Projet de fin d'études

Développement d'une application d'évaluation visuelle des cartes d'explication sur des images

Encadrants

BENOIS-PINEAU Jenny, GIOT Romain, MANSENCAL Boris

Collaborateurs

ZHUKOV Alexey, BEDOURET Baptiste, EL KERZAZI Mohammed-Amine, REBIERE-POUYADE Lilian

Février 2023

Table des matières

1 Introduction

Dans le domaine de l'intelligence artificielle, la classification repose dans le fait qu'un modèle, par exemple CNN, puisse décider de la classe d'appartenance d'une entrée donnée, parmi la taxonomie définie. Il est alors important de pouvoir faire comprendre, à un sujet humain, les raisons de cette décision par le biais d'outils visant à apporter des informations. Dans le domaine de la classification d'images, on parle alors de cartes d'explication. Celles-ci sont définies sur des images corrompues, c'est-à-dire ayant subies des dégradations, ayant pour but de mettre en évidence les régions importantes des images qui ont contribué à une prédiction.

Dans ces conditions, des mesures des observations d'images sont utilisées pour évaluer les techniques de traitement d'images. L'évaluation de la qualité de ces cartes d'explication repose habituellement sur des mesures obtenues de manière automatique. Dans ce cas, il est intéressant de pouvoir présenter ces cartes d'explications à des sujets humains. Ainsi, la perception humaine permettant d'avoir un jugement différent, un sujet humain doit pouvoir émettre une évaluation pour ainsi donner son avis sur la qualité des cartes issues des décisions d'un classifieur. De ce fait, la norme IUT-R BT.500-14 et la méthode du stimulus unique permettent de définir le protocole d'évaluation de l'expérience.

La problématique est donc la suivante : développer un outil pour une expérience participative d'évaluation des cartes d'explication par des sujets humains.

Dans ce rapport, nous allons tout d'abord aborder, dans l'état de l'art 2, les protocoles d'évaluation, échelles de notation et la composition des cartes d'explication des classifieurs. Ensuite, dans la section 3, nous présenterons les spécifications de l'interface, de l'expérience et des données, avant de décrire les fonctionnalités de l'application dans la section 4. De plus, la section 5 abordera l'architecture de notre projet ainsi que des détails sur l'implémentation des fonctionnalités de l'application. Enfin, la section 6 présentera les résultats des tests effectués avant de finir avec la conclusion dans la section 7.

2 État de l'art

Dans cette section, il est question d'aborder l'existant autour des mesures d'observations des images, des évaluations des classifieurs CNN, ou encore des méthodes et outils concernant l'expérience visuelle.

2.1 Mesures d'observation des images

Les mesures d'observation des images sont effectuées dans plusieurs contextes d'évaluation des méthodes de traitement d'image, comme la détection des régions importantes des images [1], du codage des images et des vidéos [2], dans la prédiction automatique de l'attention visuelle ou d'intérêt de l'utilisateur dans les images [3].

Ces mesures sont basées sur l'enregistrement des fixations du regard des utilisateurs qui observent le contenu visuel : image ou vidéo [1].

Dans le domaine de visualisation d'information, les études de qualité des interfaces de visualisation sont également faites en impliquant des cohortes des sujets de tests [4].

2.2 Évaluation des cartes d'explications des classifieurs CNN

Les outils d'explication de la classification de l'intelligence artificielle sont nécessaires pour faire comprendre aux décideurs humains pourquoi le classificateur CNN a pris la décision précise quant à l'appartenance de l'échantillon de données à une des classes de la taxonomie définie. À ce jour, il existe une quantité importante d'outils d'explication des décisions des classifeurs CNN pour les images [5]. Ainsi se pose le problème d'évaluation de la qualité de ces outils.
Un explicateur est un outil qui vise à fournir des informations sur le processus de prise de décision d'un modèle CNN en générant une explication pour une entrée donnée. Pour la classification d'images, notamment, le résultat d'un explicateur représente une carte d'explication. C'est une carte de chaleur mettant en évidence les régions importantes de l'image d'entrée qui ont contribué à la prédiction du modèle [6].

L'état de l'art en matière d'évaluation des explicateurs implique plusieurs mesures visant à évaluer la qualité des cartes générées. Ces mesures sont obtenues généralement de façon automatique, par exemple la mesure de stabilité des cartes d'explication [7] qui permet d'évaluer la stabilité de la carte d'explication dans le cas de dégradations sur les images d'entrée classifiée par le modèle pré-entraîné. Les métriques par insertion et suppression des pixels en fonction de la carte d'explication permettent de mesurer les changements des scores de classification et ainsi, de façon indirecte, d'évaluer la carte [8], [9]. Les mesures de compacité des cartes ont été également proposées par les auteurs de [8] supposant que l'interprétation des cartes par les humains est simplifiée si la carte d'explication est compacte. Toutes ces mesures sont calculées de façon automatique pour qualifier les outils d'explication. La première tentative de comparer les cartes d'explication avec la perception humaine du contenu visuel dans une seule et même tâche de classification (humain/CNN) a été entreprise dans [10]. Ici, les auteurs comparent les cartes obtenues, par plusieurs outils d'explication, avec les cartes d'attention visuelle résultantes d'une expérience psychovisuelle avec l'enregistrement des points de fixation du regard de l'utilisateur, en utilisant les métriques de comparaison de cartes de saillance visuelle. Il est donc intéressant de confronter la carte d'explication obtenue par un explicateur avec le jugement humain. Les utilisateurs doivent évaluer la carte en donnant leur score d'opinion pour la qualité de cette dernière et donc répondre à la question si la carte explique bien la décision (correcte ou erronée) du classificateur.

Ainsi, *l'objectif du projet consiste à développer un outil pour l'expérience participative visuelle des utilisateurs afin d'évaluer les cartes d'explication des outils d'explication.*

Cette expérience consiste à présenter deux images à l'utilisateur :
— une image corrompue ;
— la carte d'explication établie à partir de cette image corrompue

Comme information sémantique, l'utilisateur a également accès à la classe de vérité terrain et la classification établie par le réseau profond. De plus, une expérience se décompose en plusieurs séances qui doivent être unique à chaque utilisateur. Chaque séance de test est alors établie de manière aléatoire et ne doit pas durer plus d'une heure. Ainsi, cela permet à l'observateur de rester concentré durant la totalité de sa participation.

2.3 Expérience visuelle participative

Pour mener à bien une expérience visuelle participative d'évaluation, nous devons définir :
— le protocole d'évaluation ;
— l'échelle de notation

2.3.1 Protocoles d'évaluation

Les expériences visuelles pour l'évaluation de la qualité des images, par exemple [2], ou comparaison des modèles d'attention profonds avec le jugement humain [11], doivent respecter un certain protocole. Les objectifs de ce dernier consistent, mis-à-part la réponse au problème d'évaluation, à
— diminuer la *fatigue visuelle* ;
— réduire les effets de mémoire visuelle
des observateurs participants à l'expérience d'évaluation.

Ce type de protocole a été élaboré par la communauté d'évaluation de la qualité d'images et normalisée dans la norme IUT-R BT.500-14 [12]. Cette norme définit l'évaluation de la qualité d'images en mode "single stimulus", c'est-à-dire l'observation d'une seule image et double stimulus. Ce dernier consiste à observer l'image d'origine et l'image dégradée simultanément. Néanmoins, les préconisations pour répondre aux contraintes de la *fatigue visuelle* et de la latence visuelle (mémoire visuelle) sont les mêmes. Il s'agit de paramétrer la durée de l'expérience et les modalités de visionnage des images avec "la réinitialisation" de l'attention humaine par l'affichage des images neutres (grises). Dans la conception de notre protocole d'évaluation des cartes d'explication, nous devons donc suivre ces préconisations d'une expérience visuelle, tout en mettant en place un moyen d'évaluation.

2.3.2 Échelles de notation

Dans le domaine d'évaluation de la qualité d'images, quand une seule et même image est évaluée par plusieurs observateurs, les schémas d'évaluation "continue" et discret ont été proposés (cf. la norme IUT-R BT.500-14 [12]). Dans le premier cas, les observateurs utilisent un outil analogique, un potentiomètre, pour noter la qualité d'image de façon continue. Dans le cas d'évaluation

discrète, ils affectent les scores. Par ailleurs, en évaluation des interfaces visuelles, l'évaluation sur une échelle discrète est souvent adoptée. Il s'agit de l'échelle de Likert [13] qui est utilisé également pour des sondages, afin de faire une analyse psychologique ou de mesurer une attitude chez les individus. Cette échelle est composée de cinq niveaux :

— Excellent
— Bon
— Assez bon
— Médiocre
— Mauvais

L'utilisation de cette échelle dans l'expérience est simple. Ainsi, dans notre travail, nous mettons en place les outils graphiques pour assurer la notation des cartes visuelles selon l'échelle de Likert.

Voici les spécifications en ce qui concerne l'évaluation des cartes d'explications :

La qualité de l'explication est considérée comme étant bonne si :

— les zones de pixels qui correspondent à la classe en question sont affichées en zone de chaleur sur la carte d'explication.
— la classe attribuée par le classifieur est mauvaise, mais la carte de chaleur est la plus chaude sur des pixels qui ne correspondent pas à la vraie classe de l'image.

La qualité d'explication est considérée comme étant mauvaise si :

— la classification est correcte, mais la zone de pixel correspondant à la zone de chaleur n'est pas sur le résultat de la classification.
— la classe attribuée par le classifieur est mauvaise, mais la zone de chaleur correspond à la classe qui a été attribué par l'utilisateur.

Ainsi, avec l'outil graphique de notre application, l'observateur doit affecter un score de Likert à la carte d'explication observée selon ces définitions.

2.3.3 Quelles cartes d'explication à évaluer ?

Pour évaluer une carte d'explication des décisions des réseaux profonds sur des images, les auteurs de [14] proposent d'engendrer des dégradations contrôlées sur l'image d'entrée qui doit être soumis au classificateur CNN. Les cartes sont ensuite évaluées en fonction de dégradations pour les images bien classées, et les images qui changent leur affectation suite à la dégradation. Dans la suite de cette section, nous expliquons la génération des dégradations sur les images selon [14].

Ces dégradations comprennent : le bruit gaussien additif, le flou gaussien, la distorsion de luminosité uniforme et la distorsion de perspective.

Bruit Gaussien Additif :

Le bruit gaussien est considéré comme additif et indépendant du signal de l'image d'origine, où chaque valeur de pixel est incrémentée d'un décalage généré aléatoirement α :

$$I'(u,v) = I(u,v) + \alpha(u,v) \tag{1}$$

Ici, $\alpha(u,v) \sim \frac{1}{\sigma_{agn}\sqrt{2\pi}} \times \exp\left(-\frac{t^2}{2\times\sigma_{agn}^2}\right)$ avec t la variable indépendante, σ_{agn} est le paramètre d'échelle de la distribution gaussienne.

La valeur de σ_{agn} est dérivée de la valeur maximale de décalage absolu $|k|$, qui est la magnitude du bruit gaussien additif, telle que la probabilité que la magnitude du bruit soit supérieure à $|k|$ est de 0.05, conformément à la règle des deux sigmas de la distribution gaussienne. Pour choisir k, les auteurs de [14] fixent l'amplitude maximale de la différence entre la valeur bruitée d'un pixel dans un canal couleur γ. La valeur k est ensuite choisie dans l'intervalle :

$$-\gamma/\sqrt{H\times W} \leq k \leq \gamma/\sqrt{H\times W} \tag{2}$$

où H et W sont respectivement la hauteur et la largeur de l'image. Le paramètre d'échelle σ_{agn} est déduit comme étant :

$$\sigma_{agn} = k/2 \tag{3}$$

Un exemple d'images avec du bruit généré pour différents paramètres k est donné sur la Figure 1 ci-dessous. Plus l'amplitude k du bruit est élevée, plus l'image est corrompue.

(a) (b) (c)

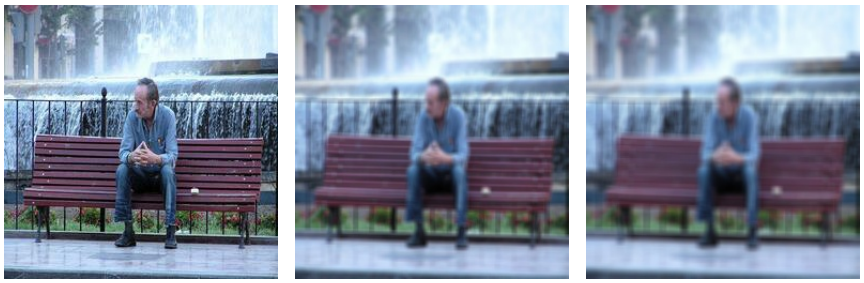
FIGURE 1 – Image originale corrompue par un bruit gaussien avec différents paramètres de décalage maximal : (a) original image, (b) k=50, (c) k=125

Flou Gaussien :
La distorsion de flou gaussien consiste en une convolution de l'image d'entrée $I(u,v)$ avec un noyau de filtre gaussien $g(\mu,\nu)$:

$$I'(u,v) = I(u,v) * g(\mu,\nu) \tag{4}$$

Ici $*$ désigne l'opération de convolution. Le noyau du filtre gaussien est défini comme $g(\mu,\nu) = \frac{1}{A} \times \exp\left(-\frac{\mu^2+\nu^2}{2\times\sigma_{gb}^2}\right)$, où A est un facteur de normalisation, σ_{gb} est le paramètre d'échelle du filtre gaussien.

Pour produire la même quantité d'images corrompues que la distorsion due au bruit gaussien, la taille s du masque du filtre est variée ainsi que le paramètre d'échelle σ_{gb} du filtre gaussien. Le même filtre g est appliqué aux trois composantes de couleur des images RVB.
Des exemples d'images floues sont donnés sur la Figure 2 ci-dessous.



(a) (b) (c)

FIGURE 2 – Image originale (a) corrompue par un flou gaussien avec un $\sigma_{gb} = 6$ et différents paramètres de noyau : (b) 5x5, (c) 7x7

Distorsion de Luminosité Uniforme :
La distorsion uniforme de la luminosité est effectuée en ajoutant un décalage aléatoire β aux trois composantes de couleur de l'image. La valeur de β est choisie au hasard en utilisant la méthode décrite dans Bruit Gaussien Additif, où le paramètre d'amplitude maximale du décalage k est utilisé comme paramètre. La nouvelle valeur de couleur dans chaque canal est calculée comme suit :

$$I'(u,v) = min\left(255, max\left(0, I\left(u,v\right) + \beta\right)\right) \tag{5}$$

Cela permet de s'assurer que les valeurs de couleur obtenues se situent dans la plage valide de [0,255].

Des exemples de distorsion de la luminosité sont donnés sur la Figure 3.

(a) (b) (c)

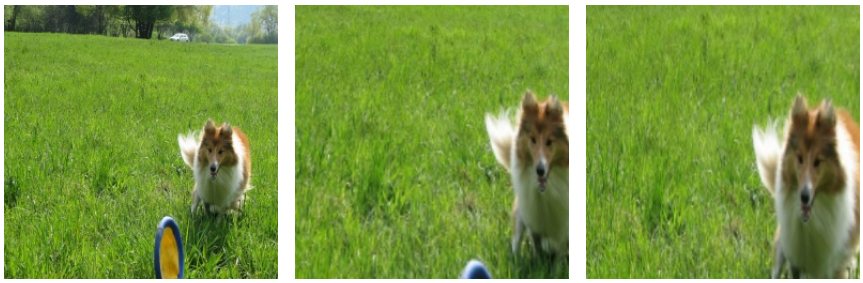
FIGURE 3 – Image originale corrompue par la luminosité avec différents paramètres de décalage maximal : (a) image originale, (b) β=50, (c) β=125

Distorsion de Perspective :
Distorsion de perspective fréquente dans des images naturelles est une distorsion géométrique du plan de l'image : $F : I'(u', v') = I(F(u, v))$. La transformation s'exprime comme suit en coordonnées homogènes :

$$u' = H \times u \tag{6}$$

Pour plus de détails, voir [15]. La matrice d'homographie H a huit paramètres définis par des paires de points correspondants dans les images source et cible $(u_1, v_1), (u'_1, v'_1), ..., (u_4, v_4), (u'_4, v'_4)$. Les valeurs des pixels manquants dans l'image cible sont interpolées de manière bilinéaire pour générer une distorsion de perspective sans trous dans l'image cible.

