Composantes du Rapport de Projet

1. Définition de la Tâche

Description de la tâche: Expliquez ce que fait votre système (entrée et sortie).

Le système qui sera développé devra prendre en entrée une police de caractère ainsi qu'une image textuel ayant cette même police de caractère. Après traitement, Elle retournera une chaîne de caractère correspondant au texte donner en entrée sous forme imagée. Le modèle doit constamment apprendre de nouvelles polices de caractère. Elle doit s'adapter aux nouvelles polices de caractères utilisées et nommer par l'utilisateur.

Pertinence IA: Justifiez l'utilisation de l'intelligence artificielle pour cette tâche.

Dans les années précédentes, un programme algorithmique était utilisé pour reconnaître les écritures et les différentes polices de caractère. Cependant, ces programmes manquaient de fiabilité. il était parfois nécessaire d'écrire un nouveau programme et un nouvel algorithme pour des police de caractère plus récente. ces outils informatiques manquait d'adaptabilité. L'utilisation de l'intelligence artificielle basée sur un modèle s'avère une solution efficace, puisqu'elle permet une fonctionnalité d'apprentissage qui lui permet d'intégrer des nouvelles polices de caractère. Ainsi les modèles peuvent être ajusté lorsque de nouvelles police de caractère font leur apparition.

2. Brève Revue de la Littérature

- **OCRopus** : Un projet modulaire également basé sur LSTM. Bien qu’il soit moins maintenu aujourd’hui, il a inspiré les versions récentes de Tesseract. Il est utile pour ceux qui veulent concevoir des pipelines OCR personnalisés, mais demande plus de configuration.
- **Kraken** : Basé sur OCRopus, Kraken est orienté vers la reconnaissance de documents historiques ou manuscrits. Il prend en charge des scripts complexes et offre une interface d'entraînement conviviale. C’est un bon choix pour les projets nécessitant la reconnaissance de textes anciens ou dans des alphabets non latins.
- **EasyOCR** : Basé sur PyTorch, ce moteur se distingue par sa simplicité d’utilisation et sa prise en charge de plus de 80 langues. Il est moins flexible que Tesseract pour l'entraînement personnalisé, mais il offre de bons résultats rapidement sur des textes imprimés standards.
- **Calamari OCR** : Un moteur OCR moderne utilisant TensorFlow et compatible avec les fichiers d'OCRopus. Il permet l'entraînement multi-polices et multi-langues et propose

des outils puissants de fusion de modèles (voting). Il est particulièrement adapté aux scénarios où plusieurs modèles OCR doivent être combinés.

Dans la littérature, nous pouvons citer deux. Implémentations du OCR :

- La première consistait à simplifier la documentation des transactions dans le commerce électronique . Le OCR servait à numériser les transactions sous format papier. Ainsi, il était plus facile de parcourir l'historique des transactions. Le modèle atteignait un score F1 de 0,7703 lors de ses tests
- Le deuxième utilisait le OCR afin de lire des caractères sur des interfaces graphiques comme des boutons, des textes et des champs de texte. Ce modèle utilise un réseau de neurones dans sa méthode d'apprentissage. Il a été implémenter sur deux architectures. Soit l'architecture en cascade et l'architecture "bout-en-bout". Ce modèle a eu un score F1 de 94%

3. Matériel et Méthodes

Infrastructure

Lors de ce projet, nous utiliserons le langage de programmation Python afin d'entraîner le modèle pour chaque police de caractère entrée par l'utilisateur. Ce langage de programmation est très utilisé pour ce qui est de l'intelligence artificielle de l'analyse de données et du machine Learning. Sa syntaxe est aussi très simpliste et très simple d'utilisation.

Nous utiliserons aussi un modèle de base conçu pour du OCR. Le modèle Tesseract est un modèle d'OCR qui est facile à agrémenter grâce aux nombreux outils disponible comme cntraining, combine_tessdata, lstmtraining, text2image, etc. Il est ainsi possible d'apprendre au modèle à reconnaître plus efficacement et avec plus de précision certaines polices d'écriture.

