fiable.

Nous sommes partis de l'idée que nous allons utiliser une interface graphique qui présentera au milieu le " C de snellen ". Le C de snellen est une lettre qui sera orientée vers la gauche, droite, en haut et en bas, et aura une taille différente en fonction de la réponse du patient. Le patient devra juste indiquer à l'aider des boutons, l'orientation de la lettre. Si le patient a juste, nous allons modifier la taille de la lettre et la mettre en plus petit. Si le patient a faux, nous allons garder la même taille de la lettre mais nous allons l'orienter différemment.

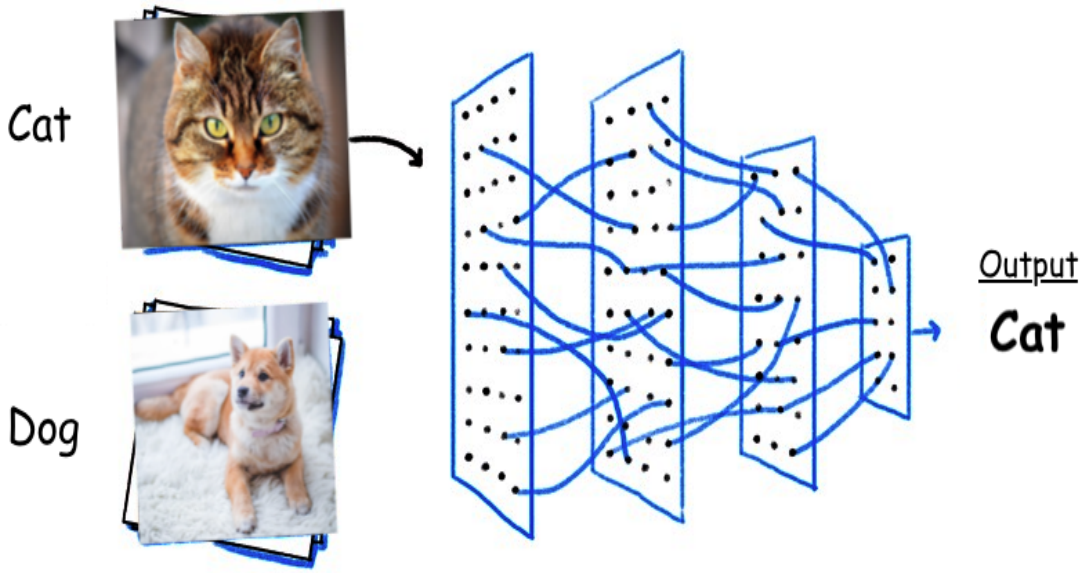
Maintenant, on doit trouver une solution pour l'image. Au début nous avons pensé à donner beaucoup d'images avec des tailles différents et des orientations différentes du C de snellen à notre programme, on les étiquette avec l'orientation de la lettre et on a juste à comparer l'étiquette de l'image qui est présentée au patient et voir si ce dernier a eu la bonne réponse. Le problème c'est que cela va nous occuper beaucoup trop de mémoire et cela n'est pas efficace pour un exercice que nous voulons rendre presque " optimale ". Alors nous avons pensé à une deuxième solution. Cette dernière s'agit de travailler avec une seule image. Cette image nous allons la retourner et la redimensionner selon les réponses de notre patient. Or, un gros problème apparaît : Comment comparer la réponse du patient avec l'orientation de l'image ? Nous ne pouvons pas étiqueter les images car elles sont créées à l'instant t. Pour résoudre ce problème, nous avons pensé à coder un réseau de neurones qui reconnaîtra l'orientation de l'image au moment de la création de l'image. Ce réseau de neurones est appelé : classification d'images. Le principe est très simple, nous allons donner une image à notre réseau et ce dernier va s'occuper de le classer dans une catégorie. Les catégories sont numérotées : 0 pour indiquer que l'image est orientée vers le bas, 1 pour indiquer que l'image est orientée vers la gauche et ainsi de suite. Donc nous allons passer une image à notre réseau et ce dernier va nous donner la classe à laquelle elle appartient. Nous avons donc trouvé un moyen de trouver l'orientation

de notre C de snellen à l'instant où la nouvelle image est créée, et ce résultat nous pouvons le comparer à la réponse du patient. Notre exercice va donc travailler avec une seule et unique image que nous allons modifier.

Patrick s'est occupé du réseau de neurone. Au début, il a commencé à se documenter sur le réseau de neurone car il n'en avait jamais fait et il a trouvé une vidéo qui faisait une introduction dans ce domaine en proposant de coder son premier réseau. Le réseau était plutôt simple, nous avions à disposition des pétales roses et des pétales bleus, la longueur et la largeur de chaque pétale. Le but du réseau est le suivant : Suivant une longueur et une largeur donnée, deviner la couleur du pétale. Il a donc commencé à comprendre la propagation, la rétropropagation, les fonctions d'activations etc... Une fois le réseau terminé, il devait trouver maintenant un moyen d'implémenter un réseau qui puisse, à partir d'une image, trouver son orientation. Au début il pensait à des calculs d'angles sur l'image pour trouver la direction mais il a vite changé d'approche lorsqu'il a vu que cette solution n'était pas la plus optimale. Il a ensuite pensé à l'OCR qui s'occupait de reconnaître les caractères. Or ce n'est pas vraiment reconnaître le caractère qu'il lui fallait, c'est plus classer le caractère qu'il fallait. Donc au début il s'est documenté sur le convolutional network.