Des exemples d'images déformées sont illustrés sur la Figure 4 ci-dessous.

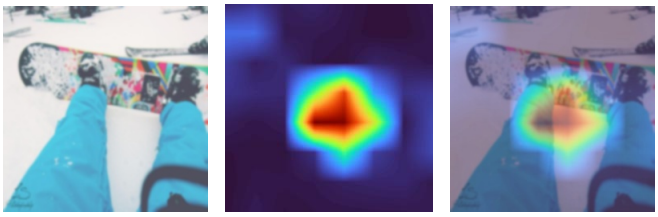


(a) (b) (c)

FIGURE 4 – Image originale (a) corrompue par un changement de perspective : (a) image originale, (b) haut, (c) droit

Tandis que dans des protocoles automatiques d'évaluation, les métriques sur les cartes d'explication sont calculées dans l'espace de représentation des cartes $R^{[0;1]}$, pour une expérience d'évaluation visuelle, les cartes de chaleur doivent

être superposées sur les images dont la classification elles expliquent. Sur la Figure 5 ci-dessous.



(a) (b) (c)

FIGURE 5 – Une image corrompue par un flou gaussien k = 1.25, sa carte d'explication obtenue avec l'algorithme FEM [16] et leur version combinée : (a) image corrompue, (b) carte d'explication, (c) sup image

2.4 Conclusion

Ainsi, différentes méthodes de mesures d'observation sont effectuées dans plusieurs contextes d'évaluation comme des méthodes de traitement des images ou la détection des régions importantes dans les images. Ce dernier point est celui utilisé au sein de l'expérience participative à mettre en place. Pour évaluer et mesurer ces observations sur des images, l'utilisation d'outils d'explication est nécessaire pour prendre des décisions à l'aide des classifieurs CNN. L'utilisation d'une carte d'explication est un explicateur qui vise à fournir des informations sur le processus de décision d'un modèle CNN. Plusieurs mesures directes ou indirectes sont utilisées de manière automatique pour évaluer la qualité des cartes d'explications par exemple, la mesure de stabilité de la carte d'explication. Mais ces mesures peuvent être évaluées par jugement humain. En effet, l'expérience expliquée dans ce rapport consiste à présenter une image ayant subit une dégradation (ex : flou gaussien) et la carte d'explication qui résulte de cette image. Le sujet humain intervient pour évaluer la décision prise par le réseau de neurones CNN sur la carte d'explication à l'aide de l'échelle de Likert.

3 Problème à résoudre : Conception de l'interface d'évaluation des cartes d'explication

Dans la conception de l'interface d'évaluation, il est nécessaire de prendre en compte à la fois les spécifications graphiques telles que la disposition des éléments, et dynamiques, c'est-à-dire, le scénario d'évaluation.

3.1 Spécification graphique de l'interface

Dans la conception de l'interface, l'objectif principal consiste à éviter à l'utilisateur toute distraction visuelle. Ainsi, l'interface développée doit être mini-

maliste sur le plan graphique. Elle doit donc contenir :

1. Une image qui a été classifiée ;
2. Une carte d'explication superposée en carte de chaleur sur l'image ;
3. La vérité terrain de la classe de l'image ;
4. La classe de l'image prédite par le classifieur ;
5. La colonne de boutons radio de l'échelle de Likert

La disposition des éléments sur l'interface doit être dans l'ordre de lecture "naturel", à savoir de gauche à droite. Cela a pour but d'éviter la *fatigue visuelle* de l'utilisateur. Par ailleurs, la couleur d'arrière-plan de l'écran doit être *la plus neutre possible* afin d'éviter toute distraction de l'utilisateur par les éléments visuels non nécessaires, afin de réduire également la *fatigue visuelle*. Les polices et la taille des caractères utilisés doivent assurer une très bonne lisibilité de l'information, et ceci en mode "reactive", c'est-à-dire adapté à la résolution de l'écran.

La maquette de l'interface répondant à ces spécifications est présentée sur la Figure 6 ci-dessous :

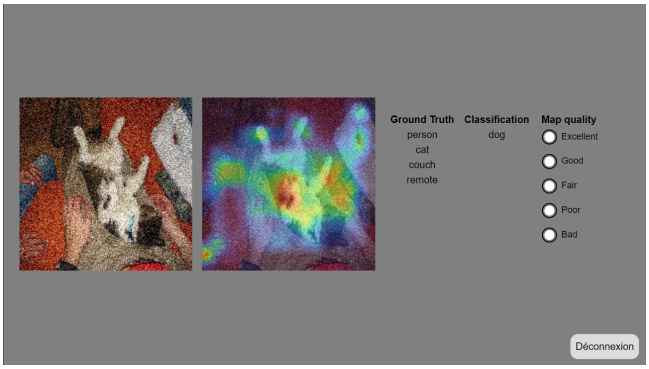


FIGURE 6 – Un exemple des pages de la maquette de l'interface graphique

3.2 Déroulement de l'expérience

Concernant l'organisation de l'expérience, il est important de prendre en compte l'aspect visuel par rapport à l'utilisateur. La norme IUT-R BT.500-14 [12] (cf. 2.3.1) présente également un scénario de présentation bien précis. En effet, le temps d'évaluation de chaque carte ne doit pas dépasser quelques secondes. La norme IUT-R BT.500-14 [12] préconise l'affichage des images à évaluer pendant le temps t_I minimal de 10 secondes. L'utilisateur doit réagir en évaluant la carte, donc ce temps d'affichage t_I peut être raccourci si la réaction de l'utilisateur est survenue avant la durée minimale de 10 secondes. Par ailleurs, il est nécessaire de limiter t_I car, pendant l'expérience visuelle, l'utilisateur peut

être distrait et ainsi ne pas pouvoir évaluer les images ou les cartes affichées. La limite haute de t_I qui peut être proposée est de 15 sec.
Un temps dit de réinitialisation de l'attention visuelle t_r, selon la norme d'une durée de 5 secondes, doit être accordé entre deux évaluations des images ou des cartes.

Pour la réinitialisation de l'attention visuelle, la norme IUT-R BT.500-14 préconise l'utilisation d'une image uniforme d'un gris moyen affichée à l'utilisateur pendant le temps t_r. Ainsi, la scène visuelle précédente "oubliée", l'utilisateur pourra se concentrer sur la nouvelle carte pour l'image suivante sans être influencé et sans garder en mémoire les images précédentes.

3.3 Scénario dynamique d'évaluation

Pour chaque expérience, il est défini le scénario dynamique suivant :

1. Test d'Ishihara
2. L'utilisateur crée son compte ou se connecte sur son compte existant
3. L'utilisateur prend connaissance de la notice, l'expérience commence seulement après sa lecture
4. Phase d'entraînement : 3 paires d'images pour commencer afin de comprendre le déroulement de l'évaluation
5. Phase d'évaluation (durée 1 heure)
6. Les images suivantes sont affichées lorsque l'utilisateur a cliqué sur un des boutons

Avant de prendre part à l'expérience, les utilisateurs doivent se soumettre à un test d'Ishihara. Il s'agit d'un test de perception des couleurs dont l'objectif principal est de détecter toute éventuelle anomalie concernant la vision et la perception des couleurs. En effet, étant donné que la carte d'explication est superposée en carte de chaleur sur l'image, ce test est fondamental pour être sûr de pouvoir correctement participer à l'expérience.

En ce qui concerne l'expérience, la phase d'entraînement permet de montrer concrètement à l'utilisateur comment se déroule une session. Durant cette préparation, les évaluations faites par ce dernier ne doivent pas être prises en compte dans les résultats enregistrés en fin d'expérience. À partir du moment où l'utilisateur démarre la phase d'évaluation, chacun de ses choix, et le temps t pour prendre ces choix, sont enregistrés dans le fichier final de l'utilisateur.

3.4 Protection des données personnelles

Durant l'expérience, nous avons besoin de recueillir et conserver les données suivantes sur l'utilisateur :

— son adresse mail ;
— sa tranche d'âge ;
— son mot de passe ;
— date de création du compte.

La sauvegarde de ces données sera effectuée sur le serveur protégé du LABRI **AIVCALC5**. Le contenu du fichier des données enregistrées est illustré dans la Table 1

identifiant	adresse mail	mot de passe	tranche d'âge	date de création du compte
1	adresse1@email.com	mdp1	18 - 21	2023-03-02 11 :30 :17
2	adresse2@email.com	mdp2	22 - 25	2023-03-10 17 :02 :42
3	adresse3@email.com	mdp3	26 - 30	2023-03-21 14 :17 :54
4	adresse4@email.com	mdp4	31 - 40	2023-02-28 09 :52 :04

TABLE 1 – Tableau représentatif de la base de données pendant l'expérience

À la fin de l'expérience, seuls l'ID de l'utilisateur, sa tranche d'âge et les résultats d'évaluation sont conservés pendant dix ans. Toutes autres informations seront automatiquement effacées dans un délai de six mois après le début de l'expérience, c'est pourquoi il est important de conserver la date de première création du compte. De plus, au lieu de supprimer les colonnes 'adresse mail' et 'mot de passe' dans la base de données, chaque ligne de ces deux colonnes est mise à jour avec l'ID de l'utilisateur. Le contenu du fichier d'un utilisateur est illustré dans la Table 2.

identifiant	adresse mail	mot de passe	tranche d'âge
1	1	1	18 - 21
2	2	2	22 - 25
3	3	3	26 - 30
4	4	4	31 - 40

TABLE 2 – Tableau représentatif de la base de données après l'expérience

Ces délais et conditions ont été abordés avec le service DPO de l'Université de Bordeaux, le juriste M Bénard. L'expérience sera inscrite dans le registre des données personnelles de l'Université de Bordeaux et une attestation provisoire doit être délivrée avant cette inscription.

3.5 Conclusion

Ainsi, le déroulement de l'expérience doit suivre un protocole bien précis, tout comme la conception de l'interface. Tout d'abord, celle-ci doit être la plus minimaliste et ergonomique possible, l'utilisateur devant être focalisé uniquement sur les cartes d'explications qu'il doit évaluer. Dans la continuité de cet

objectif, la disposition des éléments doit se faire dans l'ordre de lecture "naturel", tandis que la couleur de l'arrière-plan doit être la plus neutre possible. De plus, des temps d'affichage des images t_I et de réinitialisation visuelle t_r bien précis doivent être respectés pour prendre en compte l'aspect visuel. Enfin, la sensibilité des données personnelles des utilisateurs représente une part importante du projet. Seulement son adresse mail, sa tranche d'âge ainsi que son mot de passe sont recueillis, et peuvent être conservés seulement durant une période de 6 mois concernant l'adresse mail et le mot de passe. En revanche, les autres informations ainsi que les résultats des évaluations des utilisateurs sont des données qui sont conservées durant la durée de l'étude, soit 10 ans.

4 User Story

Chacune des sections ci-dessous abordent les différentes composantes de notre outil et montrent son utilisation par le biais de figures.

4.1 Création du compte

Pour pouvoir participer à l'expérience, il faut tout d'abord que l'utilisateur crée un compte. Pour cela, il peut cliquer sur le lien "créer un compte", et accède ainsi à une nouvelle page composée d'un formulaire d'inscription dont plusieurs champs sont à renseigner, cf. Figure 7 :

FIGURE 7 – Création de compte

Comme nous pouvons l'observer ci-dessus, l'utilisateur doit indiquer son adresse mail, sélectionner sa tranche d'âge, puis créer et confirmer son mot de passe. Chacune de ces informations est alors enregistrée dans la base de données.

Plusieurs cas d'erreurs sont traités lors du renseignement de ce formulaire. En effet, une adresse mail déjà utilisée, une confirmation de mot de passe différente de celui créé, ou encore le non-remplissage d'un champ empêche la création d'un

nouveau compte. L'utilisateur doit également accepter les conditions générales d'utilisation, qui lui sont présentées sur la Figure 8 ci-dessous, pour pouvoir créer un compte. Dans le cas contraire, l'accès à l'outil d'expérimentation lui est refusé.

université de BORDEAUX

EVALUATION
Version n°2.0 du 07/02/2023

NOTICE D'INFORMATION A DESTINATION DES PARTICIPANTS

[Evaluation visuelle des cartes d'explication des décisions des Réseaux Profonds (IA) sur des images]

EVALUATION

Coordonnées du Responsable scientifique :
Jenny Benois-Pineau
Laboratoire LABRI
N° téléphone 0540008424
Adresse mail jenny.benois-pineau@u-bordeaux.fr

Coordonnées du Délégué à la Protection des Données :
Université de Bordeaux
DPO
35 place Pey Berland
33000 Bordeaux
dpo@u-bordeaux.fr

Madame, Monsieur,

L'Université de Bordeaux vous propose de participer à un projet de recherche porté par un laboratoire de l'Université de Bordeaux. Avant de prendre une décision, il est important que vous lisiez attentivement ces pages qui vous apporteront les informations nécessaires concernant les différents aspects de cette recherche. N'hésitez pas à poser toutes les questions que vous jugerez utiles au responsable scientifique de l'étude que vous pouvez joindre aux coordonnées ci-dessus.

Pourquoi cette recherche ?
[Explication des décisions des outils d'Intelligence Artificielle (IA) tels les réseaux profonds est un des aspects centraux de la recherche en IA explicable et en IA de confiance. Une telle explication est particulièrement facilitée sur les données visuelles de type image. En effet, l'humain prend sa décision par rapport au contenu saillant de l'image qu'il perçoit. Un réseau profond prend la sienne en fonction des poids entrainés sur l'ensemble d'apprentissage. Un outil d'explication automatique de la décision d'un réseau entrainé par rapport à l'image d'entrée génère une carte d'explication qui montre les éléments dans l'image qui ont servi le plus à la décision. L'évaluation des performances des outils d'explication reste un problème ouvert. Dans le cadre de notre étude nous souhaitons utiliser le jugement humain pour évaluer la qualité des cartes d'explication et de le confronter avec les résultats d'évaluation automatique par des méthodes que nous avons proposées auparavant.]

Quel est l'objectif de cette recherche ?
L'objectif de l'expérience à laquelle vous allez participer consiste à comparer les résultats d'évaluation obtenus par des méthodes automatiques avec l'évaluation par des personnes.

Comment va se dérouler cette recherche

Vous allez observer, sur un écran d'ordinateur :
- Une image ;
- Son résultat de classification en ensemble des classes connues, comme par exemple : « paysage », « chien » ou « voiture » ;
- Sa vraie classe – la vérité terrain
- La carte de chaleur superposée sur l'image qui met en valeur par la couleur plus chaude (« jaune », « orange », « rouge ») les pixels dans l'image qui ont contribué le plus à la décision.
- L'échéancier selon l'échelle de Likert de la qualité de la carte d'explication.

Page 1 sur 3

FIGURE 8 – Conditions générales d'utilisation

Une fois que le compte de l'utilisateur est créé, un identifiant unique lui est attribué lors de l'importation de ses informations dans la base de données. Cette création de compte permet également de générer les différents fichiers *CSV* propres à l'utilisateur grâce à son identifiant :

— un fichier dit **aléatoire** contenant les N images à évaluer durant l'expérience dans un ordre aléatoire. Ce fichier permet de créer un scénario unique pour chaque utilisateur.
— un fichier dit de **session** contenant toutes les nb_images à afficher durant une session.
— un fichier de **sauvegarde** dans lequel sont enregistrés tous les choix et temps t pris par l'utilisateur pour chaque choix.

Tous ces fichiers comportent en leur nom l'identifiant id de l'utilisateur actuel pour éviter toute concurrence d'accès et permettre qu'ils soient uniques pour chaque utilisateur.