De plus, nous avons également utilisé la librairie TESSTRAIN qui implémentes les différents outils d'entrainement du modèle Tesseract en un outil simple qui ne demande que des images et des fichiers avec le texte sur l'image.

Méthode

Afin de parvenir à l'objectif mentionné plutôt, nous utiliserons les entrées de l'utilisateur afin de générer aléatoirement du contenu textuel imagée et structuré dans la police de caractère entrée par l'utilisateur. Les chaînes de caractère utilisées pour générer ces images seront aussi conservés. Pour les images textuelles, il sera utilisé plusieurs types de polices de caractères. Cela a pour objectif de donner une base plus solide sur différentes polices de caractères au modèle. Ensuite, le modèle sera entraîné sur les images de manière supervisé. Les couples (images, chaînes de caractère) formeront dataset utiliser pour

entraîner le modèle. Ensuite, le modèle sera testé sur le texte écris manuellement par l'utilisateur.

De cette manière, nous assurons que le modèle s'entraîne de manière supervisé en requérant un minimum d'interaction avec l'utilisateur.

Il est important de générer des chaînes de caractère qui forment des phrases dans une certaine langue. Les programmes aussi sont normalement spécialisés dans une seule langue à la fois. C'est pour cela que chaque police de caractère sous forme de chaîne textuelle et d'image ont été écrit en anglais par souci de généralité. Ce sont toujours les mêmes chaînes de caractère qui sont transformées pour chaque police de caractère. il est également très important que le contenu de ces chaines de caractères représente fidèlement la réalité au niveau de plusieurs critères primordials. il est tout d'abord important que le texte soit une série de mots structuré en phrases, que toutes les lettres soit présentes ainsi que les différents signe de ponction et signe spéciaux.

Des fichiers "ttf" permettent aussi de faire référence à différentes polices de caractères, afin de créer les images sous forme textuelles. Ce sont ces fichiers qui définissent les polices de caractère

Après la création de chaîne, de caractère sous forme d'image et sous forme textuelle, certains fichiers de référence pour la création du modèle ont été créé. notamment des fichiers de mapping, afin d'indiquer au modèle l'emplacement de chaque caractère.

Pour entrainer un modèle sur une police précise nous allons tout d'abord créer un script python utilisant l'outil Text2image de Tesseract afin de créer des images ainsi que des textes pour l'entrainement. Ce script prendra de nombreux paramètres en compte comme le nombre d'image, les phrases utilisées, la police, la taille de la police dans les image et la résolution. Ce script coupler à la libraire TESSTRAIN permet de créer notre modèle basé sur le modèle anglais de Tesseract entrainé additionnellement pour être plus efficace et précis sur des polices plus complexes.

Au niveau des données d'entrainement, nous avons simplement demander à ChatGPT de nous générer 50 phases en anglais, nous avons ensuite pris le soin d'ajouter des phrases contenant simplement toutes les lettres de l'alphabet en minuscule et majuscule ainsi que les signes de ponctions afin d'éviter que certains caractères peut communs soient oubliées.

Exemple de Données d'entrainements

Pour tester notre programme, nous avons choisi 5 police qui sont illustré ci dessous :

Vladimir Script

Stencil

THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG.

Juice ITC

The quick brown fox jumps over the lazy dog.

Brush Script MT Italic

The quick brown fox jumps over the lazy dog.

Bradley Hand ITC

The quick brown fox jumps over the lazy dog.

évaluation

Nous avions initialement envisagé d'évaluer les performances de notre modèle à l'aide du F1-score, un indicateur couramment utilisé en classification. Cependant, nous nous sommes rapidement rendu compte que ce choix n'était pas le plus adapté au contexte spécifique de la reconnaissance de texte. En effet, le F1-score repose sur une comparaison binaire entre des classes prédéfinies, ce qui ne permet pas de capturer avec précision les erreurs au niveau des caractères ou des mots. Par exemple, si le modèle prédit **"bongour"** au lieu de **"bonjour"**, il s'agit d'une erreur mineure en OCR, le mot reste lisible et très proche du mot attendu, mais cela serait considéré comme entièrement faux avec un F1-score, ce qui ne reflète pas fidèlement la qualité de la prédiction.

