Stage Data Scientist - Vérification de la qualité des interventions

Détection de pose de PTO - Comparaison des photos avant et après intervention

Rédigé par Idir Sadaoui,

Responsable UFR : Anne Sabourin
Responsable de stage : Farid Zamoum (farid.zamoum@orange.com)
Tutrice de stage : Marie-Angèle Faye (marie-angele.faye@orange.com)

Stage effectué chez Orange, 2 avenue de la Gare, Montigny-le-bretonneux

Année universitaire 2022-2023

Contents

1 Présentation de l'entreprise **2**
1.1 L'entreprise Orange . . . 2
1.2 La DTSI et l'équipe VQI . . . 2

2 Résumé du cas d'usage **3**
2.1 Contexte et définition du besoin métier . . . 3
2.2 Cas d'usage et solution actuelle . . . 3
2.3 Axe d'amélioration . . . 4

3 Google Cloud Platform **5**
3.1 Services utilisés . . . 5
3.2 Vertex AI . . . 5

4 Approches **7**
4.1 Le jeu de données . . . 7
4.2 Labellisation . . . 7
4.2.1 Label Studio . . . 7
4.2.2 Découpage du jeu de données . . . 8
4.3 Notebooks et IDE . . . 10
4.4 Image Matching . . . 11
4.5 Image Similarity . . . 12
4.5.1 Libraries et modèles . . . 12
4.5.2 CNN et Image Embedding . . . 12
4.6 AutoML . . . 14
4.7 AutoML + Image Similarity . . . 16
4.8 Méthode retenue . . . 17
4.9 Potentielle amélioration du modèle . . . 18

5 Évaluation **19**

6 Partie MLOps **21**
6.1 Vertex Pipelines . . . 21
6.2 Git et GitLab . . . 24
6.3 Processus de mise en production . . . 24

7 Conclusion **26**

1 Présentation de l'entreprise

1.1 L'entreprise Orange

Orange est une entreprise qui opère dans le secteur des télécommunications et des services numériques, elle a été fondée en 1988 et est aujourd'hui un acteur clé à l'échelle mondiale.
Orange offre une vaste gamme de services et d'innovations et est implantée en France et dans de nombreux pays.
Ses activités englobent la téléphonie fixe et mobile, la fourniture d'accès à Internet, ainsi que des solutions sur mesure pour les entreprises, de la cybersécurité aux solutions de communication unifiée.
En outre, Orange se distingue par ses avancées dans des domaines émergents tels que l'intelligence artificielle, la 5G et l'Internet des objets, contribuant ainsi à façonner la transformation numérique et à répondre aux besoins croissants de connectivité et de services numériques de ses clients et partenaires à travers le monde.

1.2 La DTSI et l'équipe VQI

La Direction Technique et du Système d'information (DTSI) a été créée le 1er juillet 2014 par le rapprochement des directions DTF (Direction Technique France) et DSIF (Direction du Système d'Information France).
La DTSI regroupe les activités d'ingénierie et déploiement, exploitation et intervention sur les réseaux, les services, les SI et les infrastructures, ainsi que de maîtrise d'ouvrage, de maîtrise d'œuvre et de soutien du système d'information en France.
C'est au sein de la DTSI que se trouve l'équipe du chef de projet Big Data, Farid Zamoum (responsable du stage en entreprise).
Elle rassemble une dizaine de personnes, principalement des Data Engineers, Data Scientists, Product Owners et Architectes Data, tous chargés de projets liés à la Big Data et à l'IA.

L'équipe Vérification Qualité des Interventions (VQI) fait partie intégrante de l'équipe dirigée par Farid Zamoum.
Elle est constituée de deux leads Data Scientists, à savoir Marie-Angèle Faye (tutrice de stage) et Alix Martin, accompagnés par deux Data Engineers, Emmanuel Bizieau et Gautier Rouziès, ainsi qu'un Proxy Product Owner, Gaetan Marquis.
C'est au sein de l'équipe VQI que j'ai effectué mon stage.
Par la suite, cette équipe s'est complétée par l'arrivée d'un second stagiaire, Kalidou Dia.