Pour le moment, seul le fichier aléatoire est modifié. Pour ce faire, le scénario étant unique pour chaque utilisateur, ce fichier contient les 1800 images à évaluer, récupérées dans un ordre aléatoire. Il contient également les 24 images utilisées pour la phase d'entraînement, communes cette fois-ci à tous les utilisateurs.

4.2 Connexion à l'expérience

Dans le cas où toutes les informations nécessaires sont correctement entrées et récupérées, l'utilisateur peut alors se connecter à l'expérience. Il doit simplement renseigner son adresse mail et mot de passe liés au compte précédemment créé. Un exemple est présenté sur la Figure 9 :

FIGURE 9 – Connexion au compte

L'utilisateur a alors accès à un descriptif sur le contexte de l'étude et prend connaissance des consignes relatives au déroulement de l'expérience, cf. Figure 10 :

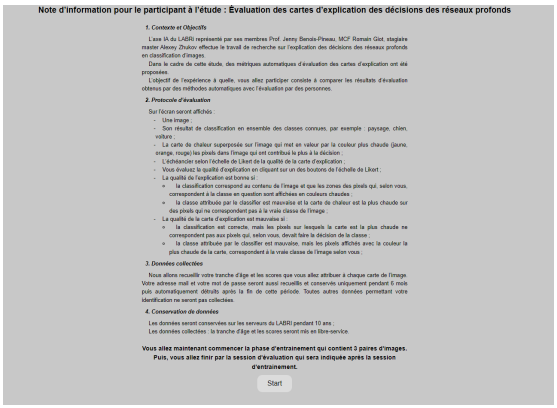


FIGURE 10 – Descriptions et consignes de l'expérience

Chaque fois que l'utilisateur arrive sur cette page, le fichier *CSV* de session est rempli seulement s'il est vide, c'est-à-dire si l'utilisateur a bien fini la session précédente, donc sans se déconnecter en cours de route. Le fichier dépend directement du fichier *CSV* aléatoire créé lors de la création du compte, à partir duquel on récupère les 243 premières images. Les 3 premières images sont destinées à la phase d'entraînement, tandis que les 240 images suivantes sont soumises à une évaluation de la part de l'utilisateur.

L'utilisateur peut démarrer la session quand il le souhaite, en appuyant sur le bouton *start*, et charger l'interface principale utilisée tout le long de l'expérience, cf. Figure 11.

FIGURE 11 – Interface graphique de l'entraînement et de l'évaluation

L'utilisateur, en ayant lancé la session, peut alors visualiser la nouvelle interface composée de :

— une image d'origine ;
— la carte d'explication de cette image ;
— la vérité terrain ;
— la classification ;
— une colonne de bouton radio lui permettant d'évaluer les images

4.3 Déroulement de la session

Pour commencer la session, l'utilisateur prend part à une phase d'entraînement visant à lui permettre de découvrir et de comprendre l'interface et les interactions possibles. Comme évoqué précédemment, cette phase se compose de 3 images, en revanche, contrairement à la phase d'évaluation, ni ses choix, ni les temps t associés à ses choix, ne sont enregistrés. D'autre part, comme durant la phase d'évaluation, l'utilisateur a un maximum de 15 secondes pour effectuer un choix, tandis que la réinitialisation visuelle entre deux images, c'est-à-dire seulement l'affichage de l'arrière-plan, se fait lors dès 5 secondes suivantes, cf. Figure 12 :

FIGURE 12 – Réinitialisation visuelle entre deux évaluations

Passé cette phase de 3 images, l'utilisateur va pouvoir démarrer la phase d'évaluation dès qu'il le souhaite en appuyant sur le bouton *start*, cf. Figure 13.

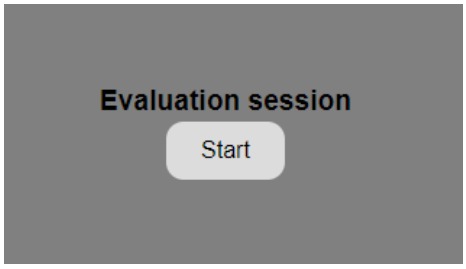


FIGURE 13 – Lancement de la phase d'évaluation

Durant la phase d'évaluation, tous les choix de l'utilisateur et les temps t pris pour effectuer ses choix sont dorénavant enregistrés.

4.4 Déconnexion en cours de session

S'il n'a pas le temps de finir une session, l'utilisateur doit pouvoir se déconnecter de la session en cours. Pour cela, il a accès au bouton *déconnexion* durant les 20 secondes de chaque évaluation, en bas à droite de la Figure 14 ci-dessous. Ce dernier est dissimulé lors de la réinitialisation visuelle.

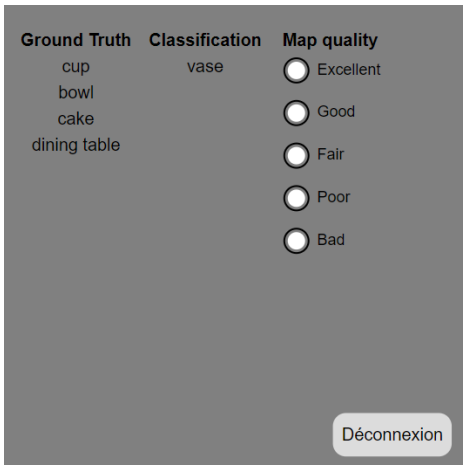


FIGURE 14 – Bouton de déconnexion en bas à droite de l'interface d'évaluation

L'utilisateur est dans ce cas directement redirigé vers la page de connexion (cf. Figure 9).

Durant cette déconnexion, il est nécessaire de sauvegarder les choix jusqu'à présent effectués par l'utilisateur dans le fichier *CSV* de sauvegarde, et également d'enlever les images déjà présentées dans le fichier *CSV* de session, sauf celles pour l'entraînement. Ce fichier ne contient plus alors que les 3 images d'entraînement pour cette session ainsi que les images qu'ils restent à évaluer.

Lors de sa reconnexion, l'utilisateur visualise toujours la page de description de l'expérience et des consignes (cf. Figure 10). Le fichier de session n'est pas modifié puisque l'utilisateur s'est déconnecté en cours de session. L'utilisateur prend tout de même part à la phase d'entraînement, visualise la page de séparation entre les phases (cf. Figure 13), puis passe à la phase d'évaluation. Le fichier de session étant modifié en fonction, les images déjà évaluées ne sont évidemment plus affichées, et l'utilisateur peut alors continuer la session actuelle par la première image non évaluée.

4.5 Fin de session

Une fois la dernière carte d'explication évaluée, l'utilisateur a terminé la session actuelle. Il a alors accès à un nouvel écran pour le lui signaler, cf. Figure 15 ci-dessous, et a la possibilité de se déconnecter de l'application, ou bien de relancer une nouvelle session.

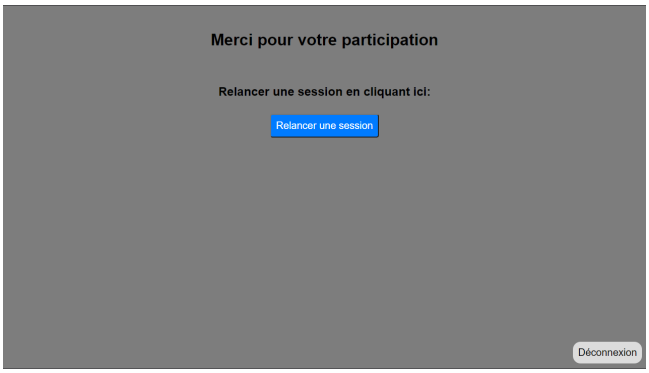


FIGURE 15 – Fin de session

Le fait de relancer la session le ramène à l'écran des consignes (cf. Figure 10), tandis que, comme dans le cas de la déconnexion en cours de session, le fait de se déconnecter le ramène à l'écran de connexion (cf. Figure 9).

Les différentes évaluations récupérées durant la session sont enregistrées dans le fichier *CSV* de sauvegarde, nommé avec l'identifiant *id* de l'utilisateur, qui doit contenir les résultats de chaque session. Chacune des évaluations, ainsi que leurs temps t respectifs, sont ajoutés à la fin du fichier pour, une fois l'expérience finalisée, contenir les choix de chacune des 1800 images à évaluer.

4.6 Fin d'expérience

L'utilisateur a effectué toutes les sessions, c'est-à-dire l'évaluation de toutes les images, et a donc terminé l'expérience. Dans ce cas, la page de fin de session est modifiée en fonction pour empêcher l'utilisateur de relancer une session, cf. Figure 16 :

FIGURE 16 – Fin d'expérience

Il en est de même si l'utilisateur se connecte à l'application. Dans le cas où il a fini l'expérience, au lieu d'arriver sur la page des consignes (cf. Figure 10), il est directement redirigé sur cette page qui lui informe de la fin de l'expérience (cf. Figure 16).

Parmi les fichiers *CSV*, seul le fichier de sauvegarde doit être conservé, les fichiers aléatoires et de session peuvent alors être supprimés. Concernant la base de données, certaines informations liées au compte de l'utilisateur peuvent être conservées seulement pendant 6 mois. Ceux-ci concernent chacune des informations à l'exception de l'identifiant qui a été créé pour l'utilisateur ainsi que la tranche d'âge de l'utilisateur.

4.7 Page administrateur

L'administrateur peut avoir accès au contenu des fichiers *CSV* de résultats d'évaluations liées à chaque utilisateur. Une page de l'interface lui est alors entièrement dédiée, inaccessible par les utilisateurs. L'administrateur peut accéder à cette page en se connectant avec une certaine adresse mail et un certain mot de passe, comme on peut le voir sur la Figure 17 :

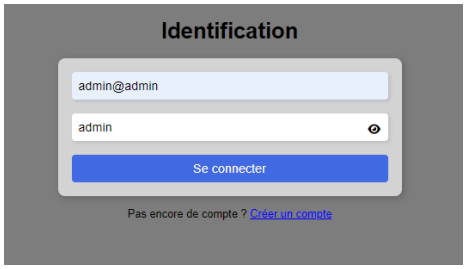


FIGURE 17 – Connexion de l'administrateur

L'administrateur a alors accès à une nouvelle interface avec laquelle il peut interagir, cf. Figure 18 :

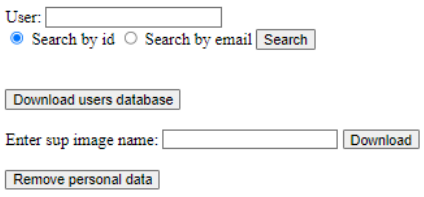


FIGURE 18 – Page administrateur

L'administrateur a alors deux possibilités :

— Indiquer l'identifiant ou adresse mail d'un utilisateur : il peut alors visualiser les informations de l'utilisateur, cf. Figure 19 ci-dessous, à partir desquels il peut soit les supprimer, soit télécharger le fichier de sauvegarde de cet utilisateur, contenant les images qu'il a dû évaluer, et les scores donnés à chacune d'elles dans un temps t.
— Indiquer le nom d'une image de carte d'explication *sup* : dans ce cas, l'administrateur a accès aux données concernant la seule image indiquée. Dans le fichier généré par cette opération, il visualise les noms des fichiers qui contiennent une évaluation des utilisateurs pour cette image, ainsi que les scores attribués par chacun d'entre eux en un temps t.

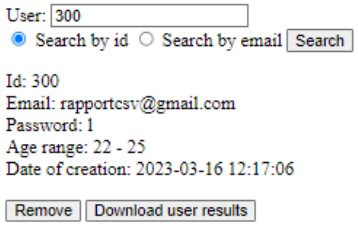


FIGURE 19 – Informations sur un utilisateur

Enfin, l'administrateur peut également charger un fichier contenant les informations de la base de données concernant les utilisateurs, mais aussi anonymiser ces informations pour tous les utilisateurs en effaçant les adresse mail et mot de passe par les identifiants respectifs de chacun.

5 Implémentation

En ce qui concerne le développement de l'application, nous allons présenter tout d'abord l'architecture du projet. Par la suite, l'implémentation de l'interface, de la base de données et des données des fichiers *CSV* seront abordés plus en détails.

5.1 Diagramme UML partie client

Dans cette section, l'architecture de notre outil est présentée grâce au diagramme UML sur la Figure 20 ci-dessous, effectué avec l'outil *Miro*.

Dans l'architecture de notre projet, un dossier *site* est créé, à partir duquel il y a deux dossiers : *front* pour le côté frontend, et *back* pour le côté backend de l'outil. Les classes en bleu représentent les pages web de l'interface de l'application dans le dossier *front/html/*, celles en vert symbolisent les scripts du côté frontend dans le dossier *front/js/*. Du côté backend, les classes en noir sont associées aux comptes utilisateur dans le dossier *back/account/*, celle en gris à la manipulation des fichiers *CSV* dans le dossier *back/csv/*, tandis que celles en

gris clair correspondent à certaines vérifications et mise en place de session dans le dossier *back/utils/*.

La classe jaune **Controller**, implémentée en PHP, se situe dans le dossier *site* et représente le script principal de notre architecture. En effet, ce script permet de définir les chemins d'accès, faire le lien entre certains scripts et les pages html, mais aussi de passer d'une page html à l'autre.

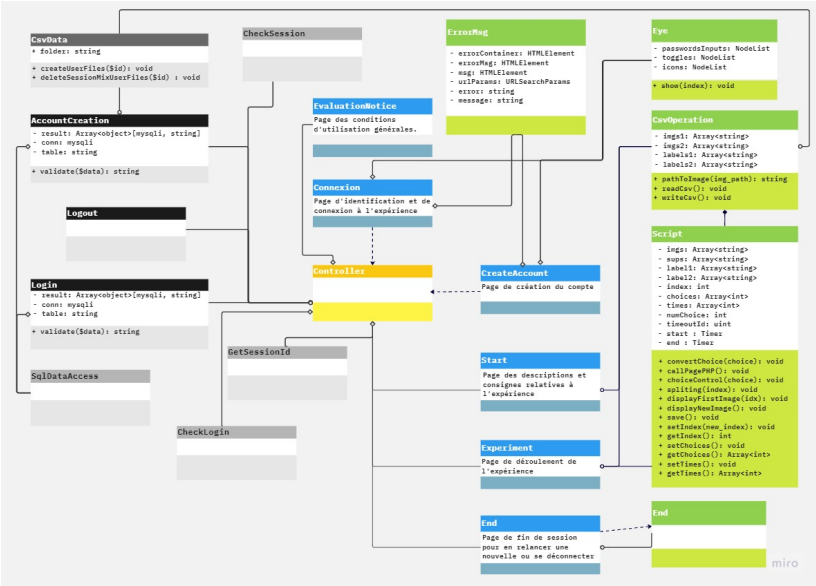


FIGURE 20 – Diagramme UML complet de l'architecture du projet partie client

5.1.1 Pages web de l'interface

Pour commencer, les pages en bleu concernent uniquement l'interface utilisateur. Elles sont implémentées grâce au langage HTML et stylisées à partir des fichiers CSS du même nom.

Tout d'abord, les pages **CreateAccount** et **Connexion** permettent d'afficher les formulaires liés à la création du compte pour la première, tandis que la deuxième est liée à la connexion à l'expérience. Pour créer un compte, il est nécessaire d'accepter les conditions générales d'utilisation de la page **EvaluationNotice**, liée à la page **CreateAccount**, pour ensuite se connecter à l'application.

La page **Connexion** se trouve dans le dossier *site*, puisque c'est la première page de notre interface.

Une fois connecté, la page **Start** sert à l'affichage des consignes relatives au déroulement de l'expérience, et permet également de démarrer une session, gérée par la page **Experiment**. C'est à partir de cette page que sont affichées les deux images, la vérité terrain et la classification des images, ainsi que la

colonne de boutons radio permettant à l'utilisateur d'évaluer les images.

Depuis la page **Experiment**, il y a deux chemins possibles :

— L'utilisateur termine l'expérience, il a alors accès à la page **End**.

— L'utilisateur se déconnecte de la session en cours, dans ce cas cela le redirige vers la page **Connexion**.