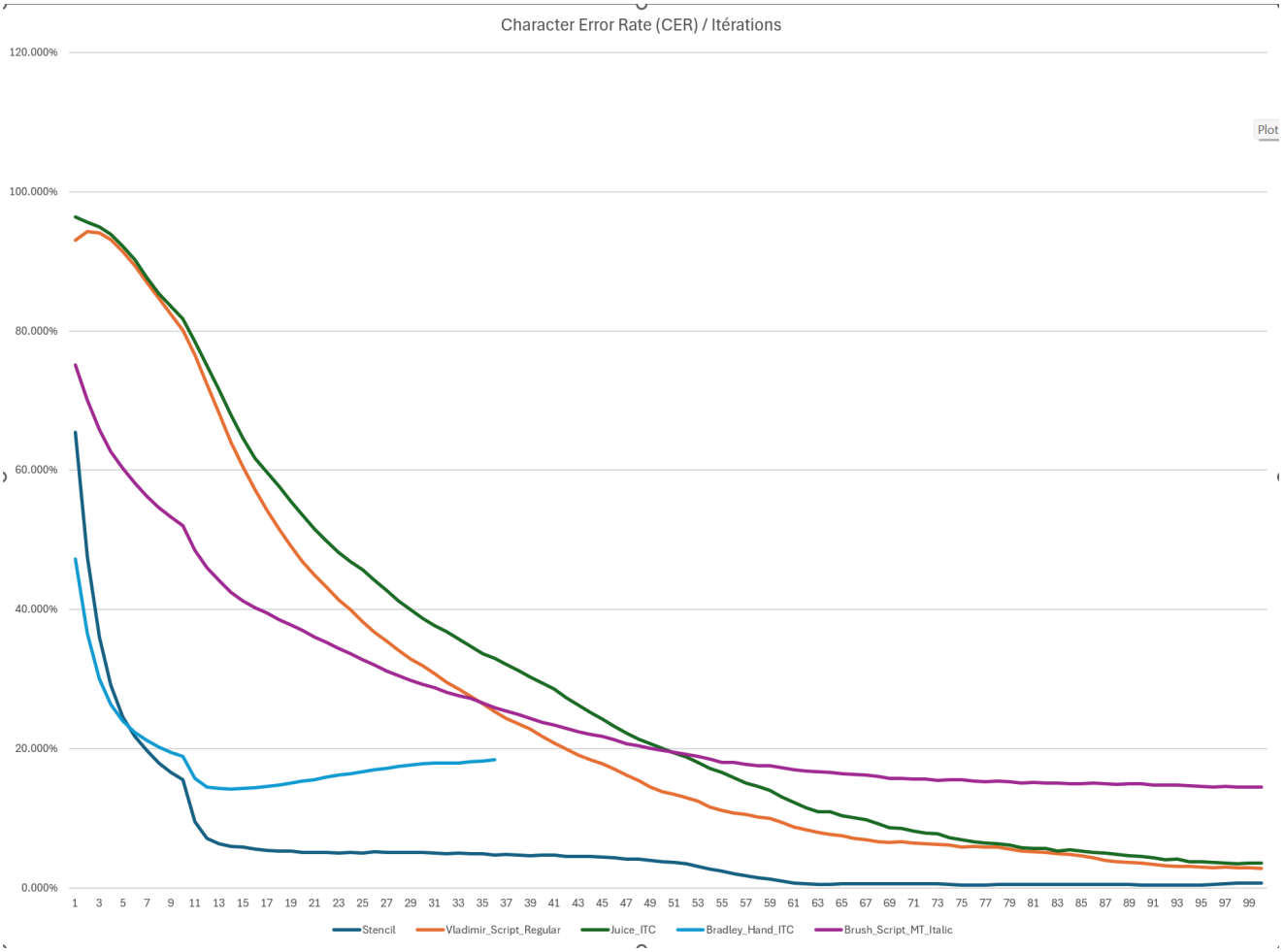
Pour cette raison, nous avons opté pour deux métriques beaucoup plus pertinentes dans le domaine de l'OCR : le **CER (Character Error Rate)** et le **WER (Word Error Rate)**. Ces mesures sont issues du domaine de la reconnaissance vocale et sont désormais largement utilisées pour évaluer la précision des systèmes de transcription automatique. Le **CER** mesure le taux d'erreur au niveau des **caractères** en calculant la distance de Levenshtein entre le texte prédit et le texte de référence, c'est-à-dire le nombre minimum d'opérations (insertions, suppressions ou substitutions) nécessaires pour transformer l'un en l'autre, divisé par le nombre total de caractères attendus. De même, le **WER** applique ce même principe, mais au niveau des **mots**. Ces métriques permettent donc de quantifier précisément à quel point une prédiction est proche du texte correct, même en cas d'erreurs