VQI est une solution visant à évaluer la bonne exécution d'une intervention, prévenant ainsi l'accumulation des malfaçons qui peuvent entraîner des litiges entre les opérateurs.
Cette approche permet d'auditer la totalité des interventions en réduisant l'implication humaine dans le processus de vérification.

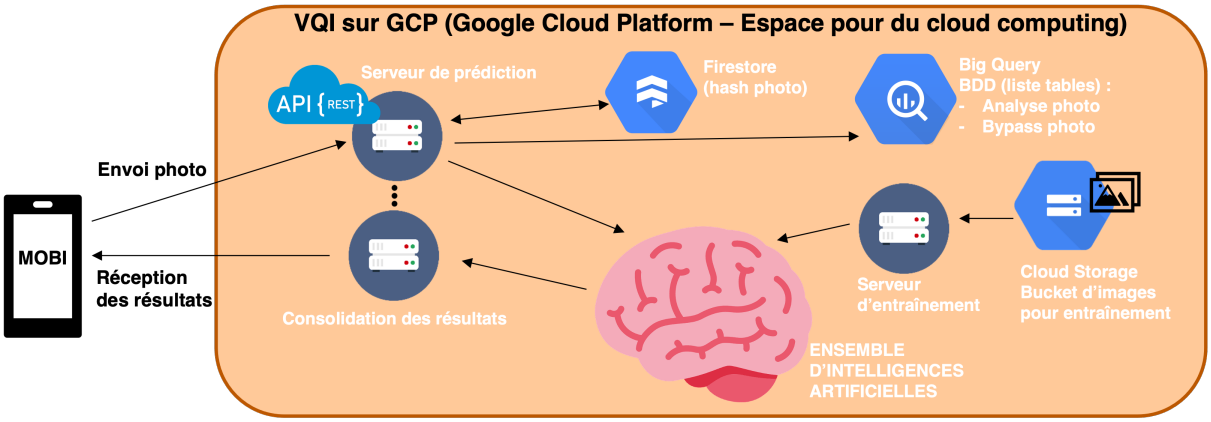


Figure 1 : Schéma fonctionnel de VQI

VQI est hébergé sur la plateforme *Google Cloud Platform* (GCP), car le projet a été initialement développé par une entreprise externe avant d'être repris par Orange.
Cette décision a été prise afin de préserver le travail existant dans son environnement d'origine.
Quant à MOBI, il s'agit d'une application installée sur les appareils mobiles des techniciens, regroupant l'ensemble des fonctions nécessaires aux métiers de l'intervention.
Cette application est largement utilisée par les techniciens chargés de l'installation et de la maintenance de la fibre optique chez les clients Orange.
L'objectif est que seulement trois secondes maximum s'écoulent entre l'envoi d'une photo et la réception des résultats correspondants.
En tout, une dizaine de modèles fonctionnent en permanence sur VQI, incluant des classifications d'images ainsi que des détections d'objets.

2 Résumé du cas d'usage

2.1 Contexte et définition du besoin métier

L'Autorité de régulation des communications électroniques, des postes et de la distribution de la presse (ARCEP) est une autorité administrative indépendante française chargée de réguler les communications électroniques, postales et la distribution de la presse.
Les demandes de l'ARCEP pour le besoin métier se résument aux règles suivantes :

- *« l'opérateur d'infrastructure est responsable du bon fonctionnement, de l'intégrité et de la sécurité de ses réseaux.*
  Si les opérateurs commerciaux et leurs sous-traitants doivent intervenir sur l'infrastructure dans le cadre des interventions en mode dit « STOC » (sous-traitance opérateur commercial), ces interventions restent sous le contrôle contractuel et technique de l'opérateur d'infrastructure. »
- *« L'OC doit: Communiquer un compte rendu d'intervention détaillé et enrichi de photos PTO, photos PBO Avant/Après intervention, photos PM Avant/Après (porte ouverte Gd angle) pour chaque raccordement...*
  Principe d'une grille de sanction progressive et proportionnée d'un OC si non respect par une entreprise sous-traitante du cadre, pouvant aller jusqu'à l'exclusion de l'entreprise concernée. »

où l'Opérateur d'Infrastructure (OI) déploie et exploite les réseaux de télécommunications et l'Opérateur Commerciaux (OC) propose des offres d'accès internet en utilisant le réseau déployé par l'OI.