En ce qui concerne la page **End**, l'utilisateur peut relancer une session et fait donc appel à la page **Start**, ou bien se déconnecter de l'expérience en étant redirigé vers la page **Connexion**.

5.1.2 Scripts côté frontend

En ce qui concerne les classes en vert, côté client, celles-ci sont directement liées aux interactions entre l'utilisateur et les pages web de l'interface 5.1.1. Ces différents scripts sont implémentés en langage JavaScript.

Tout d'abord, le script **Eye** est directement lié aux pages **CreateAccount** et **Connexion** pour gérer le fait d'afficher, ou cacher, les caractères du mot de passe dans les formulaires respectifs des deux pages. Le script **ErrorMsg** est, lui aussi, lié à ces deux pages, mais gère les différentes erreurs qui peuvent intervenir suite à la création de compte, ou à la connexion, avec l'affichage de message d'erreurs. Par exemple, cela peut concerner une adresse mail déjà utilisée lors de la création du compte, ou encore la connexion avec une adresse mail incorrect.

Pour une session, le script **Script** gère directement la page **Experiment** dont le défilement des images, la réinitialisation visuelle, ou encore le stockage des choix et du temps t de l'utilisateur.

Ces deux dernières données sont directement transmises au script **CsvOperation**. Ce script sert à récupérer et envoyer des informations concernant les fichiers *CSV*. En effet, celui-ci permet de poster des requêtes côté serveur :

— requêtes "GET", envoyés depuis la page **Start**, pour générer les sessions et récupérer les images et classes à afficher depuis un fichier *CSV*. Celles-ci sont transmises ensuite au fichier **Script**.

— requête "POST", envoyé depuis la page **Experiment**, pour transmettre au serveur un tableau regroupant les choix, et temps t associés à ces choix, afin de les sauvegarder dans un fichier *CSV*.

Enfin, le script **End** est associé à la page du même nom **End**, et permet de faire varier les affichages de cette page selon la situation actuelle durant l'expérience. En effet, si l'utilisateur n'a pas fini l'expérience, il pourra en relancer une grâce à un bouton approprié, alors que, dans le cas contraire, il n'aura accès qu'au bouton de déconnexion avec un message lui indiquant la fin de l'expérience.

5.1.3 Base de données et fichiers

Dans cette partie, les classes sont implémentées en PHP, dont certaines utilisent l'extension MySQLi qui permet de faire le lien entre les programmes et la base de données MySQL.

Parmi les classes en noir, **AccountCreation** et **Login** sont toutes deux respectivement liées aux pages de création de compte **CreateAccount** et de connexion **Connexion**. Par le biais de requêtes "POST" liées aux formulaires, elles gèrent directement les informations transmises par l'utilisateur dans le formulaire d'inscription pour les enregistrer dans la base de données dans le premier cas, puis de vérifier les entrées du formulaire de connexion pour permettre l'accès au compte dans le deuxième cas. Ces deux classes établissent donc une connexion avec la base de données pour pouvoir en traiter les informations.

Le script **AccountCreation** fait appel à la classe grise **CsvData** pour générer les fichiers *CSV* avec l'identifiant utilisateur liés à la création de compte. Le script **CsvData** s'occupe de la modification des fichiers *CSV* et traite plus particulièrement les requêtes envoyées depuis le script **CsvOperation**. Elle permet de créer le scénario unique d'un utilisateur, générer des sessions différentes, et enfin sauvegarder les choix de l'utilisateur à l'aide de différents fichiers *CSV*.

Enfin, le script **Logout** permet simplement de déconnecter l'utilisateur de la base de données et intervient directement avec les pages **Experiment** et **End**.

5.1.4 Dossier utils

Tout d'abord, Les différents scripts de cette partie, sont eux, aussi implémentés en PHP. Le script **CheckLogin** est lié au script **Controller**, donc aux pages html, mais aussi aux classes noires liées à la base de données, pour vérifier à tout instant si l'utilisateur est correctement connecté à l'expérience. Le script **SqlDataAccess** défini la base de données, et est directement liée aux script **AccountCreation** et **Connexion** pour ajouter les utilisateurs et les connecter à cette base de données.

Enfin, le script **GetSessionId** permet de récupérer l'id de l'utilisateur actuellement connecté, tandis que **CheckSesion** permet de contrôler l'avancée durant l'expérience et savoir si celle-ci est finie ou non, notamment pour pouvoir lancer le script **End**.

5.2 Diagramme UML partie administrateur

La Figure 21 montre l'architecture de la partie administrateur. On y accède par le script **Login**, en saisissant les données appropriées, avant d'être ensuite redirigé vers le script **AdminConnexion**, qui charge la page html **Admin** requise. Toutes les fonctions de la page se trouvent dans le script **Admin**. Chaque fonctionnalité présente sur la page html et liée au script appelle le fichier php correspondant. Enfin, la base de données est accessible via le script **SqlDataAccess** et les contrôles d'accès font que seul l'hôte local peut accéder à la page elle-même et à ses fonctions.

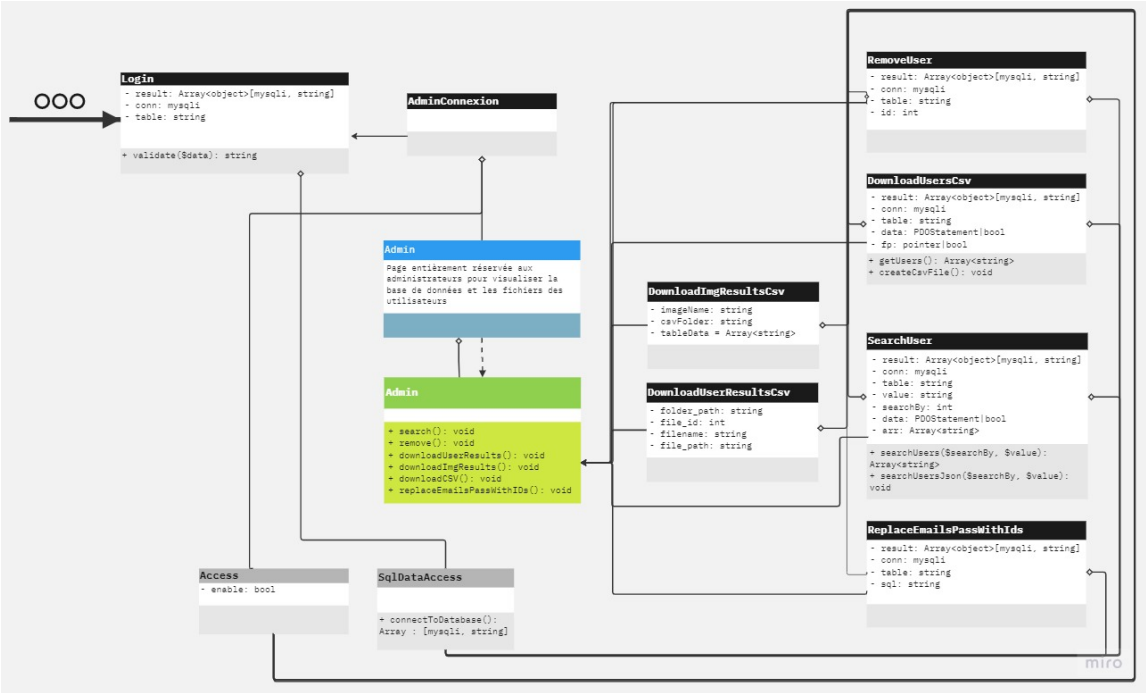


FIGURE 21 – Diagramme UML complet de l'architecture du projet partie administrateur

5.3 Organisation de l'interface

Cette section aborde les outils et contraintes que nous avons dû satisfaire pour composer l'interface de l'application.

5.3.1 Choix de présentation d'interface

La principale motivation dans la composition de l'interface de notre outil est de minimiser le nombre d'informations parvenant à l'utilisateur, et donc principalement la *fatigue visuelle* de ce dernier. De ce fait, le choix est fait de mettre en place une interface ergonomique, permettant de se focaliser uniquement sur les images qui doivent être évaluées.

Dans un premier temps, divers essais ont été faits concernant le choix des couleurs. En ce qui concerne l'arrière-plan, l'objectif est de minimiser la *fatigue visuelle* en utilisant un gris neutre, c'est pourquoi nous avons choisi d'utiliser un gris "neutre" (rgb(125,125,125)) cf. Figure 22, selon le nuancier de couleur [17]. Celui-ci reste le même tout au long des séances et notamment pour la partie réinitialisation visuelle entre deux évaluations d'images.

FIGURE 22 – Couleur d'arrière-plan et de réinitialisation visuelle.

De plus, pour conserver une certaine sobriété dans la présentation, la couleur de la police est définie en noir (#000000), tout en choisissant une taille suffisante, 18 ou 20 pixels, pour un certain confort visuel concernant la lisibilité.

Enfin, le choix de la police 'Arial' est uniformisé sur l'ensemble de l'interface pour ne pas mettre en avant certains détails textuels par rapport aux images. Sa taille est adaptée, tout comme celle des images, pour varier en fonction de la taille globale de la fenêtre, permettant ainsi de réaliser l'expérience sur différentes plateformes.

En ce qui concerne le côté pratique, la méthode d'évaluation doit être suffisamment claire pour l'utilisateur. Celle-ci est donc définie en colonne et la zone de sélection des boutons est élargie pour permettre au sujet de pouvoir facilement valider son choix dans le temps imparti.

5.3.2 Page de connexion et de création de compte

La page de connexion est un formulaire d'authentification de l'utilisateur. Il contient un champ de saisie de l'adresse électronique, un champ de saisie du mot de passe avec un bouton de basculement pour afficher/masquer le mot de passe, et un bouton de soumission. L'attribut action du formulaire spécifie le script côté serveur responsable du traitement des données du formulaire lors de la soumission. Dans ce cas, il s'agit de **back/account/login.php**. Les données du formulaire sont envoyées à l'aide de la méthode HTTP "POST". La *div* de la page avec l'id "error" contient un message d'erreur à afficher si l'utilisateur saisit un courriel ou un mot de passe incorrect, et contient également un lien permettant de récupérer le mot de passe. Enfin, un paragraphe avec l'identifiant "message" contient un lien vers une page de création de compte pour les utilisateurs qui n'ont pas encore de compte.

La page d'enregistrement d'utilisateur comprend un formulaire avec des champs pour l'adresse électronique, l'âge, le mot de passe et la confirmation du mot de passe, ainsi qu'une case à cocher pour l'acceptation des conditions générales du site. Il y a également un lien vers un document externe de conditions générales. Les données du formulaire sont soumises à un script PHP situé à l'adresse **../back/account/account_creation.php** lorsque l'utilisateur clique sur le bouton "Créer le compte". Le script est chargé de gérer la

création du compte de l'utilisateur en fonction des données saisies. La page comprend un style de base utilisant un fichier CSS externe situé à l'adresse **../css/create_account_style.css**. En outre, elle comprend un script permettant de basculer la visibilité des champs de mot de passe lorsque l'utilisateur clique sur l'icône en forme d'œil. L'icône de l'œil est une icône "Font Awesome" chargée à partir d'un *CDN*. Il existe également une fonction JavaScript appelée *checkPassword()* qui est déclenchée lorsque l'utilisateur soumet le formulaire.

5.3.3 Déroulement d'une session

Une fois la connexion à l'expérience aboutie, l'utilisateur se dirige donc sur la page de prise en compte des consignes relatives à l'expérience qui est représentée par le fichier **start.html**. Les consignes indiquent que lorsque nous cliquons sur le bouton "Start", l'expérience commence dans un premier temps par la session d'entraînement contenant 3 images, puis la session de test qui correspond à 240 paires d'images à évaluer. Une fois cliqué sur ce bouton, une nouvelle page nommée **experiment.html** est chargée. Cette page contient, à gauche, l'image qui a été classifiée, et à sa droite la carte d'explication superposée en carte de chaleur sur l'image. Enfin, sont affichés également la vérité terrain de la classe de l'image, la classe de l'image prédite par le classifieur et les boutons selon l'échelle de Likert. Toutes ces données sont contenues dans un "container" centré verticalement au centre de la page, explicité dans le fichier **experiment-style.css**.

Concernant la gestion de l'apparition des éléments de l'interface, la page **experiment.html** est liée à deux fichiers javascript. Le premier est le fichier **csv-operation.js** qui dans la fonction **readCsv** va faire une requête de type "GET" pour utiliser le contenu du fichier *session_user_id.csv*. Il va dans un premier temps créer les chemins d'accès aux images à l'aide d'une fonction **pathToImage**. Ces chemins vont être utilisés pour stocker, dans des tableaux, les images et classes pour chaque colonne du fichier session.

Une fois ces tableaux créés et remplis, ils vont être passés au fichier **script.js**. Ce dernier va tout d'abord afficher la première paire d'image à l'aide de la fonction **displayFirstImage** correspondant à l'indice 0. On peut voir que *timeoutId* est un entier positif qui identifie le minuteur créé par l'appel à *setTimeout()*. Le principe est que si on est inactif pendant plus de 20 secondes, cette fonction va automatiquement appeler la fonction **choiceControl**. On a donc finalement moins de 20 secondes pour faire un choix. Ensuite, la fonction **choiceControl** est la partie interaction avec l'utilisateur, c'est-à-dire que c'est cette fonction qui est appelée lorque l'utilisateur clique sur un bouton pour faire un choix. Le principe est le suivant :

— À l'indice 1 ou à l'indice $\geq$3, nous devons tout d'abord réinitialiser la variable *timeoutId*, puis cacher tous les élements (correspondant à la *div container*) pour afficher uniquement l'arrière-plan de couleur gris pendant 5 secondes avec la méthode *setTimeout()*. C'est ce qu'on appelle la phase de "réinitilisation visuelle". Cela nous amène donc à une dernière fonction nommée **displayNewImage** qui va permettre d'afficher

les images et classes suivantes.

— À l'indice ≥3, le choix et le temps t pris par l'utilisateur sont stockés au moment où la fonction **displayNewImage** est appelé, jusqu'au moment où on clique sur un bouton radio. Ce temps t est calculé grâce à l'objet javascript *Date()*.
— Dans la paire d'image représentant la séparation entre la phase d'entraînement et d'évaluation (entre l'indice 2 et 3), les éléments de l'interface (*div container*) sont cachés pour laisser place à une page uniquement composée d'un bouton "Start". Le *timeoutId* est donc réinitialisé à 0, et la phase de test peut être lancée grâce au bouton.
— Si toutes les images ont été évaluées, les choix de l'utilisateur sont sauvegardés à l'aide de la fonction *save* avant d'appeler la page de fin de session **end.html**. La fonction *save* est aussi appelée lorsqu'on clique sur le bouton déconnexion qui est affiché durant l'évaluation. Elle appelle la fonction *writeCsv* qui concatène les deux tableaux de choix et de temps, puis envoie une requête "POST" pour écrire dans le fichier *save_user_id.csv* les deux colonnes choix et temps.

La fonction **displayNewImage** est l'action d'afficher les prochaines paires d'images avec les labels et le bouton de déconnexion en bas à droite de la page. La requête *ajax* est utilisée pour cacher les chemins d'accès aux images au format *.jpg* et d'unifier les accès en un seul endroit. Le but de cette fonction est donc d'afficher les tableaux d'images et de labels récupérés à partir du script **csv-operation.js** à l'indice spécifié.

5.4 Données des utilisateurs

Pour que les utilisateurs puissent participer à l'expérience, ils doivent dans un premier temps créer un compte pour pouvoir accéder à l'outil. Dans le cadre de notre projet, des données telles que l'adresse mail, la tranche d'âge et le mot de passe, créé par l'utilisateur, doivent être conservées pour lui permettre de pouvoir se connecter à tout moment à l'outil. Pour gérer ces données, le système de gestion de base de données MySQL est utilisé, avec l'application Web phpMyAdmin.