Le principe étant le suivant : on analyse une image, on en retire des informations (extraction de pixel dans une zone donnée) et à la fin de cette extraction, il y a le MaxPool qui va prendre le pixel ayant le plus grand chiffre et il va mettre ce dernier dans une case. La méthode marche bien avec des images compliquées mais là nous voulons juste classer un C qui est déjà en noir et blanc. Donc le convolutional network semble trop gourmand pour ce que l'on veut faire. En se documentant sur le web, il tombe sur " image classification "

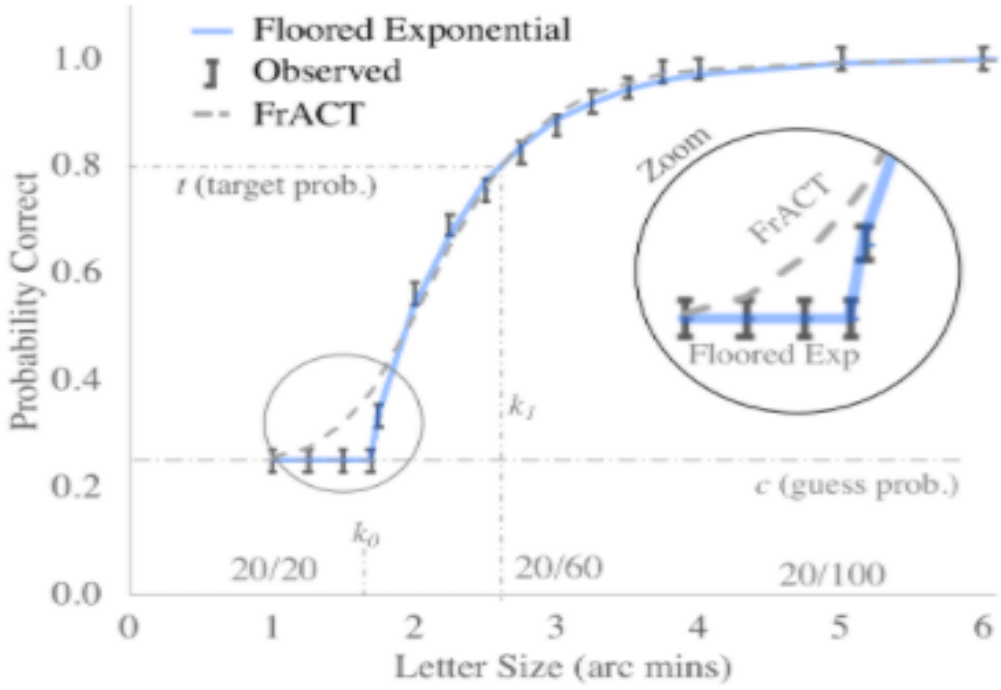


qui consiste à étiqueter une image. Par exemple, si je passe comme image une image d'un chien, le réseau va me l'étiqueter en tant que chien. Le principe colle parfaitement avec ce que Patrick veut faire. Il veut un réseau qui puisse prendre en paramètre une image et qu'ensuite se réseau étiquette l'image avec le chiffre qui lui correspond. Par exemple, si l'image est orientée vers le bas, le réseau va l'étiqueter avec un 0 (car notre 0 est l'étiquette qui correspond à la direction " bas " de notre image). Maintenant la méthode trouvée, il faut la coder. Pour commencer, il faut créer une base de données. Notre base de données ne va pas être super grande, nous allons avoir 4 dossier numéroté de 0 à 3, chaque dossier correspondant à l'orientation de notre image. Notre réseau va prendre en paramètre des C de snellen de taille différente, du coup dans chacun de nos dossiers, nous allons retrouver des C de tailles différents (la plus grandes étant de 128x128, nous allons ensuite à chaque fois diviser par deux pour à la fin se retrouver avec une toute petite lettre de 7x7). Donc notre base de données est composée de la manière suivante : Nous avons 4 dossiers, chacun de ses dossiers sont étiquetés, et dans chaque dossier il y a 5 images du C de snellen orientées de la même direction où juste la taille de l'image va changer. Maintenant que nous avons notre base de données, il faut l'importer dans notre code. Pour commencer, il faut extraire chaque image de notre dossier ainsi que leurs étiquettes. Pour se faire nous avons une première boucle for qui va parcourir chaque dossier, puis une deuxième boucle for qui va parcourir chaque image de notre dossier. Dans cette boucle for, nous allons redimensionner chaque image en 32x32 car c'est la taille la plus idéale pour travailler avec. A la fin nous aurons une liste d'images ainsi qu'une liste avec les étiquettes de chaque image. Par exemple, si le premier élément de notre liste est un C de snellen orienté vers le bas, alors le premier élément de ma liste d'étiquette sera un 0 et ainsi de suite. Maintenant il faut convertir ces listes en des arrays. Pour se faire nous allons nous servir de la bibliothèque numpy. Une fois que j'ai mes deux array, il faut que je les divise en plusieurs liste. Nous allons donc introduire 4 nouvelles variables. X.train (array d'images que nous allons passer à notre réseau pour qu'il s'entraîne), X.test (array d'images que nous allons utiliser pour tester notre réseau), Y.train(array des étiquettes de chaque image dans X.train que nous allons passer à notre réseau pour qu'il s'entraîne) et Y.test(array des étiquettes de chaque image dans X.test pour vérifier si l'étiquette proposée par notre réseau coïncide avec l'étiquette contenu dans cet array). Nous avons distribué chaque image de façon aléatoire et les array X.train et X.test n'auront pas la même taille. En effet, X.test n'aura que 20 pourcent des données (soit 4 images)