L'EPIC 341 est un projet qui regroupe plusieurs cas d'usages et a deux buts principaux :

- Vérifier la conformité des photos prises par les techniciens dans le cadre de CRIOC.
  Le projet CRIOC (Compte Rendu d'Intervention Opérateur Commercial) est la mise en œuvre de la demande de l'ARCEP de fourniture d'un compte rendu d'intervention détaillé et enrichi par Orange en tant qu'opérateur commercial.
- Vérifier à chaud la qualité de l'intervention des techniciens, en contrôlant les éventuelles malfaçons.
  Ces techniciens travaillent directement pour Orange OC ou pour des sociétés sous traitantes d'Orange OC (ETR (Entreprise Travaux Réseaux)).

2.2 Cas d'usage et solution actuelle

Le Point de Terminaison Optique (PTO) est un boîtier installé à l'intérieur d'un logement fibré et habituellement fixé au mur à portée de la box fibre.
Ce boitier est équipé d'une prise optique à laquelle se connecte la box Internet pour donner accès au réseau fibre optique de l'opérateur.

La détection des poses de PTO est un cas d'usage de l'EPIC 341 destiné à améliorer la qualité des interventions à l'achat.

Ce cas d'usage comprend du contrôle de conformité à chaud (en temps réel) ainsi que de la détection de malfaçons à chaud et se découpe en trois phases d'analyse :

- La validation de la conformité de la photo avant intervention
- La détection de PTO sur la photo après intervention
- La détection de pose de PTO

- Le premier point comprend la détection d'un début de construction par un classifieur d'équipement (label `pto_avant`).
  Un début de construction peut être la présence d'un trou dans un mur avec un câble sortant, la présence de deux trous de fixation sur un mur et un trou avec câble sortant, la présence d'un support de pose de PTO ou la présence de rail DIN.

- Le deuxième point comprend la détection de l'équipement PTO par un classifieur d'équipement (label `pto`).

- Le troisième point comprend la validation des points précédents ainsi qu'une vérification de la date de prise de la photo après intervention qui doit être postérieure de plus de 1min30s par rapport à la photo avant intervention et antérieure à 23h59 de la même date.

2.3 Axe d'amélioration

La comparaison des photos avant et après l'intervention intervient en complément de la troisième phase d'analyse pour ajouter une vérification supplémentaire dans le but d'améliorer la fiabilité des résultats, notamment en détectant d'éventuels cas de fraudes.
Cette comparaison consiste à vérifier la présence d'éléments similaires entre les deux images, afin de confirmer la nouvelle pose de PTO ou non.

Les difficultés que l'on peut rencontrer lors de cette amélioration concernent principalement les différences de position, de zoom, de rotation, de qualité et de luminosité entre les photos avant et après intervention.
Le recul sur les murs peut également poser une difficulté, car il faudrait disposer d'assez de contextes sur les deux photos pour pouvoir les comparer convenablement.

Avant

Après

Figure 2 : Exemple de photo avant et après intervention (début de construction : support de pose)

3 Google Cloud Platform

Google Cloud Platform [17] (GCP) est une plateforme cloud complète qui permet aux entreprises et aux développeurs de créer, déployer et gérer des applications et des services en utilisant les ressources et les outils fournis par Google dans le cloud.
C'est l'environnement principal de travail de l'équipe VQI, où la grande majorité du travail est effectuée sur cette plateforme, que ce soit pour les tâches de data engineering ou de data science.
GCP fournit une multitude d'applications polyvalentes et optimisées afin de faciliter le déroulement des projets de bout en bout.
Google donne accès à l'ensemble de ses ressources, aux modèles pré-entraînés, aux CPU et GPU, ainsi qu'à des documentations complètes pour chaque outil.