5.4.1 Base de données

La création de compte se fait grâce à PHP, qui utilise l'extension MySQLi, pour transmettre les données de l'utilisateur dans une base de données MySQL. Il commence par inclure un fichier appelé **csv_data.php**, et définit ensuite une fonction appelée *validate* qui supprime les caractères inutiles des entrées utilisateur et prévient les attaques par injection SQL. Le script définit ensuite les paramètres de connexion à la base de données et établit une connexion avec celle-ci. Il vérifie que la connexion est réussie et récupère les données d'un formulaire que l'utilisateur a rempli. Il valide les données et vérifie si le courriel existe déjà dans la base de données. S'il n'existe pas, il insère les données de l'utilisateur dans la base de données et crée des fichiers pour l'utilisateur à l'aide

de la fonction *createUserFiles* du fichier **csv_data.php**. Il met également à jour les colonnes "email" et "mot de passe" avec l'ID de l'utilisateur après 6 mois. En cas d'erreur, le script affiche un message d'erreur. Enfin, il ferme la connexion à la base de données.

Un autre script PHP valide et traite les données de connexion des utilisateurs. Le script définit d'abord une fonction appelée *validate* qui sépare puis supprime les barres obliques et convertit les caractères spéciaux en entités HTML dans les données d'entrée, puis renvoie le résultat. Il définit ensuite les paramètres de connexion à la base de données et tente de se connecter à la base de données à l'aide de la fonction *mysqli_connect*. Si la connexion échoue, un message d'erreur s'affiche et le script se termine. Si la connexion réussit, le script récupère l'adresse électronique et le mot de passe de l'utilisateur à partir d'un formulaire de connexion utilisant la variable globale "$_POST", valide les données à l'aide de la fonction *validate* et vérifie si les données correspondent à un "login" administrateur prédéfini. Si la connexion est réussie, l'utilisateur est redirigé vers une page d'administration. Dans le cas contraire, le script interroge la base de données pour vérifier si la combinaison de l'adresse électronique et du mot de passe correspond à un utilisateur existant. Si une correspondance est trouvée, la session de l'utilisateur est lancée et son identifiant est stocké dans une variable de session. En fonction de l'existence ou non de certains fichiers de données, l'utilisateur est redirigé vers une page de début ou de fin d'expérience. Si aucune correspondance n'est trouvée, un message d'erreur s'affiche et le script se termine. Enfin, la connexion à la base de données est fermée à l'aide de la fonction *mysql_close*.

5.5 Données sur les images

L'expérience de chaque utilisateur doit être unique. Pour le déroulement de l'expérience, plusieurs paramètres sont déterminés :

— chaque utilisateur doit évaluer 1800 images
— une session se compose de 243 images

Contenant un grand nombre d'images dans la base de données originelle, le nombre d'images à évaluer est réduit au nombre de 1800. Pour une session, comme discuté dans la section 2, celle-ci ne doit pas durer plus d'une heure. C'est pourquoi, en partant du principe que le temps "idéal" pour évaluer une image, en comptant la réinitialisation visuelle, est de 15 secondes, il est alors possible d'afficher 240 images par sessions, en plus des 3 images d'entraînement. Avec ces dispositions, cela signifie que l'expérience se divise en 8 sessions, chacune composée de 243 images, avec la dernière se composant uniquement de 123 images.

Dans ce cas, trois types de fichier sont créés :

— un fichier aléatoire avec les 1800 images à évaluer dans un ordre aléatoire, et les 24 images d'entraînement
— un fichier de session avec les 243 images présentées durant une session
— un fichier de sauvegarde dans lequel toutes les images évaluées sont ajoutées avec le choix et le temps t pris pour faire ce choix.

L'arborescence suivante est alors établie, dans laquelle chaque type de fichier est regroupé dans un dossier spécifique, comme on peut le voir sur la Figure 23 :

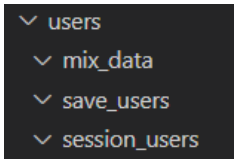


FIGURE 23 – Arborescence des dossiers des fichiers *CSV*

Pour pouvoir les différencier, tous les fichiers sont créés lors de la création du compte et comportent, dans leur nom, l'identifiant *id* de l'utilisateur pour qu'ils soient propres à chacun.

Enfin, pour gérer la manipulation et les modifications des fichiers, des requêtes sont envoyées au serveur. Pour cela, nous utilisons l'API *Fetch* pour envoyer ces requêtes et avoir accès au contenu des fichiers *CSV* côté interface. Pour sécuriser ces requêtes, et notamment les chemins d'accès vers ces fichiers, le modèle *AJAX* est utilisé pour empêcher de pouvoir accéder à ces fichiers sans passer par les requêtes. Enfin, les opérations sur les fichiers telles que l'ouverture, la lecture ou l'écriture se font à partir des fonctions PHP-CSV appropriées.

5.6 Fichiers de données images

En ce qui concerne les fichiers sources contenant le jeu de données, nous avons créé les fichiers *CSV* *uni_test.csv* et *uni_train.csv*. Le premier contient les 1800 images choisies qui doivent être évaluées par les utilisateurs, tandis que le deuxième fichier contient les 24 images qui servent à l'entraînement.

Il y a 4 types d'images possibles dans les données :

— blur
— brightness
— geometric
— noise

Ce sont les types d'altérations appliquées sur les images dans le jeu de données. Le nombre d'images de chaque type dans les fichiers *uni_test.csv* et *uni_train.csv* est réparti afin de permettre de s'entraîner et d'évaluer chacune des dégradations équitablement lors d'une expérience.

De plus, les fichiers *CSV* sont définis selon les paramètres suivants :

— "Key" - clé
— "Noisy_image" - image d'origine
— "Sup_image" - image avec carte d'explication
— "Distorsion" - altération appliquée à l'image
— "Explanation_method" - méthode d'explication
— "Noisy_image_class" - classification de l'image
— "Original_image_class" - vérité terrain de l'image

Ce sont ces paramètres qui sont lus dans les fichiers *CSV* pour transmettre les données côté interface, afin de pouvoir afficher les images et les classes au cours de l'expérience.

5.6.1 Fichier aléatoire

Durant l'expérience, un utilisateur doit donc évaluer 1800 images. Pour ce faire, à partir du fichier *uni_ test.csv*, contenant toutes les images et classes à notre disposition, nous générons le fichier aléatoire *mixed_ data_ user_ id.csv*. Lors de la création de compte, nous ouvrons le fichier *uni_ test.csv*, mélangeons les données contenues dans ce fichier, avant de les transmettre au nouveau fichier source *mixed_ data_ user_ id.csv* de l'utilisateur. L'utilisateur doit également visualiser des images d'entraînement au nombre de 3 par session, pour un nombre total de 24. Ces 24 images d'entraînement sont contenues à partir du fichier *uni_ train.csv* et n'ont pas besoin d'être attribuées dans un ordre aléatoire.

Pour ensuite simplifier la création du fichier de session, le fichier *mixed_ data_ user_ id.csv* est composé des 3 premières images contenues dans le fichier *uni_ train.csv*, puis des 240 images premières images du fichier *uni_ test.csv*, et ainsi de suite de manière récursive. Dans l'exemple ci-dessous la Figure 24, 3 images sont utilisées pour les entraînements, mais 6 images d'évaluation pour chaque session. On peut alors voir l'organisation du fichier *mixed_ data_ user_ id.csv* avec, en bleu, les clés des lignes utilisées pour l'entraînement, tandis qu'en rouge, ce sont celles utilisées pour la phase d'évaluation.

	Key	Noisy_image	Sup_image	Distortion	Explanation_Method	Noisy_image_class	Original_image_class
2	0	COCO_train2014_000000003671_blur-k-2-0.jpg	COCO_train2014_000000003671_blur-k-2-0_GRAD-CAM_sup.jpg	blur-k-2-0	GRAD-CAM	vase	cup_bowl_cake_dining table
3	1	COCO_train2014_000000003671_blur-k-2-0.jpg	COCO_train2014_000000003671_blur-k-2-0_FEM_sup.jpg	blur-k-2-0	FEM	vase	cup_bowl_cake_dining table
4	2	COCO_train2014_000000003671_blur-k-2-0.jpg	COCO_train2014_000000003671_blur-k-2-0_ML-FEM_sup.jpg	blur-k-2-0	ML-FEM	vase	cup_bowl_cake_dining table
5	217	COCO_train2014_000000001424_noise-k-25-0.jpg	COCO_train2014_000000001424_noise-k-25-0_FEM_sup.jpg	noise-k-25-0	FEM	teddy bear	person_knife_cake_chair_potted plant_dining table_vase
6	263	COCO_train2014_000000001670_noise-k-25-0.jpg	COCO_train2014_000000001670_noise-k-25-0_ML-FEM_sup.jpg	noise-k-25-0	ML-FEM	person	person_backpack_skis
7	18	COCO_train2014_000000000110_geometric-k-1-0.jpg	COCO_train2014_000000000110_geometric-k-1-0_GRAD-CAM_sup.jpg	geometric-k-1-0	GRAD-CAM	person	person_cup_fork_knife_pizza_chair_dining table
8	229	COCO_train2014_000000001510_geometric-k-5-0.jpg	COCO_train2014_000000001510_geometric-k-5-0_FEM_sup.jpg	geometric-k-5-0	FEM	broccoli	broccoli_dining table
9	66	COCO_train2014_000000000389_geometric-k-5-0.jpg	COCO_train2014_000000000389_geometric-k-5-0_GRAD-CAM_sup.jpg	geometric-k-5-0	GRAD-CAM	person	person_backpack_tie
10	217	COCO_train2014_000000001424_brightness-k-25-0.jpg	COCO_train2014_000000001424_brightness-k-25-0_FEM_sup.jpg	brightness-k-25-0	FEM	teddy bear	person_knife_cake_chair_potted plant_dining table_vase
11	3	COCO_train2014_000000003729_blur-k-2-0.jpg	COCO_train2014_000000003729_blur-k-2-0_GRAD-CAM_sup.jpg	blur-k-2-0	GRAD-CAM	cat	cat_suitcase_bed
12	4	COCO_train2014_000000003729_blur-k-2-0.jpg	COCO_train2014_000000003729_blur-k-2-0_FEM_sup.jpg	blur-k-2-0	FEM	cat	cat_suitcase_bed
13	5	COCO_train2014_000000003729_blur-k-2-0.jpg	COCO_train2014_000000003729_blur-k-2-0_ML-FEM_sup.jpg	blur-k-2-0	ML-FEM	cat	cat_suitcase_bed
14	6	COCO_train2014_000000000009_geometric-k-10-0.jpg	COCO_train2014_000000000009_geometric-k-10-0_GRAD-CAM_sup.jpg	geometric-k-10-0	GRAD-CAM	person	bowl_orange_broccoli
15	223	COCO_train2014_000000001424_geometric-k-10-0.jpg	COCO_train2014_000000001424_geometric-k-10-0_FEM_sup.jpg	geometric-k-10-0	FEM	person	person_knife_cake_chair_potted plant_dining table_vase
16	53	COCO_train2014_000000000349_blur-k-6-0.jpg	COCO_train2014_000000000349_blur-k-6-0_ML-FEM_sup.jpg	blur-k-6-0	ML-FEM	tv	train_potted plant
17	194	COCO_train2014_000000001145_blur-k-2-0.jpg	COCO_train2014_000000001145_blur-k-2-0_ML-FEM_sup.jpg	blur-k-2-0	ML-FEM	person	person_bottle_cup_knife_spoon_bowl_chair_couch_potted plant_dining table_oven_refrigerator_book
18	278	COCO_train2014_000000001756_noise-k-175-0.jpg	COCO_train2014_000000001756_noise-k-175-0_ML-FEM_sup.jpg	noise-k-175-0	ML-FEM	baseball bat	person_handbag_bottle_cup_chair_laptop
19	286	COCO_train2014_000000001785_brightness-k-175-0.jpg	COCO_train2014_000000001785_brightness-k-175-0_FEM_sup.jpg	brightness-k-175-0	FEM	kite	person_backpack_sports ball_kite

FIGURE 24 – Exemple création fichier *mixed_ data_ user_ id.csv*

5.6.2 Fichier de session

Le fichier de session est celui contenant les 243 images à évaluer durant une session, 123 images dans le cas de la dernière session. Le contenu du fichier *session_ user_ id.csv* est modifié à chaque fois que l'utilisateur charge la page de description des consignes. Cette opération n'a lieu seulement si le fichier est vide, c'est-à-dire que l'utilisateur a entièrement effectué la session précédente sans la quitter. Dans ce cas, on extrait les 243 premières images du fichier *mixed_ data_ user_ id.csv* en les supprimant, pour les stocker dans le fichier *session_ user_ id.csv*.

Sur la figure 26 ci-dessous, nous considérons les lignes d'images extraites de l'exemple précédent (cf. Figure 24), tandis que le fichier *mixed_ data_ user_ id.csv* est modifié selon la Figure 25 ci-dessous. Une session est composée dans cet exemple de 9 images : 3 pour l'entraînement et 6 pour l'évaluation.

	Key	Noisy_image	Sup_image	Distortion	Explanation_Method	Noisy_image_class	Original_image_class
2	3	COCO_train2014_000000003729_blur-k-2-0.jpg	COCO_train2014_000000003729_blur-k-2-0_GRAD-CAM_sup.jpg	blur-k-2-0	GRAD-CAM	cat	cat_suitcase_bed
3	4	COCO_train2014_000000003729_blur-k-2-0.jpg	COCO_train2014_000000003729_blur-k-2-0_FEM_sup.jpg	blur-k-2-0	FEM	cat	cat_suitcase_bed
4	5	COCO_train2014_000000003729_blur-k-2-0.jpg	COCO_train2014_000000003729_blur-k-2-0_ML-FEM_sup.jpg	blur-k-2-0	ML-FEM	cat	cat_suitcase_bed
5	6	COCO_train2014_000000000009_geometric-k-10-0.jpg	COCO_train2014_000000000009_geometric-k-10-0_GRAD-CAM_sup.jpg	geometric-k-10-0	GRAD-CAM	person	bowl_orange_broccoli
6	223	COCO_train2014_000000001424_geometric-k-10-0.jpg	COCO_train2014_000000001424_geometric-k-10-0_FEM_sup.jpg	geometric-k-10-0	FEM	person	person_knife_cake_chair_potted plant_dining table_vase
7	53	COCO_train2014_000000000349_blur-k-6-0.jpg	COCO_train2014_000000000349_blur-k-6-0_ML-FEM_sup.jpg	blur-k-6-0	ML-FEM	tv	train_potted plant
8	194	COCO_train2014_000000001145_blur-k-2-0.jpg	COCO_train2014_000000001145_blur-k-2-0_ML-FEM_sup.jpg	blur-k-2-0	ML-FEM	person	person_bottle_cup_knife_spoon_bowl_chair_couch_potted plant_dining table_oven_refrigerator_book
9	278	COCO_train2014_000000001756_noise-k-175-0.jpg	COCO_train2014_000000001756_noise-k-175-0_ML-FEM_sup.jpg	noise-k-175-0	ML-FEM	baseball bat	person_handbag_bottle_cup_chair_laptop
10	286	COCO_train2014_000000001785_brightness-k-175-0.jpg	COCO_train2014_000000001785_brightness-k-175-0_FEM_sup.jpg	brightness-k-175-0	FEM	kite	person_backpack_sports ball_kite

FIGURE 25 – Fichier *mixed_ data_ user_ id.csv* après remplissage du fichier *session_ user_ id.csv*

	Key	Noisy_image	Sup_image	Distortion	Explanation_Method	Noisy_image_class	Original_image_class
2	0	COCO_train2014_000000003671_blur-k-2-0.jpg	COCO_train2014_000000003671_blur-k-2-0_GRAD-CAM_sup.jpg	blur-k-2-0	GRAD-CAM	vase	cup_bowl_cake_dining table
3	1	COCO_train2014_000000003671_blur-k-2-0.jpg	COCO_train2014_000000003671_blur-k-2-0_FEM_sup.jpg	blur-k-2-0	FEM	vase	cup_bowl_cake_dining table
4	2	COCO_train2014_000000003671_blur-k-2-0.jpg	COCO_train2014_000000003671_blur-k-2-0_ML-FEM_sup.jpg	blur-k-2-0	ML-FEM	vase	cup_bowl_cake_dining table
5	217	COCO_train2014_000000001424_noise-k-25-0.jpg	COCO_train2014_000000001424_noise-k-25-0_FEM_sup.jpg	noise-k-25-0	FEM	teddy bear	person_knife_cake_chair_potted plant_dining table_vase
6	263	COCO_train2014_000000001670_noise-k-25-0.jpg	COCO_train2014_000000001670_noise-k-25-0_ML-FEM_sup.jpg	noise-k-25-0	ML-FEM	person	person_backpack_skis
7	18	COCO_train2014_000000000110_geometric-k-1-0.jpg	COCO_train2014_000000000110_geometric-k-1-0_GRAD-CAM_sup.jpg	geometric-k-1-0	GRAD-CAM	person	person_cup_fork_knife_pizza_chair_dining table
8	229	COCO_train2014_000000001510_geometric-k-5-0.jpg	COCO_train2014_000000001510_geometric-k-5-0_FEM_sup.jpg	geometric-k-5-0	FEM	broccoli	broccoli_dining table
9	66	COCO_train2014_000000000389_geometric-k-5-0.jpg	COCO_train2014_000000000389_geometric-k-5-0_GRAD-CAM_sup.jpg	geometric-k-5-0	GRAD-CAM	person	person_backpack_tie
10	217	COCO_train2014_000000001424_brightness-k-25-0.jpg	COCO_train2014_000000001424_brightness-k-25-0_FEM_sup.jpg	brightness-k-25-0	FEM	teddy bear	person_knife_cake_chair_potted plant_dining table_vase

FIGURE 26 – Fichier *session_ user_ id.csv* pour une session

Cette modification de fichier est faite par requête envoyée au serveur. En effet, nous passons par la requête "GET" pour demander au serveur de modifier les fichiers en lecture/écriture. La requête renvoie l'identifiant *id* de l'utilisateur actuel pour savoir quel fichier de session doit être lu. Pour pouvoir utiliser les données du fichier depuis l'interface, nous envoyons une nouvelle requête "GET" au serveur pour demander l'accès au contenu du fichier nommé avec l'identifiant *id* récupéré grâce à la première requête. Ainsi, ce contenu est renvoyé dans le corps de la requête, données alors directement traitées du côté de l'interface.