partielles, et donnent une évaluation beaucoup plus nuancée et représentative des performances réelles d’un modèle OCR.

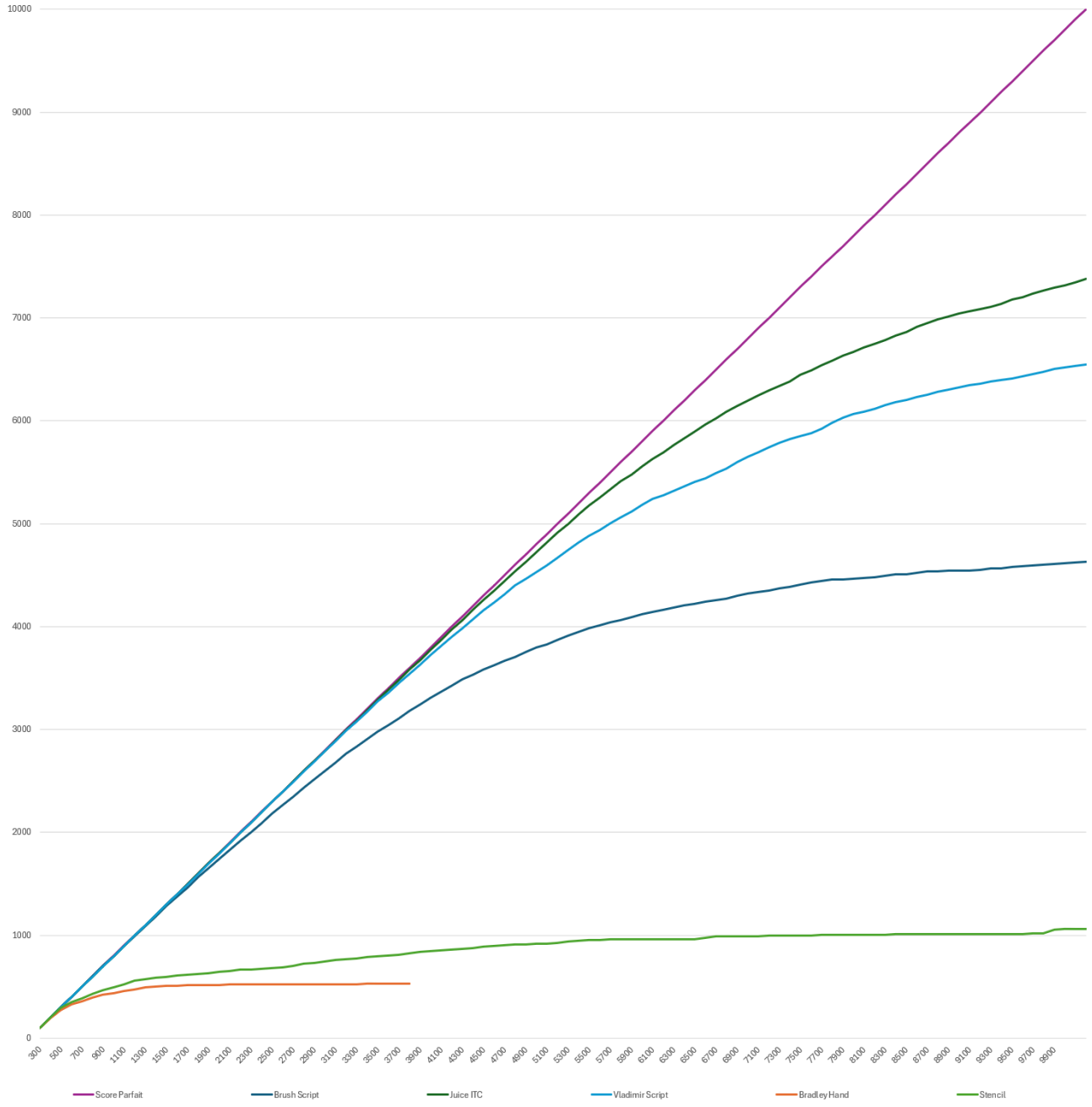
4. Résultats

Résultats de l'entrainement

Ce graphique présente le CER en fonction du nombre itération de l'entraînement du modèle tout les modèles ont été entrainé avec un maximum de 10.000 itérations afin d'éviter du surentrainement.



Ce graphique présente l'évolution des itérations retenues. Plus la valeur s'éloigne de la courbe du score parfait, moin il y a eu d'itération utile à l'entrainement du modèle.



Au niveau des performance finale, on peut observer que les police Brush_script et Bradley ont obtenu un score CER assez élevé de plus de 15 %

On peut voir ici qu'il y a un coude qui se forme pour chacune des polices de caractère. Ainsi presque tous les modèles ont un niveau de précision CER acceptable. Seuls les modèles Bradley_Hand_ITC et Stencil ne semble pas avoir un coude acceptable.

La raison pour laquelle certaines polices d'écriture n'ont pas un coude acceptable est simplement le fait que ce sont des polices d'écriture qui sont très peu utilisées et moins lisible. Les caractères ont notamment des formes assez inhabituelle. Ce sont des police de caractère qui se rapproche beaucoup plus des lettres attachées que de véritables caractères écrit par un ordinateur.