3.1 Services utilisés

Voici une brève introduction des services GCP utilisés durant mon stage (hors *Vertex AI*):

- `BigQuery` [19] est un service cloud qui propose une analyse rapide de grandes quantités de données à l'aide de requêtes `SQL`.
  Il s'intègre facilement avec d'autres services de GCP, notamment le `Cloud Storage`.
  Dans le cadre du cas d'usage, les requêtes permettent principalement de récupérer des images dans le `Cloud Storage` avec des filtres spécifiques en fonction du besoin.
  `BigQuery` m'a été utile dans la partie évaluation du modèle, qui sera détaillée dans la suite du rapport.
- `Cloud Storage` [25] est un service de stockage pour de grandes quantités de données dans des buckets, c'est ici que sont stockées toutes les photos prises par les techniciens.
  On y conserve également les fichiers nécessaires à l'entraînement des modèles ainsi qu'au fonctionnement des pipelines.
  Le requêtage sur le `Cloud Storage` est facilité par `BigQuery`, et l'importation de données est réalisée grâce au *Google Cloud SDK* [23] et à la fonction `gsutil -m cp -r`, qui permet de lier des sources de données extérieures à GCP.
- `Billing` [3] est un service qui permet de visualiser et de gérer les dépenses liées à l'utilisation des services cloud de Google.
  On peut notamment y consulter un tableau de bord des coûts ainsi que la facturation détaillée de chaque service utilisé sur GCP.
  `Billing` m'a été utile dans la partie d'évaluation du modèle.

3.2 Vertex AI

Vertex AI [1] est l'outil principal que j'ai utilisé durant mon stage.
C'est une plateforme de machine learning entièrement gérée et développée par GCP.
Elle propose un ensemble complet d'outils et de services pour faciliter la création, la formation et le déploiement de modèles d'apprentissage automatique à grande échelle.
L'avantage de *Vertex AI* réside dans la simplification du processus d'apprentissage automatique, permettant une réduction de la complexité opérationnelle et une optimisation des performances.

Vertex AI possède plusieurs outils, voici les principaux utilisés lors du stage :

- `Workbench` est un environnement de développement collaboratif permettant de créer des notebooks basés sur différents environnements tels que *PyTorch* ou *Tensorflow* [29].
  Il vise à simplifier le processus de création, de collaboration et de gestion des projets d'apprentissage automatique.
  J'ai principalement utilisé l'outil `Workbench` pour explorer mon cas d'usage, car les notebooks sont directement connectés aux autres outils de GCP, ce qui facilite la transition entre les outils.
  De plus, `Workbench` offre la possibilité de choisir les spécifications techniques de nos notebooks, en sélectionnant les ressources CPU et GPU proposées par Google.
  Cette approche permet de réduire le temps de calcul et d'exécution des modèles complexes, ce qui favorise la fluidité de l'exploration.

- `Pipelines` est un système de gestion de flux de travail qui simplifie la mise en œuvre des processus d'apprentissage automatique en automatisant des étapes telles que le prétraitement des données, la formation des modèles, l'évaluation des performances et le déploiement en production.
  On peut créer des workflows en utilisant des composants préexistants, en en développant de nouveaux ou en réutilisant des parties de workflows déjà créées.
  Cette approche facilite l'organisation et la maintenance des projets, tout en permettant une bonne collaboration entre les membres de l'équipe, de plus, les workflows sont facilement partageables et reproductibles.
  Enfin, `Pipelines` gère automatiquement l'exécution des différentes tâches du workflow, ce qui permet de se concentrer sur le développement du modèle plutôt que sur les détails de l'infrastructure.
  Cette partie sera développée en détail dans la suite du rapport.

- `Datasets` est une plateforme qui permet de stocker, importer, explorer et préparer les données nécessaires pour l'entraînement des modèles.
  Cet outil permet d'explorer les données afin de comprendre et de visualiser leurs caractéristiques, de détecter les valeurs manquantes et de vérifier leur qualité avant de les utiliser pour l'entraînement des modèles, on peut y effectuer des opérations de prétraitement, de nettoyage et de transformation des données pour les préparer à l'apprentissage automatique.
  De plus, il est possible de créer plusieurs versions des jeux de données, ce qui facilite la gestion de l'historique des modifications et garantit la reproductibilité des expériences.
  Enfin, `Datasets` est intégré à d'autres services de *Vertex AI*, tels que `Training`, ce qui facilite l'entraînement des modèles avec les jeux de données préparés.