et X.train aura 80 pourcent des données (soit 16 images). Maintenant que nous avons créé une base de données, que nous l'avons importé et que nous l'avons séparé correctement, nous pouvons passer au traitement de l'image. Même si notre image est en noir et blanc, cela ne signifie pas que l'image n'est pas en RGB. Or, pour traiter notre image, il faut qu'elle soit en noir et blanc, c'est à dire qu'elle n'ait qu'un seul channel.
Pour se faire nous allons juste passer chaque image de notre array X.train dans une fonction appelée PreProcessing qui va elle se charger de convertir les images RGB en noir et blanc. Nous avons donc un array X.train qui contient des images qui ne présente qu'un seul channel. Maintenant que nous avons traité nos images, nous pouvons commencer à coder notre réseau. Pour commencer, nous allons " aplatir ". Nous avons donc une première couche qui sera celle des " input " qui sera de taille l'array X.train que nous venons d'aplatir. Ensuite, nous allons placer une hidden layer de taille 128 neurones. Nous avons choisi 128 neurones car cela semblait être le nombre de neurones nécessaire pour que notre programme soit efficace. Si nous en mettons trop peu, il faudra d'avantage entraîner notre programme, si nous en mettons trop, il faudra utiliser davantage de mémoire pour entraîner notre programme. Nous avons donc une hidden layer de taille 128 et nous allons utiliser la fonction " ReLU " comme fonction d'activation de nos neurones. Nous choisissons la fonction " ReLU " au lieu de la " sigmoïd " par soucis d'efficacité (notre but étant d'entraîner notre programme le moins possible et d'obtenir des résultats fiables, nous visons une précision au-delà de 98 pourcent). Pour que notre réseau soit encore plus efficace, nous allons rajouter une deuxième hidden layer avec les mêmes paramètres que la hidden layer précédent. Ensuite nous allons terminer notre modèle avec une dernière couche qui représentera notre couche " output ". Cette couche aura que 4 neurones : nos 4 étiquettes. On lui passe en paramètre la fonction d'activation SoftMax. Nous avons fini de d'implémenter notre réseau de neurone, passons à l'explication de son entraînement. Notre réseau a pour l'instant 4 couches et chaque couche est reliées par des synapses qui ont tous des poids différents. Nous allons entraîner notre réseau avec notre array X.train (valeur d'entrée) et notre array Y.train (valeur de sortie). Pendant l'entraînement de notre réseau, ce dernier va comparer valeur de sorties avec les étiquettes de chaque image et il va lui-même modifier les poids de ses synapses grâce à la rétropagation jusqu'à ce qu'il atteigne une précision de plus de 98 pourcent. Nous allons donc entraîner notre réseau au minium 15 fois. Nous pouvons l'entraîner 20 fois pour obtenir une précision de 1. Maintenant nous avons un réseau de neurone capable de classer une image dans une catégorie en fonction de son orientation.

Nous avons donc une interface est un réseau de neurone qui arrive à étiqueter une image selon son orientation. Maintenant, nous devons trouver une solution pour calculer la probabilité que le patient a une bonne réponse mais qu'il l'a deviné sans savoir l'orientation du C. Nous devons aussi gérer le cas où le patient clic sans faire exprès sur une mauvaise réponse et nous devons savoir à quel moment nous devons réduire la taille de l'image pour qu'on obtienne un test d'acuité visuelle complet, fiable et efficace.

Tout d'abord, nous avons besoins une notion importante qui va nous servir tout le long. Notre algorithme va utiliser une fonction appelée " Visual Response Function (VRF) ". La fonction VRF est une fonction qui détermine la probabilité qu'un humain devine une lettre de taille k. Après de nombreux test sur des patients, des recherches ont aboutis à 2 VRF. La première VRF étant la fonction logistique. Or cette fonction rencontre des problèmes lorsque la lettre devient trop petite et que les patients commencent à " deviner " la lettre. Les chercheurs ont donc abouti à une nouvelle fonction qui est la fonction " Flood exponential "

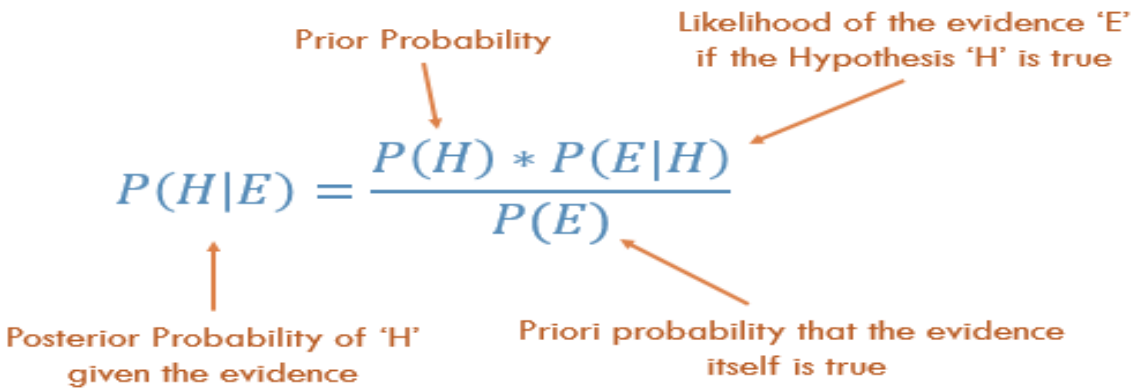


Floored Exponential
A Floored Exponential is a maximum between a constant floor and an exponential function. For visual acuity we parameterise it as:

$$v(x, k_0, k_1) = \max\left\{c,\ 1 - (1-c)\left(\frac{1-\tau}{1-c}\right)^{\frac{x-k_0}{k_1-k_0}}\right\},$$

where x is the font size (in arcmins) of the letter being tested, c is the probability of a correct answer when guessing randomly, and k_0 is the font size at which a patient can start to discern visual information. In an acuity test, we are trying to identify k_1: this is the font size at which a patient can see with probability τ, where τ is some predefined constant "target probability" specific to the type of eye exam. In this paper we use $\tau = 0.80$ which means at font size k_1, a patient can correctly guess letters with 80% probability.