Côté interface, une fois les données récupérées, la bibliothèque *Papaparse* est utilisée pour parser le contenu du fichier de session, et ainsi récupérer les noms des images et des classes à afficher pour la session.

5.6.3 Fichier de sauvegarde

Au cours de l'expérience, l'utilisateur effectue les évaluations des différentes images qui lui sont présentées en un temps t. Du côté interface, ces données sont stockées dans deux tableaux respectifs : *choices* et *times*. Une fois ces derniers scindés, le tableau *stats* est transmis au serveur dans le corps de la requête "POST". Côté serveur, nous pouvons alors récupérer le tableau transmis. Ensuite, nous ouvrons les deux fichiers de session et de sauvegarde. Hormis les 3 premières images d'entraînement, toutes les lignes contenues dans le fichier de session *session_ user_ id.csv* (cf. Figure 26) sont recopiées pour les transmettre au fichier *save_ user_ id.csv* (cf. Figure 27), en ajoutant deux colonnes :

— les choix : convertis à 0 si aucun choix, sinon respectivement de 1 à 5 selon l'échelle d'évaluation
— les temps : chaque temps t est calculé en secondes et arrondi à l'entier le plus proche

	Key	Noisy_image	Sup_image	Distortion	Explanation_Method	Noisy_image_class	Original_image_class	Choices	Times
2	217	COCO_train2014_000000001424_noise-k-25-0.jpg	COCO_train2014_000000001424_noise-k-25-0_FEM_sup.jpg	noise-k-25-0	FEM	teddy bear	person_knife_cake_chair_potted plant_dining table_vase	5	4
3	263	COCO_train2014_000000001670_noise-k-25-0.jpg	COCO_train2014_000000001670_noise-k-25-0_ML-FEM_sup.jpg	noise-k-25-0	ML-FEM	person	person_backpack_skis	4	12
4	18	COCO_train2014_000000000110_geometric-k-1-0.jpg	COCO_train2014_000000000110_geometric-k-1-0_GRAD-CAM_sup.jpg	geometric-k-1-0	GRAD-CAM	person	person_cup_fork_knife_pizza_chair_dining table	2	6
5	229	COCO_train2014_000000001510_geometric-k-5-0.jpg	COCO_train2014_000000001510_geometric-k-5-0_FEM_sup.jpg	geometric-k-5-0	FEM	broccoli	broccoli_dining table	4	8
6	66	COCO_train2014_000000000389_geometric-k-5-0.jpg	COCO_train2014_000000000389_geometric-k-5-0_GRAD-CAM_sup.jpg	geometric-k-5-0	GRAD-CAM	person	person_backpack_tie	1	13
7	217	COCO_train2014_000000001424_brightness-k-25-0.jpg	COCO_train2014_000000001424_brightness-k-25-0_FEM_sup.jpg	brightness-k-25-0	FEM	teddy bear	person_knife_cake_chair_potted plant_dining table_vase	1	7

FIGURE 27 – Fichier *save_user_id.csv* à la fin d'une session

Dans le cas où la moitié de la taille du tableau *stats* est inférieur au nombre de lignes dans le fichier *session_user_id.csv*, cela signifie que la session n'a pas été finalisé. Dans ce cas, seulement les images évaluées sont supprimées du fichier de session, voir la Figure 28, et ajoutées au fichier de sauvegarde, la Figure 29 ci-dessous. Dans l'exemple ci-dessous, nous quittons la session en cours après avoir passé l'entraînement et évalué les deux premières images du fichier de la Figure 26.

	Key	Noisy_image	Sup_image	Distortion	Explanation_Method	Noisy_image_class	Original_image_class
2	0	COCO_train2014_000000003671_blur-k-2-0.jpg	COCO_train2014_000000003671_blur-k-2-0_GRAD-CAM_sup.jpg	blur-k-2-0	GRAD-CAM	vase	cup_bowl_cake_dining table
3	1	COCO_train2014_000000003671_blur-k-2-0.jpg	COCO_train2014_000000003671_blur-k-2-0_FEM_sup.jpg	blur-k-2-0	FEM	vase	cup_bowl_cake_dining table
4	2	COCO_train2014_000000003671_blur-k-2-0.jpg	COCO_train2014_000000003671_blur-k-2-0_ML-FEM_sup.jpg	blur-k-2-0	ML-FEM	vase	cup_bowl_cake_dining table
5	18	COCO_train2014_000000000110_geometric-k-1-0.jpg	COCO_train2014_000000000110_geometric-k-1-0_GRAD-CAM_sup.jpg	geometric-k-1-0	GRAD-CAM	person	person_cup_fork_knife_pizza_chair_dining table
6	229	COCO_train2014_000000001510_geometric-k-5-0.jpg	COCO_train2014_000000001510_geometric-k-5-0_FEM_sup.jpg	geometric-k-5-0	FEM	broccoli	broccoli_dining table
7	66	COCO_train2014_000000000389_geometric-k-5-0.jpg	COCO_train2014_000000000389_geometric-k-5-0_GRAD-CAM_sup.jpg	geometric-k-5-0	GRAD-CAM	person	person_backpack_tie
8	217	COCO_train2014_000000001424_brightness-k-25-0.jpg	COCO_train2014_000000001424_brightness-k-25-0_FEM_sup.jpg	brightness-k-25-0	FEM	teddy bear	person_knife_cake_chair_potted plant_dining table_vase

FIGURE 28 – Fichier *save_user_id.csv* après avoir quitté une session

	Key	Noisy_image	Sup_image	Distortion	Explanation_Method	Noisy_image_class	Original_image_class	Choices	Times
2	217	COCO_train2014_000000001424_noise-k-25-0.jpg	COCO_train2014_000000001424_noise-k-25-0_FEM_sup.jpg	noise-k-25-0	FEM	teddy bear	person_knife_cake_chair_potted plant_dining table_vase	5	4
3	263	COCO_train2014_000000001670_noise-k-25-0.jpg	COCO_train2014_000000001670_noise-k-25-0_ML-FEM_sup.jpg	noise-k-25-0	ML-FEM	person	person_backpack_skis	4	12
4									

FIGURE 29 – Fichier *save_user_id.csv* après avoir quitté une session

Cela signifie également que lors de la prochaine connexion, le fichier de session n'a pas besoin d'être mis à jour et ainsi conserver l'avancée lors de la précédente session. L'utilisateur procède de nouveau à la phase d'entraînement. Dans le cas contraire, le fichier de session est vidé, permettant de pouvoir le remplir lors de la prochaine session.

5.6.4 Conservation des fichiers

En fin d'expérience, le fichier le plus important est celui contenant les évaluations de l'utilisateur, le fichier de sauvegarde. Concernant les deux autres fichiers, aléatoire et de session, ils sont tous deux supprimés dans le seul cas où ils sont tous les deux vides :

— si le fichier aléatoire est vide, cela signifie qu'il n'y a pas d'autre session à effectuer.
— si le fichier de session est vide, cela signifie qu'il n'y a pas de session en cours.

6 Partie tests

L'écriture des tests est basée sur les différentes tâches à effectuer en rapport avec le projet. Ceux-ci dépendent donc des fonctionnalités des éléments principaux de l'outil telles que l'affichage des éléments, les choix utilisateurs, les fichiers *CSV* ou encore la base de données, et sont répartis par fichiers. L'énumération de ces tests dans chacune des sections ci-dessous reprend l'ordre établi dans les fichiers de tests. Des besoins non-fonctionnels tels que les temps d'exécution, ou la portabilité, sont également présentés plus bas.

6.1 Tests Unitaires

Pour l'implémentation des tests unitaires, nous utilisons le framework de test appelé **Jest**. Les méthodes utilisés pour le développement de l'application implique également l'installation de plusieurs bibliothèques listées ci-dessous :
— fast-text-encoding
— text-encoding
— jest-environment-jsdom
— jquery
— mysql
— node-localstorage
— papaparse

Enfin, le lancement de ces tests a été effectué sous les environnements *Linux* et *Windows*.

6.1.1 Tests sur les fichiers *CSV*

Dans cette partie, nous effectuons des tests concernant l'utilisation et la modification des fichiers *CSV* durant l'expérience, contenus dans le fichier **csv_data.test.js**. Pour ces tests, deux comptes différents sont créés en copiant les contenus des fichiers de session et aléatoire générés pour pouvoir conserver puis, par la suite, comparer le contenu de ces copies selon les situations des tests. Ces copies sont dans le dossier /**data**/**csv_tests**/ et seront évoquées dans les tests suivants comme fichiers d'origines.

Test 1 : Création fichier aléatoire

Description et objectif : vérifier que le fichier aléatoire a bien été crée.
Données utilisées et résultats attendus :
— Données utilisées : fichier mixed_data d'origine (id :404) et fichier uni_test.csv

— Résultat attendu : le fichier mixed_data contient 1824 lignes dans un ordre aléatoire.
Scénario du test : On vérifie si le fichier mix_data a été crée. Si il contient 1824 lignes qui correspondent à la totalité des images de l'expérience et si les lignes sont mélangées en comparant avec les lignes de uni_test.csv.
Analyse du test :
— Si il contient 1824 lignes dans un ordre aléatoires. -> OK
— Le fichier est vide ou n'est pas mélangé ->erreur.
Discussion des résultats : Le fichier existe et contient 1824 lignes dans un ordre aléatoire.
Résultat correct.

Test 2 : Lecture contenu fichier aléatoire

Description et objectif : vérifier que le fichier aléatoire contient les bonnes lignes.
Données utilisées et résultats attendus :
— Données utilisées : fichier mixed_data d'origine (id :404), fichier uni_test.csv et fichier uni_train.csv.
— Résultat attendu : le fichier mixed_data contient les lignes du fichier uni_train et les lignes du fichier uni_test.
Scénario du test : On vérifie si le fichier mixed_data contient les lignes de uni_train. (3 lignes toutes les 240 lignes). De même pour le uni_test.csv.
Analyse du test :
— Si il contient les lignes des 2 fichiers uni. -> OK
— Si il ne contient pas les lignes des 2 fichiers ou les lignes ne sont pas les mêmes-> erreur.
Discussion des résultats : Le fichier mixed_data contient les bonnes lignes.
Résultat correct.

Test 3 : Lecture contenu fichier session

Description et objectif : vérifier que le fichier session contient les bonnes lignes (images).
Données utilisées et résultats attendus :
— Données utilisées : fichier de session et mixed_data d'origines (id :404).
— Résultat attendu : le fichier session est généré et contient bien 243 lignes provenant du fichier mixed_data.
Scénario du test : On vérifie si le fichier session contient 243 lignes. On regarde ensuite si le fichier de session contient les 243 premières lignes de mixed_data.
Analyse du test :
— Si il contient 243 premières lignes du fichier mixed_data. -> OK
— Si il ne contient pas les lignes de mixed_data ou si elles sont différentes-> erreur.

Discussion des résultats : Le fichier session contient les bonnes lignes.
Résultat correct.

Test 4 : Chemin vers les images

Description et objectif : vérifier que le chemin d'accès vers les images est le bon.
Données utilisées et résultats attendus :
— Données utilisées : fichier session d'origine (id :404) dossier ../site/back/data/images/ et fonction pathToImage dans le fichier csv_operation.csv.
— Résultat attendu : les images noisy et sup respectivement colonne 1 et 2 correspondent bien aux images dans le dossier images.
Scénario du test : On vérifie si les images du fichier session correspondent bien aux images qui suivent le chemin ../site/back/data/images/. Soit dans le dossier blur, soit dans le dossier noise.
Analyse du test :
— Si le chemin est le bon. -> OK
— Si les images qui suivent le chemin d'accès sont différentes des images du fichier session -> erreur.
Discussion des résultats : Le fichier session contient les bonnes images.
Résultat correct.

Test 5 : Suppression des images fichier aléatoire

Description et objectif : vérifier que la 1ere session est bien supprimée dans le fichier mixed_data.
Données utilisées et résultats attendus :
— Données utilisées : fichier mixed_data_user (id :409) et fichier uni_train.csv.
— Résultat attendu : Lorsqu'on relance une deuxième session on veut enlever les lignes de la première session dans mixed_data .
Scénario du test : On vérifie qu'il y a désormais 1582 lignes dans mixed_data. On vérifie que les 3 premières lignes de mixed_data sont égales aux lignes 5,6,7 de uni_train.csv .
Analyse du test :
— Si le fichier aléatoires toutes les sessions sauf la première. -> OK
— Si il contient la 1ère session -> erreur.
Discussion des résultats : Les lignes de la première session sont supprimées dans mixed_data.
Résultat correct.

Test 6 : Création fichier de sauvegarde

Description et objectif : Vérifier que le fichier de sauvegarde est créé avec

l'identifiant de l'utilisateur.
Données utilisées et résultats attendus :
— Données utilisées : fichier de sauvegarde et identifiant utilisateur.
— Résultat attendu : le fichier de sauvegarde est correctement créé avec, en son nom, l'identifiant de l'utilisateur (404).
Scénario du test : On crée un compte utilisateur. On regarde alors si le fichier de sauvegarde a bien été créé et qu'il est nommé avec un identifiant utilisateur.
Analyse du test :
— le fichier de sauvegarde a bien été créé et est nommé l'identifiant utilisateur -> OK
— le fichier de sauvegarde n'existe pas ou n'est pas nommé avec l'identifiant de l'utilisateur -> erreur.
Discussion des résultats : Le fichier de sauvegarde existe et a bien nommé avec l'identifiant utilisateur.
Résultat correct.