- `Training` est une fonctionnalité qui permet d'entraîner des modèles d'apprentissage automatique sur des jeux de données à grande échelle.
  Cet environnement de formation hautement optimisé et entièrement géré permet de déployer des algorithmes d'apprentissage automatique ou d'utiliser des solutions *AutoML* pour créer des modèles entièrement gérés par Google.
  L'outil `Training` sera utilisé dans le cas d'usage en tant que méthode à part entière, notamment dans le cadre de l'*AutoML*.
  Il est possible de personnaliser les options avant de lancer l'entraînement, notamment en choisissant la méthode de découpage du jeu de données.
  On peut choisir le découpage automatique proposé par Google, en précisant le pourcentage de données dans chaque ensemble, ou effectuer un découpage manuel en ajoutant une colonne spécifique au jeu de données utilisée pour l'entraînement.
  De plus, on peut choisir le nombre d'heures de calcul utilisées pour entraîner les modèles *AutoML*, appelées `Node Hours`, ainsi que la métrique à optimiser.
  Dans le cadre de ce cas d'usage de classification d'images, il est possible d'optimiser le temps d'entraînement avec la métrique par défaut `Low Accuracy`, ou bien d'optimiser l'Accuracy au maximum avec la métrique `Higher Accuracy`, ce qui prolonge la durée de l'entraînement.
  Enfin, `Training` offre également la possibilité d'entraîner de nouvelles versions des modèles en ré-entraînant ces derniers à partir des paramètres résultants de leurs dernières versions, ce qui peut être pratique lorsque de nouvelles données sont disponibles.

- `Model registry` est une fonctionnalité qui permet de gérer, de stocker et de suivre les différentes versions des modèles.
  On peut y enregistrer les modèles entraînés, prendre en charge les différentes versions, ce qui facilite la gestion de l'historique des modifications et permet d'effectuer des comparaisons entre les différentes itérations.
  De plus, on peut consulter les métriques de performance des modèles pour chaque version.
  C'est ici que sont stockés tous les modèles entraînés du projet VQI.

- `Online prediction` est une fonctionnalité qui permet de déployer des modèles d'apprentissage automatique entraînés pour effectuer des prédictions en temps réel.
  Cette option offre la possibilité de réaliser des prédictions en temps réel en utilisant les modèles déployés, et elle propose une interface API qui permet aux applications et aux services externes d'envoyer des requêtes afin d'obtenir des prédictions à partir de ces modèles.
  J'ai largement fait usage des `Sample requests` de `Online Prediction` puisqu'ils fournissent un code Python permettant d'appeler les modèles déployés pour effectuer des prédictions directement depuis les notebooks du `Workbench`.

4 Approches

Cette partie va présenter les méthodes et algorithmes qui ont été explorés pour résoudre le cas d'usage.
Nous allons nous concentrer sur trois d'entre elles, ainsi qu'un potentiel axe d'amélioration dans le futur en cas de besoin.
Les notebooks contenant les implémentations des trois méthodes sont disponibles via le lien suivant : lien

4.1 Le jeu de données

Le jeu de données à disposition comprend 9705 paires de photos avant et après intervention de nouvelle pose de PTO.
Chaque image de ce jeu de données a été soumise à un classifieur d'équipement, qui a systématiquement renvoyé `pto_avant` pour les photos avant intervention et `pto` pour les photos après intervention.
Ainsi, les deux premières phases d'analyse ont été validées.
Le delta de temps de 1 minute 30 secondes n'a pas pu être vérifié, car lors de la conception du jeu de données, la date et l'heure de prise de photo n'étaient pas encore stockées de façon permanente, toutefois, cela n'est pas bloquant pour le modèle de comparaison d'images.
Un échantillon du jeu de données est disponible via le lien suivant : lien

4.2 Labellisation

4.2.1 Label Studio

Label Studio [26] est une plateforme open-source conçue pour annoter et étiqueter des données en vue de l'apprentissage automatique et de l'intelligence artificielle.
C'est un outil de gestion de l'annotation de données qui facilite le processus de création d'ensembles de données annotées pour l'entraînement de modèles d'apprentissage automatique supervisés.
Pour la détection de pose de PTO, nous utilisons un outil de classification d'images qui consiste à attribuer un label à chaque image.
Les données sont importées sur *Label Studio* directement depuis les bases de données de photos (`Cloud Storage`) à l'aide d'un kit de développement logiciel (SDK) qui permet d'appeler l'API *Label Studio* à partir de scripts Python.
La tâche de labellisation peut être répartie entre les membres de l'équipe, la plateforme permettant d'assigner des données à labelliser à chaque membre.