Cependant puisque les autres polices de caractère sont des polices qui sont beaucoup plus semblables à des polices de caractère écrites par un ordinateur et que les caractères sont

bien définis et séparés les uns des autres, le modèle a tendance à beaucoup mieux les reconnaître.

L'objectif est en quelque sorte accomplie, puisque le modèle sera utilisé pour reconnaître des polices de caractères écrites à l'ordinateur et non de façon manuscrite par un être humain.

L'utilisation de ce modèle aussi pourra être utilisé dans l'industrie si plus développé.

Pour plus détail sur les données, consulté le document excel ou bien les logs produits par l'entrainement sur le repositoire du projet sous : TESSTRAIN_SOLUTION\logs\ {modèle_training.log}

Résultats des tests

{
Stencil
text: HELLO, WORLD!

Vladimir_Script_Regular
text: Jus)39/223 74

Brush_Script_MT_Italic
text: ello, Weordd!

Bradley_Hand_ITC
text: Hellp, Worlo !

Juice_ITC
text: Fello or!
}

Ces résultats sont très révélateurs et confirme certain doutes apparu lors de la créations des graphiques. En effet, on peut voir très clairement avec le résultat du test avec la police "Vladimir Script Regular" que l'on a affaire a un cas typique de surrentrainement ou l'entrainement semble afficher des résultats encourageant, mais c'est en réalité car les images d'entrainement sont apprises par coeur par le modèle.