qui elle arrive à nous donner des probabilités de bonnes réponses même avec des petites lettres. Donc le VRF que nous allons utiliser sera la fonction Floored Exponential. Ensuite, notre programme va toujours garder une trace des réponses de notre patient sous forme de liste de tuples où le premier élément sera la taille de la lettre et le deuxième élément si le patient a répondu correctement ou pas. Dans notre fonction Floored Exponential, nous avons une variable k1 qui est la taille de notre image et une variable téta qui est la probabilité que notre image de taille k1 soit devinée correctement, cette variable est fixée à 0,80, soit 80 pourcent de chance que le patient devine correctement notre image de taille k1. Nous avons aussi une variable k0 qui elle va déterminée la taille de l'image à laquelle notre patient commence à discerner l'orientation de notre C (taille de la première image que le patient aura devinée correctement). Pour déterminer la taille de notre prochaine lettre, nous allons utiliser une méthode dites " Posterior Probability Matching ".



Cette méthode consiste à afficher des lettres de tailles complétement au hasard et plus le patient avancera dans le test, plus l'algorithme comprendra la taille de la prochaine lettre qu'il faudra montrer au patient pour avoir un résultat précis. Ensuite, il nous reste plus qu'à calculer la probabilité que le patient est cliqué sur une réponse sans faire exprès. Pour faire cela, nous allons remplacer notre VRF Floored exponential par un autre VRF qui est la " Slip Porbability "

To account for this slip probability, We replace the VRF $v(x)$ with $v_s(x)$ where s is the slip probability:

$$v_s(x) = s \cdot c + (1 - s) \cdot v(x).$$

qui inclue dans sa formule la Floored exponential. Cet atout nous permettra d'effectuer un test encore plus précis vus que nous allons ignorer les erreurs humaines.

Pour la prochaine soutenance, nous avons pour objectif de traduire notre réseau de neurone en code python en code C et de le relier à l'interface de Rémi pour déjà effectuer un test d'acuité visuel classique en comparant les réponses de notre réseau et de notre patient. Nous allons aussi commencer à mettre en place les calculs de probabilité qu'une personne ai cliqué sur une réponse sans faire exprès et nous allons essayer d'implémenter les calculs pour trouver la taille de la prochaine image que nous allons montrer au patient pour que le test soit vraiment complet.

4.2 La génération de test servant au despistage du daltonisme

4.3 Test de daltonisme

dans l'image du test plus haut, on remarque :

— les points sont de taille et de couleur variable, dans une certaine mesure
— les points ne se chevauchent pas mais sont proches les uns des autres
— l'image contient plusieurs caractères

On souhaite créer un programme qui, à partir d'un texte et du type de daltonisme que l'on souhaite tester, génère un test fonctionnel contenant les caractères entrés.

4.3.1 Procédé

afin de créer un test semblable à celui vu précedemment, il y a une séries d'étapes algorithmiques à appliquer :

— Créer une matrice d'une taille adaptée

— tracer dans cette matrice l'empreinte du ou des caractères souhaités
— générer 2 palettes de couleurs adaptées au test que l'on effectue
— remplir la matrice de points. S'ils sont dans l'empreinte d'un caractère, on les colore avec la première palette, sinon avec la deuxième
— une fois la matrice complétée, on créé une image utilisable à partir des informations qu'elle contient.

4.3.2 Implémentation (soutenance 1)

Pour cette première soutenance, j'ai créé un programme python exécutant au moins une version primitive des étapes détaillées ci-dessus. La version finale sera programmée en langage C.

4.3.2.1 structures de données : matrices

afin de pouvoir "tracer" un caractère dans une matrice, mais également utiliser les informations contenues dans celle-ci pour créer une image après le traitement, j'ai choisi d'utiliser deux matrices de même taille. La première, que je désignerai ci-arpès "matrice RGB" ou simplement "matrice", contient les informations de coloration au format RGB, initialisée uniformément grise. C'est cette matrice qui servira à la construction de l'image retour. Il est possible que la suite du projet nécessite une transition au format RGBA, afin de faire varier la transparence des contours des points colorés.

L'autre matrice, que je désignerai ci-après "vecteur" ou "matrice de construction", contient les informations nécessaires au bon fonctionnement du programme et ne sera pas utilisée en sortie. Elle a les mêmes dimensions que la matrices RGB, mais ses éléments sont des couples de booléens dont je détaillerai l'utilité plus bas.

4.3.2.2 structures de données : bases de données

il existe 3 types et 7 sous-types de daltonisme. certains d'entre eux nécessitent plusieurs palettes de couleurs différentes pour être testés. Il me faut donc stocker ces types de daltonisme et les intervalles de couleur qui leur correspondent. La quantité d'information étant raisonnable, j'ai décidé d'utiliser un simple algorithme de pattern matching. Afin d'optimiser les performances du programme, on peut imaginer réordonner les différents cas en fonction du nombre de sollicitations. Cette base de données est codée par la fonction color-db.

Il est évident qu'il existe un grand nombre de caractères possibles, et qu'il faut être capable de tous les tracer dans la matrice de construction. Deux choix se présentent :

— Stocker une "matrice-type" pour chaque caractère proposé par l'application, dans laquelle un caractère spécifique est déjà tracé. On l'utilisera comme matrice de construction. Cette implémentation est simple, mais couteuse en mémoire et inflexible. En effet, si l'on créée des matrices à l'avance, leur taille ne sera pas ajustable, et il faudra travailler avec des dimensions prédéfinies.