Le but de cette répartition est d'accomplir la tâche plus rapidement et de travailler de manière indépendante en fonction des disponibilités de chacun.
La méthode de labellisation joue un rôle crucial, car elle influe directement sur les performances des modèles. Il est donc essentiel d'organiser une réunion de synchronisation avec l'équipe avant de commencer cette tâche.
Une fois la labellisation terminée, la plateforme propose un système de review où chacun est assigné à revoir le travail d'un autre, cela permet de corriger d'éventuelles erreurs de labellisation.
Une fois la tâche terminée, les données ainsi que les labels peuvent être exportés dans différents formats, généralement au format `json` ou `csv`.
Voici un exemple de sortie d'une ligne au format `json` :


```
1  {
2      "id": 2356857685,
3      "data": {
4          "image_value": "gs://ofr-vqi-data-pipeline-models-dev/notebook/Idir/AUTO_ML_PAIRE_PTO_V2/data/17716947_3092.jpeg"
5      },
6      "annotations": [
7          {
8              "result": [
9                  {
10                     "value": {
11                         "choices": ["Same"]
12                     }
13                 }
14             ]
15         }
16     ]
17 }
```


4.2.2 Découpage du jeu de données

La labellisation a été initialement réalisée en utilisant trois classes pour les 9705 paires d'images :

- **Same** : cette classe est attribuée lorsque les éléments de contexte entre les deux images sont similaires, indiquant que l'intervention a eu lieu au même endroit.

- **Not same** : cette classe est assignée lorsque les éléments de contexte entre les deux images diffèrent, signifiant que l'intervention n'a pas eu lieu au même endroit.