Outre ce modèle raté, on peut se tourner vers la police "Brush Script MT Italic" qui malgré ses similaritées avec la police "Vladimir Script Regular" n'est pas tomber dans le piège du surentrainement et affiche des résultats tout a fait convenable et réalistique pour une police aussi complexe.

Puis, on peut notée que le modèle "Juice_ITC" n'a pas afficher des résulats à la hauteur de ce qui est afficher dans les résultats d'entrainement.

5. Conclusion

résumé des résultats

Pour résumer les résultats, nous simplifierons en disant que nous avons accompli notre objectif qui était de concevoir un système OCR. Notre objectif était de créer un modèle ancien capable de reconnaître de l'écriture computationnelle sous forme d'image et de la convertir sous forme de chaîne de caractère. Cet objectif a été accompli avec une grande robustesse. Les seules polices de caractère qui n'ont pas été reconnues par notre modèle sont des polices de caractère qui sont moins encadré et moins computationnelle. Ce sont des palettes de Cara qui sont plus typiques d'une écriture manuscrites. Des écritures qui ne peuvent pas être bien cadré.

Perspectives et amélioration

Plusieurs mesures peuvent être employés pour améliorer l'exactitude de notre modèle.

Les données d'entrainement créeent étaient très basique et sans grande variété. Ce manque de variété rend nos modèle moin robuste et versatile. Plusieurs modifications peuvent être fait afin d'amélioré la variété des images d'entrainements.

Il pourrait s'avérer nécessaire d'entraîner le modèle sur divers arrière-plan. Dans le cas de ce projet, les images textuelles avec toutes et chacune un arrière-plan complètement blanc. Certaines images textuelles pourrais avoir un arrière-plan d'une couleur différente. Il serait donc pertinent d'entraîner le modèle sur des arrière-plan diversifiées.

Dans les données d'entraînement, l'ensemble des contenus textuel avait une taille de police identique. Dans la réalité d'un utilisateur du modèle, il se pourrait que son contenu textuel imagée soit de tête de police différente. Nous pouvons citer des exemples comme les citations, les notes dans la marge ainsi que les titres dans un texte. Entraîner le modèle sous différentes taille de police serait pertinent.

Un autre élément important à considéré est que de nombreuses police on de nombreuses variation de base comme : BOLD, ITALIC ou souligner qu'il pourrait également être pertinent d'inclure dans les images.

Si le modèle est voué a être utiliser dans le domaine financier, il pourrait également important de créer des images avec des tableau ou d'autre structure de données.

Finalement, il pourrait être pertinent de considéré toute modification qui pourrait arriver dans le contexte des images à transformer, par exemple si le but est d'analyser des documents numérisé, il pourrait être pertinent d'ajouter de nombreuse variation comme de la rotation, des taches, des modificaion de résolution.

Encore une fois, dans les données d'entraînement, l'ensemble des caractères proviennent de ceux qui sont permis dans la langue anglaise. cependant l'ensemble des utilisateurs de

ce genre de modèle n'ont pas l'anglais comme langue maternelle. nous pouvons citer les langues asiatiques, telles que le japonais ou le mandarin qui possède leur propre ensemble de caractères. nous pouvons énumérer le nombre de caractères japonais au-dessus de 3000, alors que la langue anglaise ne possède que les lettres de l'alphabet. il serait donc pertinent d'entraîner le modèle sur l'ensemble des familles de caractère qui existe dans de multiples cultures et de multiples région du monde.

Outre les caractères, il y a aussi l'ensemble des signes de ponctuation et des caractères spéciaux. La plupart du contenu textuel qui sera analysé par le modèle qui sera utilisé par l'utilisateur oran, une syntaxe nécessitant des signes de ponctuation et des caractères spéciaux. Au sein de nos données d'entraînement, il a été omis, dans son intégralité, l'existence de ces caractères. Pour ces raisons, il serait pertinent d'intégrer ses caractères spéciaux et ses signes de ponctuation à l'intérieur de nos données d'entraînement si le projet était à recommencer ou à poursuivre.