— Stocker une "fonction de traçage" pour chaque caractère proposé, qui trace mathématiquement un caractère par des opérations sur les coordonnées. Cette implémentation a l'avantage de libérer de l'espace de stockage et d'être utilisable sur toutes tailles de matrices. Cependant, l'implémentation est plus complexe et les caractères tracés géométriquement ne sont pas esthétiquement plaisants.

ces informations sont stockées dans la fonction write-char et ses sous-fonctions.

Dans la matrice de construction, on donne au deuxième booléen d'un élément la valeur "False" si ses coordonnées sont à l'intérieur du caractère tracé.

Ces deux bases de données sont à ce jour incomplètes, puisque la version finale sera codée sur un autre langage.

4.3.3 pointillage et coloration

Peu importe l'implémentation choisie, la prochaine étape consiste, à partir de la matrice de construction initialisée, à placer des points de couleurs correspondantes sur la matrice RGB.

4.3.3.1 placement d'un point

Afin de placer un point à des coordonnées données sur la matrice RGB, on doit vérifier deux informations :

— ces coordonnées vont-elles causer un chevauchement entre plusieurs points ? Pour le savoir, on récupère la première valeur de l'élément de la matrice de construction. Si la valeur est "False", alors on choisit d'autres coordonnées. La prévention du chevauchement n'est à ce jour pas totalement fonctionnelle, avec des difficultés sur l'axe y.

— Ces coordonnées sont-elles contenues dans la trace du caractère ? Si oui, on change de couleur. Le traçage des caractères dans la matrice n'est pas encore fonctionnel.

Afin de choisir à quelles coordonnées tracer un point, on se place à des coordonnées telles que la première valeur de l'élément de la matrice de construction soit "True", et on trace un point de la couleur appropriée. La taille du point est déterminée aléatoirement entre des bornes définies par une opération mathématique sur la taille de l'image.

Pour tracer un point de rayon R aux coordonnées (x,y)de la matrice, on parcourt une sous matrice de taille 2Rx2R centrée sur (x,y), et pour chaque point de cette sous-matrice, si le carré de la distance entre ce point et (x,y) est inférieur ou égal à R au carré, on le colorie, puis on donne à la première partie de l'élément de mêmes coordonnées de la matrice de construction la valeur "False", indiquant qu'on ne peut plus tracer de point à cet endroit. c'est ce qu'il se passe dans la fonction make-dot

4.3.3.2 complétion

Pour compléter la tâche, il suffit de répéter l'opération détaillée ci-dessus en choisissant à chaque fois de nouvelles coordonnées. Ce choix peut être fait itérativement ou intégrer une part de RNG. Dans cette première version, la séléction est purement itérative. Cependant cette méthode sera amenée à changer dans le futur. Lorsque l'on ne trouve plus de coordonnées valides dans la matrice de construction, on a terminé, on met donc fin à l'exécution. L'ensemble de ce processus est codé par la fonction dot-up.

4.3.4 sortie

Une fois la matrice RGB traitée, la fonction mat2img créera une image de la taille des dimensions de la matrice, en considérant chaque élément comme un pixel. Une fois l'image créée, on la renvoie. mettant fin au processus. La fonction build-test réunit et lie toutes les étapes détaillées précedemment, et renvoie à partir d'un type de daltonisme, d'une taille d'image et d'un caractère une image fonctionnelle de test.

4.4 Implémentation IA dans le test de daltonisme

Nous avons remanié notre cahier des charges afin d'avoir plus de spécifications sur l'implémentation de notre exercice. Lors de cette première période de travail étant donné que j'avais déjà une partie de création du site assez importante (je devais reprendre ma connaissance du langage HTML à zéro) je ne m'étais pas fixé d'objectifs vraiment précis si ce n'est le début d'une implémentation d'arbres en C.

La première partie de mon travail à donc été une partie de recherche d'information. Mon objectif global sur cette partie du projet étant d'implémenter un programme permettant de savoir si les réponses données à un test sont dues à la chance, au hasard, ou bien car le sujet testé connaissait la bonne réponse. Avec cette problématique en tête, j'ai tout d'abord eu un peu peur. En effet, une telle intelligence artificielle me semblait (et me semble toujours d'ailleurs) assez difficile à coder. Cette intelligence sera donc la majeure partie de mon travail pour les soutenances à venir mais nous allons un peu vite en besogne. Je me suis donc occupé d'implémenter une architecture pour créer des arbres en C. Pour cela, je me suis inspiré du cours que nous avons pu avoir au S3 sur les listes chainées en C. Une autre fonction implémenter est une fonction d'ajout d'un nœud dans un arbre. J'ai passée une partie de mon temps à chercher quelles autres fonctions aurait pu être utiles à implémenter en C. Finalement, je vous présente, pour le moment une fonction permettant de créer un nœud et une fonction permettant d'ajouter un nœud à un arbre. Cette partie de mon travail n'a pas été la plus complexe même si elle s'est révélée « piégeuse » car elle me laissait beaucoup de libertés. Je n'ai pour le moment pas trouvé d'autres fonctions à ajouter et je dois réfléchir à quels problèmes je risque de faire face avec ces deux fonctions. Pour cette partie de mon travail, la partie de recherche à été minime, étant donné que j'avais une idée dans ce en quoi je me lançais. Cependant, pour la dernière partie de mon travail, la partie recherche à été très importante et je suis d'ailleurs encore en recherche d'informations alors que j'ai déjà commencé à coder. Cette dernière partie est donc la partie relative à l'intelligence artificielle permettant de détecter si un utilisateur à eu de la chance en répondant à une question. Pour cela, on peut s'attarder sur deux types d'IA : les CART machine learning qui sont, dans les grandes lignes, des arbres de décisions et les naive bayes machine learning. Pour cette première partie de mon travail je me suis donc penché sur les CART machine learning et ce sera donc ce de quoi je vais vous parlez dans ce rapport de première soutenance. Un CART (pour Classification and regression Tree) est un modèle d'arbre permettant d'expliquer comment on trouve la valeur d'une variable basée sur les valeurs d'autres variables. Les CART sont donc des arbres de décision dans lesquelles chaque nœud contiens une prédiction pour une variable de sortie. Un exemple d'arbre de décision peut être le suivant :