Test 7 : Contenu fichier sauvegarde en fin de session

Description et objectif : Vérifier le contenu du fichier *CSV* de sauvegarde à la fin d'une session.
Données utilisées et résultats attendus :
— Données utilisées : fichiers de sauvegarde et de session d'origine (id : 404).
— Résultat attendu : le fichier de sauvegarde doit contenir les lignes du fichier de session avec deux colonnes supplémentaires pour les choix et temps.
Scénario du test : On récupère le compte précédemment créé et on effectue une session entière d'évaluation des images. On vérifie ensuite si le fichier de sauvegarde contient 240 lignes supplémentaires correspondant aux images évaluées, appartenant à session d'origine et sans compter celles de l'entraînement, avec les choix et temps utilisateurs qui y sont ajoutés. Les choix doivent être compris entre 0 et 5, et les temps t entre 0 et 20.
Analyse du test :
— le fichier de sauvegarde contient les mêmes 240 lignes que le fichier de session avant évaluation, en plus des choix et temps -> OK
— le fichier de sauvegarde contient plus ou moins de 240 lignes, différentes du fichier de session, sans les choix et temps ou avec des valeurs hors intervalle -> erreur.
Discussion des résultats : Le fichier de sauvegarde contient les mêmes 240 lignes que le fichier de session, avec des choix et temps compris dans le bon intervalle.
Résultat correct.

Test 8 : Nouveau contenu du fichier session en fin de session

Description et objectif : Vérifier le contenu du fichier *CSV* de session à

la fin d'une session, lorsqu'on lance une nouvelle session.
Données utilisées et résultats attendus :
— Données utilisées : fichier de session et celui aléatoire d'origine (id : 404).
— Résultat attendu : lorsque que l'on relance une session, le fichier de session doit contenir les prochaines images contenu dans le fichier aléatoire.
Scénario du test : Après avoir fini la première session avec le test précédent, on fait en sorte de relancer une session. On vérifie alors si le nouveau contenu du fichier de session est constitué des 243 nouvelles lignes suivantes, dans l'ordre du fichier aléatoire, et différentes de celles du fichier d'origine.
Analyse du test :
— le fichier de session contient les 243 nouvelles lignes correspondant au fichier aléatoire, et différentes de la session d'origine -> OK
— le fichier de session contient plus ou moins de 243 lignes, différentes du fichier aléatoire ou équivalentes au fichier de session d'origine -> erreur.
Discussion des résultats : Le fichier de session contient bien les 243 prochaines lignes du fichier aléatoire à évaluer, différentes du fichier de session d'origine.
Résultat correct.

Test 9 : Contenu fichier sauvegarde après déconnexion de session

Description et objectif : Vérifier le contenu du fichier *CSV* de sauvegarde lorsque qu'une session n'est pas terminée.
Données utilisées et résultats attendus :
— Données utilisées : fichiers de sauvegarde et de session d'origine (id : 409).
— Résultat attendu : le fichier de sauvegarde doit contenir seulement les lignes du fichier de session évaluées avec deux colonnes supplémentaires pour les choix et temps.
Scénario du test : On crée un nouveau compte utilisateur à partir duquel on lance une session. On quitte la session au bout de l'évaluation de 15 images. Dans ce cas, le fichier de sauvegarde doit contenir les 15 lignes correspondantes, sans celles de l'entraînement, que le fichier session, avec deux colonnes de choix et de temps.
Analyse du test :
— le fichier de sauvegarde contient les mêmes 15 lignes évaluées que le fichier de session avant évaluation, en plus des choix et temps -> OK
— le fichier de sauvegarde contient plus ou moins de 15 lignes, différentes du fichier de session ou avec les lignes d'entraînement, sans les choix et temps ou avec des valeurs hors intervalle -> erreur.
Discussion des résultats : Le fichier de sauvegarde contient les mêmes 15 lignes que le fichier de session, avec des choix et temps compris dans le bon intervalle.
Résultat correct.

Test 10 : Contenu fichier session après déconnexion session

Description et objectif : Vérifier le contenu du fichier *CSV* de session lorsque qu'une session n'est pas terminée.
Données utilisées et résultats attendus :
— Données utilisées : fichier de session d'origine et de session une fois la session quittée (id : 409).
— Résultat attendu : le fichier de session ne doit plus contenir les lignes des images évaluées.
Scénario du test : On reprend le même compte que précédemment, depuis lequel on a quitté la session. On vérifie alors si les lignes des images évaluées sont bien supprimées du fichier. Dans ce cas, les 15 premières lignes d'évaluation du fichier de session doivent être différentes du fichier de session d'origine.
Analyse du test :
— le fichier de session ne contient plus les lignes des images évaluées -> OK
— aucune ligne du fichier de session n'est supprimée -> erreur.
Discussion des résultats : Le fichier de session contient les lignes des images à évaluer, mais pas les lignes des images évaluées.
Résultat correct.

Test 11 : Contenu fichier session après déconnexion session

Description et objectif : Vérifier que le fichier *CSV* de session contient toujours les images d'entraînement après avoir quitté une session.
Données utilisées et résultats attendus :
— Données utilisées : fichier de session d'origine et de session une fois la session quittée (id : 409).
— Résultat attendu : le fichier de session doit toujours contenir les images d'entraînement.
Scénario du test : On reprend la même situation que le test précédent. Dans ce cas, le fichier de session est modifié après avoir quitté la session. On vérifie alors si le fichier de session contient toujours les lignes des images d'entraînement, puis les lignes des images non-évaluées et non celles déjà évaluées.
Analyse du test :
— le fichier de session contient les lignes des images d'entraînement puis les images pas encore évaluées, et plus les images évaluées -> OK
— les lignes des images d'entraînement sont supprimées du fichier, les lignes des images déjà évaluées sont conservées -> erreur.
Discussion des résultats : Le fichier de session contient les 3 lignes des images d'entraînement, puis les lignes des images pas encore évaluées, pour afficher lors de la prochaine session.
Résultat correct.

Test 12 : Suppression des fichiers aléatoire et de session

Description et objectif : vérifier que mix.csv et session.csv sont vides après avoir écrit dans le fichier save.csv.
Données utilisées et résultats attendus :
— Données utilisées : fichier mix.csv, session.csv et save.csv.
— Résultat attendu : Les fichiers mix.csv et session.csv doivent être supprimés.
Scénario du test : On crée mix.csv. On vérifie qu'on écrit les lignes de mix dans session. On écrit les lignes de session dans save avec deux colonnes. On vide les fichiers mix et session car ils ne servent plus à rien. On vérifie qu'ils sont vides et on les supprime.
Analyse du test :
— Si ils sont vides. -> OK
— Si ils ne sont pas vides -> erreur.
Discussion des résultats : Les fichiers mix et session.csv sont supprimés. Résultat correct.

6.1.2 Tests sur l'interface partie client

Dans le fichier **script.test.js**, les tests sont établis en lien avec le déroulement d'une session et certaines composantes d'affichage de l'interface. Dans cette partie, la plupart des tests se font à partir des différents choix possibles durant l'expérience. En effet, il y a 6 choix différents possibles :

- "Excellent"
- "Good"
- "Fair"
- "Poor"
- "Bad"
- "Null" : correspond au fait de ne pas faire de choix

Certains tests sont donc établis pour faire des essais avec chacun des choix, c'est pourquoi un même test peut être compter 6 fois dans le résultat final. De plus, on crée pour chaque cas des tableaux avec des noms d'images et classes aléatoires, que l'on transmet ensuite au script de la page html comme dans le code l'application.

Enfin, l'utilisation *Timer Mocks* [18] dans les tests permet d'éviter de fausser ces derniers avec les temps d'attente impliqués par la fonction de temps *setTimeout()*, utilisée à plusieurs endroits dans le code.

Test 13 : Choix durant la phase d'entraînement

Description et objectif : vérifier que les évaluations de l'utilisateur ne sont pas pris en compte durant la phase d'entraînement
Données utilisées et résultats attendus :
— Données utilisées : tableau des choix, choix, tableau des images
— Résultat attendu : le tableau des choix reste vide après avoir simulé un choix
Scénario du test : on affiche la première image de l'expérience à l'écran. Les images d'entraînement se situent aux indices 0, 1 et 2, donc on se positionne

successivement à ces indice là dans le tableau des images. Ces indices représente l'image actuellement affichée. On teste alors en simulant chacun des choix possibles en vérifiant que le tableau des choix reste vide.
Analyse du test :
— le tableau des choix reste vide après une évaluation quelconque sur les images d'entraînement -> OK
— le tableau contient le choix effectué après une évaluation sur l'une des images d'entraînement -> erreur.
Discussion des résultats : le tableau des choix reste vide après l'évaluation des images d'entraînement avec chacun des choix possibles.
Résultat correct.

Test 14 : Temps durant la phase d'entraînement

Description et objectif : vérifier que les temps t des évaluations de l'utilisateur ne sont pas pris en compte durant la phase d'entraînement
Données utilisées et résultats attendus :
— Données utilisées : tableau des temps, choix, temps t, tableau des images
— Résultat attendu : le tableau des temps reste vide après avoir simulé un choix
Scénario du test : à partir du test précédent, la première image de l'expérience est affichée à l'écran. Là aussi on se positionne aux indices 0, 1 et 2 dans le tableau des images. On teste alors en simulant chacun des choix possibles en vérifiant que le tableau des temps reste vide.
Analyse du test :
— le tableau des temps reste vide après une évaluation quelconque sur les images d'entraînement -> OK
— le tableau contient le temps t pris pour effectuer une évaluation sur l'une des images d'entraînement -> erreur.
Discussion des résultats : le tableau des temps reste vide après l'évaluation des images d'entraînement avec chacun des choix possibles.
Résultat correct.

Test 15 : Affichage des prochaines images

Description et objectif : vérifier que les image et carte d'explication suivantes sont affichées après un choix
Données utilisées et résultats attendus :
— Données utilisées : tableau des images, tableau des cartes d'explications, éléments de la page html
— Résultat attendu : les image et carte d'explication affichées sont bien les suivantes dans les tableaux
Scénario du test : On affiche les premières images sur l'interface. On récupère les images des *div* correspondantes à partir de la page html, on simule un choix, et enfin on vérifie si les nouvelles images sont les suivantes à l'indice $i + 1$ dans

les tableaux.
Analyse du test :
— les images suivantes affichées après un choix sont bien les suivantes dans les tableaux -> OK
— les images affichées n'ont pas changé où ne sont pas celles dans l'ordre des tableaux -> erreur.
Discussion des résultats : après chacun des choix, les images suivantes sont correctement modifiées à l'affichage sur la page html.
Résultat correct.

Test 16 : Affichage des prochaines classes

Description et objectif : vérifier que les vérité terrain et classification suivantes sont affichées après un choix
Données utilisées et résultats attendus :
— Données utilisées : tableau des vérités terrain, tableau des classifications, éléments de la page html
— Résultat attendu : les vérité terrain et classification affichées sont bien les suivantes dans les tableaux
Scénario du test : On affiche les premières classes sur l'interface. On récupère les textes des *div* correspondantes à partir de la page html, on simule un choix, et enfin on vérifie si les nouvelles classes sont les suivantes à l'indice $i + 1$ dans les tableaux.
Analyse du test :
— les classes suivantes affichées après un choix sont bien les suivantes dans les tableaux -> OK
— les classes affichées n'ont pas changé où ne sont pas celles dans l'ordre des tableaux -> erreur.
Discussion des résultats : après chacun des choix, les classes suivantes sont correctement modifiées à l'affichage sur la page html.
Résultat correct.

Test 17 : Affichage de l'arrière plan

Description et objectif : vérifier que l'arrière plan durant 5 secondes après un choix
Données utilisées et résultats attendus :
— Données utilisées : *div* de la page html, choix
— Résultat attendu : la réinitialisation visuelle doit s'afficher seulement après un choix, et durant 5 secondes
Scénario du test : On affiche des images et classes sur l'interface. On récupère l'état d'affichage de la *div* correspondante à l'affichage des images, classes et formulaire de la page html, on simule un choix, et enfin on vérifie si tous les éléments de cette *div* sont bien cachés, puis réapparaissent au bout de 5 secondes.

Analyse du test :
— les éléments de la page html sont dissimulés après un choix, durant 5 secondes, avant d'être affichés de nouveau -> OK
— les éléments de la page html ne sont pas dissimulés, durant moins de 5 secondes ou ne sont pas affichés de nouvau eu bout de ces 5 secondes -> erreur.
Discussion des résultats : après chacun des choix, les éléments de la page html sont bien cachés pendant 5 secondes, laissant place entièrement à l'arrière plan.
Résultat correct.

Test 18 : Sauvegarde du temps d'évaluation

Description et objectif : vérifier que le temps t d'évaluation pris par l'utilisateur est bien pris en compte après un choix durant la phase d'évaluation
Données utilisées et résultats attendus :
— Données utilisées : tableau des temps, temps t, choix
— Résultat attendu : le temps t pris pour faire un choix par l'utilisateur doit être stocké dans le tableau
Scénario du test : On affiche des images et classes sur l'interface. On simule l'avancer du temps, pour justement simuler le temps t de réflexion de l'utilisateur, puis on effectue chaque choix. On vérifie alors que le temps simulé t correspond bien au temps stocké dans le tableau des temps.
Analyse du test :
— le temps passé t pour évaluer est bien stocké dans le tableau -> OK
— le tableau reste vide, ou le temps passé t n'est pas le même que celui stocké dans le tableau -> erreur.
Discussion des résultats : En effectuant chaque type de choix à des temps t différents, les temps t sont bien stockés dans le tableau des temps.
Résultat correct.

Test 19 : Sauvegarde des choix d'évaluation

Description et objectif : vérifier que le choix d'une d'évaluation de l'utilisateur est bien pris en compte après un choix durant la phase d'évaluation
Données utilisées et résultats attendus :
— Données utilisées : tableau des choix, choix
— Résultat attendu : le choix de l'utilisateur doit être stocké dans le tableau
Scénario du test : On affiche des images et classes sur l'interface. On simule l'évaluation des images avec chaque choix possible, l'un après l'autre, puis on vérifie alors que la taille du tableau a augmenté de 1 pour chaque choix, et que le choix simulé correspond bien à la valeur stockée dans le tableau des choix. On vérifie également que chacun des choix est bien converti par une valeur de 0 à 5 : - "Excellent" -> 5 - "Good" -> 4
- "Fair" -> 3

- "Poor" -> 2
- "Bad" -> 1
- "Null" -> 0

Analyse du test :
— le choix fait par l'utilisateur correspond à la valeur stockée dans le tableau des choix -> OK
— le tableau reste vide, ou le choix de l'utilisateur n'est pas le même que celui stocké dans le tableau -> erreur.
Discussion des résultats : En effectuant chaque type de choix, ils sont correctement convertis et stockés dans le tableau.
Résultat correct.

Test 20 : Séparation phases d'entraînement et d'évaluation

Description et objectif : vérifier qu'une page de début d'évaluation est affichée entre les deux phases
Données utilisées et résultats attendus :
— Données utilisées : *div* de la page html
— Résultat attendu : la page de séparation des phases doit être affichée après l'évaluation de la troisième image d'entraînement (indice 2)
Scénario du test : On affiche une image sur l'interface et on se positionne à l'indice 2. On récupère les états des affichages des *div* de l'élément principal pour l'affichage de l'interface de base, et celle du texte et bouton pour l'interface de la transition des deux pages. On vérifie alors que les images et classes ne sont plus affichées, après la simulation d'un choix quelconque, et que le texte et le bouton *start* d'entre deux phases sont affichés.
Analyse du test :
— on simule un choix et s'affiche l'interface de transitions d'entre deux phases -> OK
— l'un des éléments de l'interface de transition ne s'affiche pas, ou les images et classes suivantes sont affichées -> erreur.
Discussion des résultats : En simulant un choix pour la troisième image, l'interface affiche un bouton *start* et un texte de début d'évaluation.
Résultat correct.

Test 21 : Affichage de l'interface

Description et objectif : vérifier que les images, classes et formulaire de boutons sont bien affichés au lancement d'une session
Données utilisées et résultats attendus :
— Données utilisées : *div* de la page html
— Résultat attendu : l'élément principal de la page html est bien affichée
Scénario du test : On affiche une image sur l'interface. On vérifie si le statut d'affichage du contenant des images, classes et formulaire est bien correct, c'est

à dire que ces éléments sont affichés.
Analyse du test :
— les éléments de l'interface principal sont affichés -> OK
— le contenu des éléments n'est pas affiché sur l'interface -> erreur.
Discussion des résultats : lorsque que l'on lance la session, les éléments de l'interface sont tous affichés.
Résultat correct.