Dans un arbre de décision comme celui-ci, on voit très clairement que c'est « facile » d'arriver à une conclusion en «répondant» à des questions. Par exemple si un sujet mesure plus d'un mètre quatre-vingts est directement classé homme. En répondant à la question sur la taille on a trouvé une réponse. Dans cet exemple, l'arbre de décision est simple mais apporte une solution concrète au problème donné. L'arbre de cet exemple est binaire et c'est ce type de modèle que nous utiliserons par la suite. Certaines équations peuvent rappeler les fonctions utilisées lors de l'établissement du réseau de neurones de l'OCR car celui si utilisait la rétropropagation du gradient qui peut être aussi utilisé dans ce genre d'arbre. On est dans le cas d'un apprentissage

supervisé. Il existe cependant deux types d'arbres de décision : les arbres de classification et les arbres de régression. Pour cette première soutenance, j'ai donc fait beaucoup de recherche, ce qui m'a permis d'implémenter un exemple d'arbre de classification. Le problème principal peut importe le type d'arbre est le choix des critères de segmentation. En effet, le choix des critères d'évaluation des partitions est important car c'est ce critère qui va permettre de « tracer » notre chemin dans l'arbre et donc d'atterrir sur une feuille de notre arbre précisément. Il existe de nombreux critères les plus utilisés sont l'entropie de Shannon ou l'indice de diversité de Gini. Dans le cas d'un algorithme CART, on utilise l'indice de diversité de Gini.

Le but de l'indice de diversité de Gini est de mesurer à quelle fréquence un élément quelconque de notre ensemble est mal classé si son étiquette est choisie aléatoirement. La valeur de l'indice atteint 0 quand tous les éléments de l'ensemble on la même étiquette. Pour aller plus loin, on pourrait parler du problème du surajustement des modèles et donc débattre de la taille d'un arbre mais je ne maitrise encore pas assez bien le sujet. En cherchant sur Internet, je suis donc parvenu à trouver de nombreux exemples de CART machine learning. Je suis parvenu en m'inspirant de plusieurs exemples d'implémenter un arbre de décision dans le langage python sans utiliser de pacage particulier même si cela à été plutôt dur alors qu'il ne s'agisse que de la première étape. Cependant, ce début de code me permet d'avoir une idée de ce que je dois faire pour la prochaine soutenance : implémenter un tel arbre en C et utiliser ce dernier pour prédire des résultats à partir d'autres variables.

$$I_G(f) = \sum_{i=1}^{m} f_i(1 - f_i) = \sum_{i=1}^{m} (f_i - f_i^2) = \sum_{i=1}^{m} f_i - \sum_{i=1}^{m} f_i^2 = 1 - \sum_{i=1}^{m} f_i^2$$

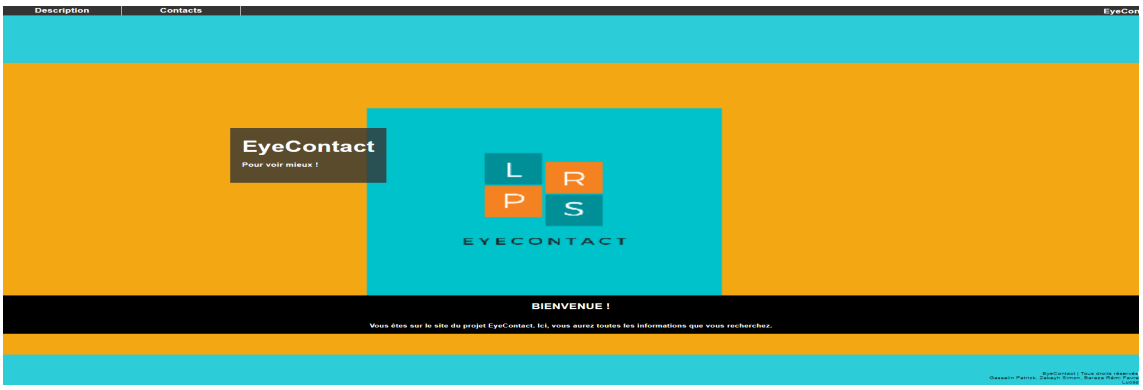
Chapitre 5

Modélisation

5.1 Site web

Depuis le mois de février 2021, nous avons entamé le quatrième semestre de notre scolarité à l'EPITA. Pendant ce quatrième semestre, nous devons coder un projet en C et nous sommes libres sur le choix du sujet. Nous nous sommes donc lancés dans l'établissement d'un logiciel permettant de détecter d'éventuels problèmes de vues. Durant cette première période de travail, je me suis occupé de diverses aspects très différents du projet. Dans un premier temps, je vais donc vous parler du site Internet que j'ai entièrement réalisé en code HTML et CSS, ce qui m'a pris beaucoup de temps puis dans une seconde partie, je vous parlerai des premières esquisses de codes de l'IA ainsi que de l'établissement de la classe arbre en C.

La première partie de mon travail à donc été d'établir le site Internet dans les grandes lignes. En effet, le site Internet présenté dans ce document est la première version de mon site. Il sera par la suite renfloué en informations diverses (différentes étapes de créations du projet, visuels utilisés ...) et j'essaierai d'apporter de meilleurs visuels afin de rendre le site plus professionnel. Comme je l'ai dit précédemment, j'ai élaboré le site en utilisant du code HTML et du code CSS et SublimeText. Par le passé, j'ai déjà pu créer un site Internet mais j'avais oublié une bonne partie de mes connaissances en HTML. J'ai donc tout repris de zéro. Pour se faire, je me suis aidé du cours disponible sur OpenClassrooms ainsi que différents sites Internet répertoriant des balises (les balises étant l'outil principal du code HTML) afin de retrouver mes connaissances passées. De ce fait, j'ai d'abord établi un premier site très archaïque (page unique, pas de fond d'écran ...). Ce site ne me plaisait pas, il me paraissait trop simple, ne m'attirait pas visuellement et comportait peu de code. Je me suis donc penché sur le code CSS de mon site qui était très faible (voir trop faible). En me penchant sur le code CSS, je me suis rappelé qu'utiliser les ID dans mes balises me permettait de « décrire » plus précisément ce que je voulais pour mon site. J'ai donc créé des « boites » pour mettre nos textes car c'est une mise en page que j'apprécie et qui me semble approprié au propos ; c'est-à-dire un site fonctionnel permettant de suivre l'avancement du projet. Rémi m'a, par la suite, envoyé ses visuels qu'il a utilisé pour l'interface, je les ai donc utilisés dans mon site comme fond d'écran. Voici un aperçu visuel du site :



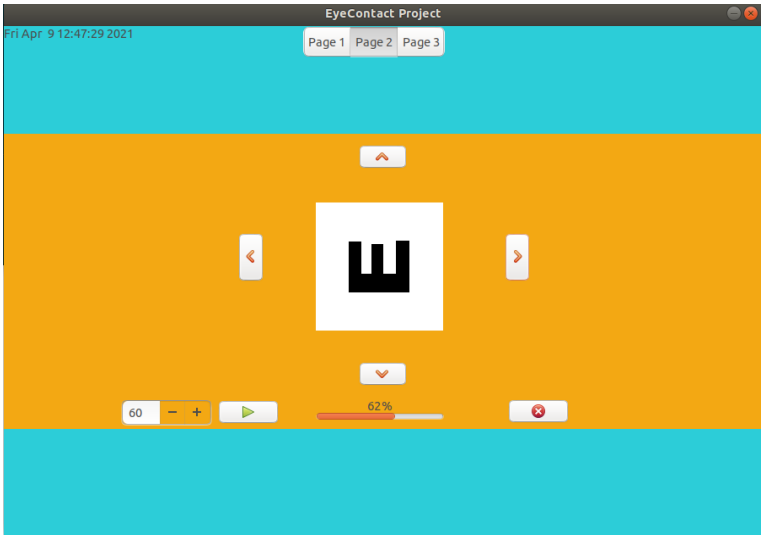
Ce site me plaisait assez, mais je me demandais s'il ne fallait pas ajouter quelque chose. Je me suis donc essayé à appliquer une ombre sur mon site afin de le rendre plus esthétique. Cette ombre m'a posée de nombreux problèmes. En effet, implémenter l'ombre à été un travail très méticuleux. L'ombre repose sur toutes les parties de mon code que ce soit le HTML et le CSS. En modifiant un peu une partie ou l'autre cela modifiait mon visuel du tout au tout. J'ai perdu beaucoup de temps sur cette partie pour un résultat qui, finalement, ne me satisfait pas. Je vous montre ci-dessous, une des pages de mon site sur lesquelles j'ai laissé cette ombre. Pour le moment, je n'ai pas encore tranché sur le fait de laisser ou non cette ombre.

5.2 interface graphique : E de Snellen

L'un des défis que s'était lancé l'équipe d'EyeContact fût de réaliser une interface des plus travaillée, cohérente, et adaptée au projet. Ainsi, Rémi s'est occupé de cette tache en recherchant d'abord parmi les outils qui lui sont proposés en C lequel correspondrait le plus au développement de cette interface. Ainsi deux librairies furent utilisées : GTK et SDL. En effet, GTK reste un outil de référence pour le Développement d'interfaces en langage C sous Linux. C'est une librairie très vaste proposant un large panel d'outils et de fonctions propres à ceux-ci pour la réalisation d'une interface complète.

Ainsi, GTK fut retenu et en plus de celui-ci s'est ajouté Glade,l'outil interactif de conception d'interface GTK pour permettre une meilleure visualisation en direct de l'interface en développement. De ce fait, un fichier main.c et main.glade furent créer, les deux bien sûr reliées grâce à un Builder initialisé dans main.c relié au main.glade. Ainsi, la réalisation de l'interface sur les deux fichiers en parallèle a pu débuter. D'abord un temps de réflexion pour imaginer le modèle global a été pris pour ensuite choisir les différents widgets qui allaient être utiliser dans l'interface. Pour rendre celle-ci assez interactive, un stack a été implémenter sur trois pages accessibles par un stack switcher situé en haut de la fenêtre. Sur la première page est visible le logo du projet disposé sur un frame coloré. Cette page a pour unique fonctionnalité d'être la première de l'interface, c'est sur la deuxième page qu'est présent le gros du projet. En effet, en arrivant sur celle-ci, nous pouvons apercevoir un frame central sur lequel est affiché la lettre E qui va être l'image par défaut du test. Sur la partie inférieure du frame sont présents des

boutons : en effet, on distingue, de gauche à droite, un sélectionneur de temps, à gauche qui, comme son nom l'indique, va permettre à l'utilisateur de choisir la durée du test qu'il va pouvoir démarre en cliquant sur un bouton start situé à sa droite. On peut aussi observer encore à droite une ProgressBar (widget gtk) qui va se régler selon le temps en seconde sélectionné par l'utilisateur et dégraisser petit à petit. On observe autour de l'image, étant centré sur un GtkFixed, container de dimensions fixes, 4 boutons disposés symétriquement afin de permettre à l'utilisateur d'indiquer la direction qu'il devine sur la lettre courante. Ainsi, pour chaque réponse donnée par l'utilisateur, les modifications de l'image rentrent en jeu :