Test 22 : Temps choix utilisateur

Description et objectif : vérifier que l'utilisateur à exactement 20 secondes maximum pour effectuer un choix
Données utilisées et résultats attendus :
— Données utilisées : *div* de la page html
— Résultat attendu : après 20 secondes écoulées, les images et classes ne sont plus affichées sur l'écran
Scénario du test : On affiche une image sur l'interface. On vérifie que les éléments de l'interface sont affichés, on simule l'avancée du temps sur 20 secondes puis on vérifie que les éléments sont cachés.
Analyse du test :
— les éléments de l'interface sont affichées puis cachés au bout de 20 secondes -> OK
— les éléments sont cachés au bout de 20 de 20 secondes -> erreur.
Discussion des résultats : les éléments de l'interface sont bien dissimulés au bout de 20 secondes pour laisser place à la réinitialisation visuelle avant la prochaine image.
Résultat correct.

Test 23 : Affichage bouton de déconnexion

Description et objectif : vérifier que le bouton de déconnexion est affiché uniquement durant l'affichage des images
Données utilisées et résultats attendus :
— Données utilisées : *div* de la page html
— Résultat attendu : le bouton de déconnexion est affichée durant une évaluation et dissimulé durant la réinitialisation visuelle
Scénario du test : On affiche une image sur l'interface et on vérifie si le bouton de déconnexion est affichée. On simule l'avancée du temps de 20 secondes, on vérifie alors si le bouton de déconnexion est dissimulé, puis afficher de nouveau au bout de 5 secondes.
Analyse du test :
— le bouton de déconnexion est affiché durant l'évaluation puis dissimulé durant la réinitialisation visuelle -> OK
— le bouton de déconnexion est affiché durant la réinitialisation visuelle ou dis-

simulé lors de l'évaluation -> erreur.
Discussion des résultats : le bouton de déconnexion est correctement affiché en bas à droite de l'écran durant l'évaluation, puis dissimulé lors de la réinitialisation.
Résultat correct.

6.1.3 Tests sur la base de données

Dans le fichier **database.test.js**, les tests sont écrits en lien avec la base de données. Un utilisateur est créé et lui est assigné un id, une adresse mail, un mot de passe, un âge et la date de création du compte. Avant de commencer les tests, une connexion à la base de données est simulé en utilisant des données telles que le nom du serveur, le port, le mot de passe et le nom de la base de données. Une vérification de la connexion est ensuite primordiale.

Test 24 : Ajout utilisateur à la base de données

Description et objectif : vérifier que l'utilisateur est bien ajouté.
Données utilisées et résultats attendus :
— Données utilisées : données utilisateur , langage SQL
— Résultat attendu : L'utilisateur a été inséré dans la table "utilisateurs".
Scénario du test : On simule l'avancée du temps. On insère l'utilisateur à l'aide de INSERT INTO. On vérifie si l'utilisateur a été inséré correctement en sélectionnant la colonne email. On vérifie alors si à la ligne 0 il y a bien la colonne email, mdp et âge et si elles correspondent bien à l'utilisateur.
Analyse du test :
— les colonnes correspondent bien aux données de l'utilisateur -> OK
— les colonnes sont vides ou les colonnes ne correspondent pas aux données de l'utilisateur -> erreur.
Discussion des résultats : L'utilisateur a été inséré correctement.
Résultat correct.

Test 25 : Suppression utilisateur de la base de données

Description et objectif : vérifier que l'utilisateur est bien supprimé.
Données utilisées et résultats attendus :
— Données utilisées : données utilisateur , langage SQL
— Résultat attendu : L'utilisateur a été supprimer de la table "utilisateurs".
Scénario du test : On retrouve tout d'abord l'utilisateur grâce à son email. (Cela peut se faire aussi avec son mot de passe ou son identifiant). On supprime ensuite l'utilisateur avec la syntaxe DELETE FROM ... WHERE.
Analyse du test :
— Le mail du user a été trouvé, on supprime donc la ligne qui correspond au mail. -> OK

— L'utilisateur n'a pas été trouvé, le mail n'a pas été trouvé ->erreur.
Discussion des résultats : L'utilisateur a été supprimé correctement.
Résultat correct.

Test 26 : Mise à jour informations utilisateur dans la base de données

Description et objectif : vérifier que le mail et le mot de passe ont été modifié et remplacer par l'id 6 mois après leur création.
Données utilisées et résultats attendus :
— Données utilisées : données utilisateur , langage SQL
— Résultat attendu : Email et mot de passe remplacer par l'id de l'utilisateur.
Scénario du test : On prend la colonne email et mot de passe. Si l'email et le mot de passe ont été crée il y a plus de 6 mois, on les remplace par l'id. On utilise pour cela la syntaxe UPDATE... SET...
Analyse du test :
— La colonne mail du user a été trouvé, on supprime l'email on le remplace par l'id 6 mois après sa création avec la syntaxe DATE_SUB(NOW(), INTERVAL 6 MONTH). (resp. mdp). -> OK
— L'utilisateur n'a pas été mis à jour ->erreur.
Discussion des résultats : L'utilisateur a été mis à jour correctement.
Résultat correct.

6.1.4 Tests sur la partie administrateur

Description et objectif : Vérifiez que vous êtes en mesure d'effectuer les opérations suivantes : trouver l'utilisateur et afficher ses informations, enregistrer la base de données dans un fichier csv, enregistrer les résultats de l'utilisateur trouvé dans un fichier csv, enregistrer les résultats associés à l'image sup particulière dans un fichier csv, effacer les informations personnelles de la base de données, supprimer l'utilisateur trouvé de la base de données.
Données utilisées et résultats attendus :
— Données utilisées : données utilisateur, langage SQL
— Résultat attendu : le cycle complet de l'action de l'administrateur.
Scénario du test : Au début, une table est ajoutée à la base de données. Ensuite, un utilisateur est ajouté à cette table. Il s'ensuit un cycle complet d'actions de l'administrateur, puis la suppression de la table. Les tests doivent être exécutés séquentiellement, sauf pour ajax.
Analyse du test :
— Un cycle complet d'actions sur la page d'administration a été effectué -> OK
— Une erreur s'est produite à un moment donné -> erreur.
Discussion des résultats : Le cycle complet s'est déroulé correctement.
Résultat correct.

6.1.5 Résultats des tests

```
Test Suites: 4 passed, 4 total
Tests:       73 passed, 73 total
Snapshots:   0 total
Time:        8.189 s
Ran all test suites in 2 projects.
PS C:\xampp\htdocs\pfe\tests>
```

FIGURE 30 – Résultats des tests

Commentaires particuliers : Comme nous pouvons le voir sur la Figure 30 ci-dessus, tous les tests implémentés et exprimés dans les différentes sections au-dessus sont passés avec succès. La seule remarque à cela est le fait qu'il y ait un message affiché lors du lancement des tests comme quoi les *console.log* que nous utilisons dans le code de l'application ne peuvent pas être affichés après la fin des tests.

6.2 Tests temps d'exécution et de portabilité

Pour tester notre application, il a également été question de sa portabilité. Dans ce cas, l'application est fonctionnelle sur les navigateurs suivants :

— Google Chrome (111.0.5563.64 (64 bits))
— Mozilla Firefox (110.0.1 (64 bits))

Pour observer l'efficacité de l'application, nous avons effectué différents tests concernant les temps d'exécution des fonctionnalités principales. Ces temps d'exécution sont calculés en effectuant les actions directement depuis le code de l'application en faisant une moyenne de 5 tentatives.

6.2.1 Fonctionnalités partie client

Le tableau ci-dessous 3 présente les temps d'exécution pour certaines fonctionnalités principales de la partie client.

Fonctionnalités de l'application	Temps d'exécution
Création de compte : création des 3 fichiers *CSV*, dont le Scénario de l'expérience	$0,1162$s
Connexion à l'expérience	$0,0203$s
Création scénario pour une session ou relancer un session	$0,0034$s
Stockage du choix et temps t puis chargement des prochaines images et classes	0.0038s

TABLE 3 – Tableau des temps d'exécution des fonctionnalités principales de l'expérience

6.2.2 Fonctionnalités partie administrateur

Dans cette partie, il s'agit des fonctionnalités liées à la partie administrateur avec les temps d'exécution présentés dans le tableau ci-dessous 4.

Fonctionnalités de l'application	Temps d'exécution
Chercher un utilisateur, parmi 20 comptes, dans la base de données pour afficher ses informations	0.0014s
Charger les résultats d'un utilisateur dans un fichier *CSV* (une session entière dans ce cas)	0.0016s
Charger les résultats pour une image dans un fichier *CSV*	0.0016
Charger la base de données dans un fichier *CSV* (20 utilisateurs dans ce cas)	$0,0014$s
Suppression d'un utilisateur et de ses fichiers de la base de données	$2,0566$s

TABLE 4 – Tableau des temps d'exécution des fonctionnalités principales de la partie administrateur

7 Conclusion

En conclusion, la mesure des observations oculaires est un outil crucial utilisé dans divers contextes comme le traitement d'images. Dans l'ensemble, la méthodologie développée dans cette étude fournit un cadre précieux pour l'évaluation de la qualité des outils de traitement et de visualisation d'images.

La conception de notre projet repose sur la mise en place d'une application d'évaluation des cartes d'explication sur des images par des sujets humains. Ces cartes d'explications sont le résultat d'informations sur le processus de décision d'un classifieur CNN 2.2 pour une entrée donnée. Les images corrompues, qui servent de données d'entrées, sont construites à partir de différents types de dégradations : le bruit gaussien additif, le flou gaussien, la distorsion de luminosité uniforme et la distorsion de perspective 2.3.3. Toutes ces cartes d'explications ont été générées en préambule de notre projet d'application. L'utilisateur prend alors part à l'expérience, afin d'évaluer la qualité des cartes d'explication, en attribuant un score à chacune d'entre elles qui lui sont présentées.

Par rapport à la demande du client, chacune des requêtes indiquées ont été implémentées. Pour mettre en place cette application, il a fallu appliquer certaines spécifications précises concernant l'expérience en elle-même, son déroulement, ainsi que sa présentation visuelle 3. En effet, l'interface de l'application se devait d'être la plus minimaliste et ergonomique possible, dans le but d'éviter toute *fatigue visuelle* auprès de l'utilisateur.

En ce qui concerne les fonctionnalités de l'application, l'utilisateur peut créer un compte, en acceptant les conditions d'utilisation, pour ensuite se connecter

à l'expérience. Côté backend, les informations liées à l'utilisateur sont stockées dans une base de données MySQL, tandis que les scénarios de chacun des utilisateurs sont générés à partir des fichiers *CSV*. Par la suite, l'utilisateur peut prendre part à l'expérience grâce à l'interface mise en place composée d'une image corrompue, une carte d'explication, la vérité terrain et une classification. L'utilisateur peut évaluer chacune des cartes d'explication affichées, à partir de l'échelle de notation de boutons radio établie selon l'échelle de Likert 2.3.2, et séparées par une réinitialisation visuelle. Les spécifications de l'interface et les temps de ces étapes sont inspirés et déterminés selon la norme IUT-R BT.500-14 2.3.1. L'interface se veut, comme souhaitée, la plus ergonomique et simplifiée possible pour éviter toute distraction visuelle, afin que le sujet humain puisse pleinement se focaliser sur l'expérience. À la fin d'une session, l'utilisateur peut se déconnecter, tout comme en cours de session, ou relancer une nouvelle session.

Par rapport à l'implémentation actuelle de l'application, certaines améliorations pourraient être envisagées. Tout d'abord, les textes de description de l'expérience, ainsi que les conditions générales, sont rédigés en français au sein de l'outil. Une première amélioration serait alors de permettre à l'utilisateur de choisir la langue utilisée pour mieux comprendre et prendre part à l'expérience. Ensuite, d'un point de vue scénario d'expérience, certaines améliorations pourraient être apportées. Actuellement, le fichier de scénario de l'expérience contient les 1800 images à évaluer dans un ordre aléatoire, incluant également les 24 images d'entraînement, mais contient parfois quelques images du même type à la suite. En effet, une même image peut subir plusieurs modifications possibles. La génération du fichier pourrait alors prendre en compte le fait de ne pas présenter la même image, ou la même dégradation, plusieurs fois de suite au sein de l'expérience.

Références

[1] Matei Mancas and Olivier Le Meur. Applications of saliency models. *From human attention to computational attention : A multidisciplinary approach*, pages 331–377, 2016.

[2] P. Le Callet, C. Viard-Gaudin, and D. Barba. A convolutional neural network approach for objective video quality assessment. *IEEE Transactions on Neural Networks*, 17(5) :1316–1327, 2006.

[3] Souad Chaabouni, Jenny Benois-Pineau, and Chokri Ben Amar. Chabonet : Design of a deep cnn for prediction of visual saliency in natural video. *Journal of Visual Communication and Image Representation*, 60 :79–93, 2019.

[4] Granhof Alexander and Eriksson Jakob. Improving the user experience in data visualization web applications, 2021.

[5] Meghna P. Ayyar, Jenny Benois-Pineau, and Akka Zemmari. Review of white box methods for explanations of convolutional neural networks in image classification tasks. *J. Electronic Imaging*, 30(5), 2021.

[6] Ramprasaath R. Selvaraju, Abhishek Das, Ramakrishna Vedantam, Michael Cogswell, Devi Parikh, and Dhruv Batra. Grad-cam : Why did you say that ? visual explanations from deep networks via gradient-based localization. *CoRR*, abs/1610.02391, 2016.

[7] David Alvarez-Melis and Tommi S. Jaakkola. Towards robust interpretability with self-explaining neural networks. *CoRR*, abs/1806.07538, 2018.

[8] Tristan Gomez, Thomas Fréour, and Harold Mouchère. Metrics for saliency map evaluation of deep learning explanation methods. In *ICPRAI (1)*, volume 13363 of *Lecture Notes in Computer Science*, pages 84–95. Springer, 2022.

[9] Vitali Petsiuk, Abir Das, and Kate Saenko. RISE : randomized input sampling for explanation of black-box models. In *BMVC*, page 151. BMVA Press, 2018.

[10] Luca Bourroux, Jenny Benois-Pineau, Romain Bourqui, and Romain Giot. Multi layered feature explanation method for convolutional neural networks. In *ICPRAI (1)*, volume 13363 of *Lecture Notes in Computer Science*, pages 603–614. Springer, 2022.

[11] Abraham Montoya Obeso, Jenny Benois-Pineau, Mireya Saraí García-Vázquez, and Alejandro Alvaro Ramírez-Acosta. Visual vs internal attention mechanisms in deep neural networks for image classification and object detection. *Pattern Recognit.*, 123 :108411, 2022.

[12] Union internationale des télécommunications. Méthodologies d'évaluation subjective de la qualité des images de télévision. *UIT*, pages 34 – 37, 10 2019. Accès le 07/02/2023.

[13] Ankur Joshi, Saket Kale, Satish Chandel, and D Kumar Pal. Likert scale : Explored and explained. *British journal of applied science & technology*, 7(4) :396, 2015.

[14] A. Zhukov, J. Benois-Pineau, and R. Giot. Evaluation of explanation methods of ai – cnns in image classification tasks with reference-based and no-reference metrics. *Advances in Artificial Intelligence and Machine Learning*, 3(1) :620–646, 2023.

[15] Richard Szeliski. *Computer Vision Algorithms and Applications. Second Edition*. Springer Cham, New York, 2022.

[16] Kazi Ahmed Asif Fuad, Pierre-Etienne Martin, Romain Giot, Romain Bourqui, Jenny Benois-Pineau, and Akka Zemmari. Features Understanding in 3D CNNs for Actions Recognition in Video. In *Tenth International Conference on Image Processing Theory, Tools and Applications, IPTA 2020*, Paris, France, October 2020.

[17] Christine Titine. Code couleurs html - Les gris. Accès le 07/02/2023.

[18] Timer Mocks · Jest, 3 2023.