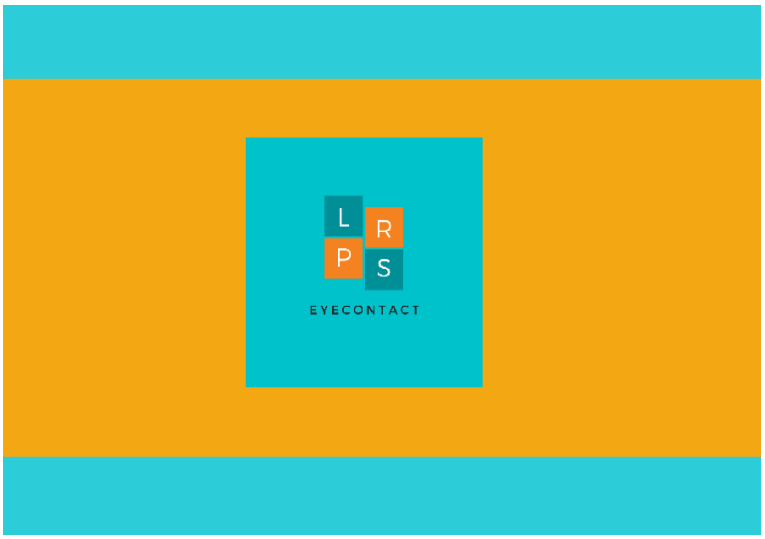


En effet, a été coder au préalable un fichier rotations.c dans lequel a été implémenter les fonctions permettant les rotations de la lettre. Sur ce fichier figurent deux fonctions de type void et prenant en paramètre toutes deux une image de type SDL*Surface et une matrice d'entier. Au préalable, notre image E de base a été chargé par SDL par la foncions SDL.load image et est donc convertie en image de type SDL surface. Suite à cela est créé une et allouée matrice d'entier de passage dont les dimensions sont celles de l'image E. L'algorithme de rotation parcours alors l'image E et effectue le principe de rotation et le remplissage de la matrice P. Etant donnée que l'image initiale du E et uniquement noire et blanc, la matrice est remplie de manière de la façon suivante : si le pixel récupérer par la rotation a une valeur R $>$ 255, on donne la valeur 1 a la valeur de la matrice, 0 sinon. Une fois la matrice remplie des valeurs 1 e 0

des pixels de l'image ainsi retournée, on change alors tous les pixels de l'image E en fonction de la matrice, l'image est ensuite sauvegardée grâce à SDL. Le fichier rotations.c possèdent deux fonctions, une effectuant la rotation vers la droite, l'autre vers la gauche. L'appel à la rotation s'effectue dans une fonction change() dans main.c elle-même appelée par les fonctions clicked des boutons entourant l'image et représentant les choix de directions de l'utilisateur. En effet, une fois que la réponse est donnée par l'utilisateur, il faut une nouvelle image d'orientation différente. L'image de base est ensuite remplacée dans le container du frame propre à l'affichage de la lettre courante.

Mais, les rotations seules ne suffisent pas pour présenter un test complet. Il fallait aussi assurer une modification des dimensions nouvelles images fournies par les algorithmes. Grace à l'outil pixbuf fourni par glade, il est possible de modifier les dimensions de l'image en créant plusieurs éléments de types GdkPixbuf. Ainsi, à chaque clique sur l'un de 4 boutons, un entier random est générer par la fonction clicked du bouton en question, compris entre 15 et 201 (la dimension fixée du container de l'image) et cet entier est utilisé en paramètre de la fonction change qui en plus de retourner l'image de manière aléatoire entre la fonction rotateleft() et rotateright(), redimensionne l'image selon l'entier random et place la nouvelle image au centre du container après avoir supprimer l'image precedente. Ainsi tout ce processus permet d'obtenir un test de snellen diversifié. Pour l'instant, il reste encore à integrer le reseau de neurones permettant de detecter l'orientation de l'image pour que soit alors mis en correlation

et en comparaison la reponse de l'utilsateur et celle du reseau de neurones. On imagine alors que pour chaque reponse de l'utilsateur, le bouton ayant été cliqué se remplisse de la couleur verte si la reponse est celle donnée par le reseau de neurones, et rouge sinon. A la fin du test, l'utilisateur aurait un résultat final mais aussi interviendra au cours du test la prediction fournit par le reseau de neurones, objectif principal du projet.



Chapitre 6

conclusion

Pour conclure ce cahier des charges, nous aimerions dire que trouver l'idée principale de notre projet, à savoir travailler sur un logiciel permettant d'aider l'Homme (à notre échelle bien sûr). N'a pas été facile. L'idée de l'optique nous est venue spontanément, aujourd'hui de plus en plus de monde porte ou est amené à porter des lunettes. En effet que ce soit pour se protéger de la lumière bleue de nos écrans ou avec l'âge qui avance, porter des lunettes s'impose à beaucoup. Notre projet peut paraitre ambitieux mais nous sommes prêts à donner de nous-mêmes afin de réaliser de quoi nous rendre fier. Nous vous remercions pour la lecture de ce cahier des charges.