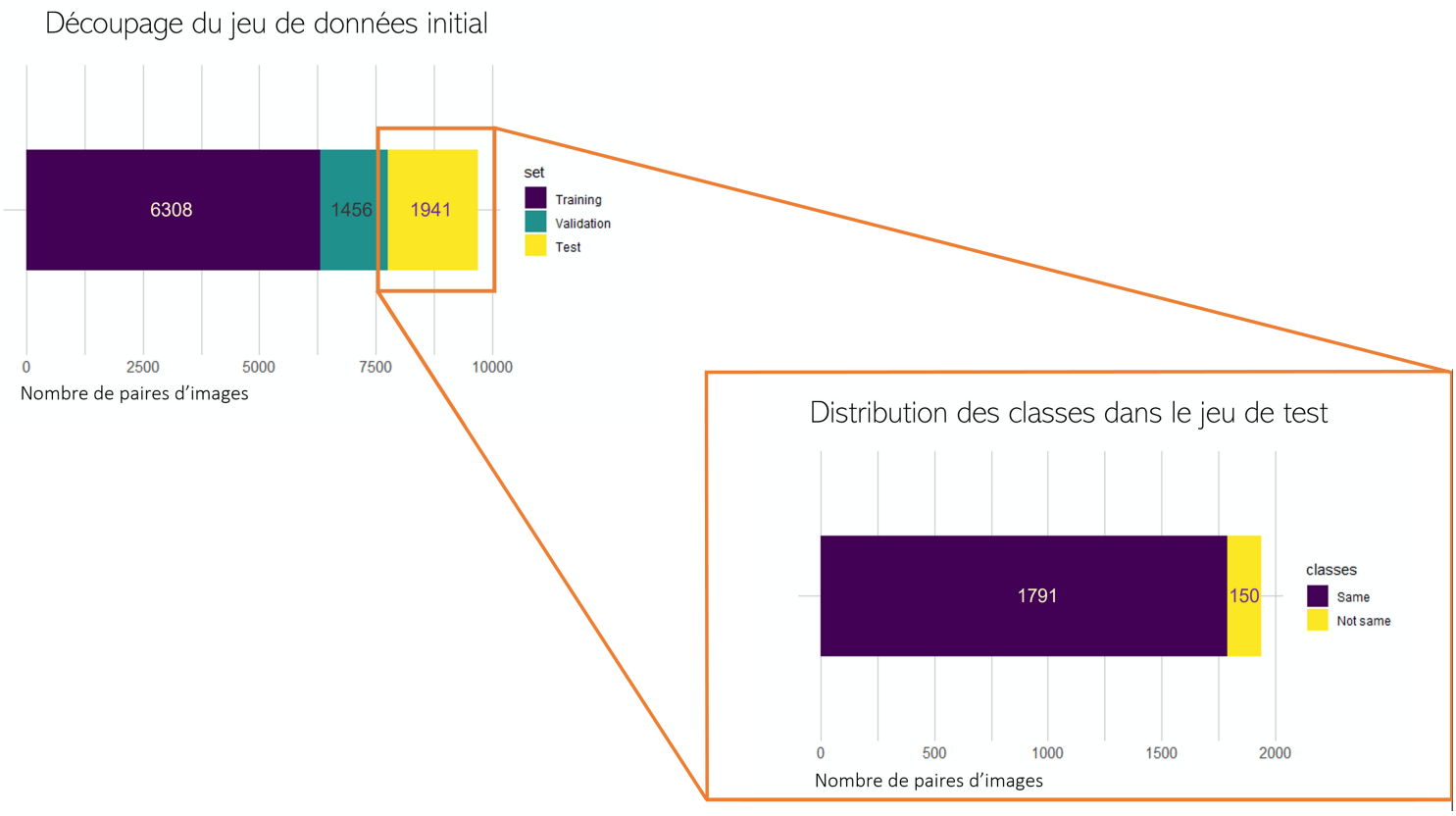
- **Not enough context elements** : cette classe est utilisée lorsque l'on ne peut pas déterminer intuitivement si l'intervention est valide.
  Les types de paires de photos incluses dans cette classe sont les suivants : la PTO de la photo après intervention n'est pas positionnée au même endroit que la trace de construction de la photo avant intervention, ou la photo avant intervention n'est pas prise au même endroit que la photo après intervention malgré des adresses, dates et heures de prise similaires.

Sur les 9705 paires d'images, les résultats de la labellisation sont les suivants : 84.4 % sont classées comme **Same**, 8.6 % sont classées comme **Not enough context elements** et 7 % sont classées comme **Not same**.

Figure 3 : Exemples de paires d'images respectivement labellisées Same, Not Same et Not enough context elements

Après avoir testé des premiers modèles sur un échantillon des données labellisées, il est apparu que les classes **Same** et **Not enough context elements** ne présentent pas de différences significatives du point de vue de leurs scores de prédiction, qui sont globalement très proches.
Par conséquent, la classe **Not enough context elements** sera intégrée à la classe **Same**, dans le but de réduire les erreurs de prédiction pour ces deux catégories.
Le modèle final comportera donc deux classes : **Same** et **Not same**.
Parmi les 9705 paires d'images, 93 % d'entre elles sont classées comme **Same**, tandis que 7 % sont classées comme **Not same**.

Le découpage du jeu de données initial de 9705 paires d'images en ensembles d'entraînement, de validation et de test a été réalisé de la manière suivante :



Le découpage du jeu de test a été effectué de manière à ce que la proportion des deux classes du jeu initial soit maintenue, c'est-à-dire environ 93 % de paires d'images classées **Same** et 7 % classées **Not same**, dans le but de refléter les proportions réelles des deux classes.

4.3 Notebooks et IDE

L'exploration des approches du cas d'usage a été réalisée à l'aide de notebooks en Python.
Le principal outil que j'ai utilisé est *JupyterLab*, une interface utilisateur web interactive plus complète que *Jupyter Notebook*, car elle permet également la gestion directe des fichiers et des dossiers grâce à un onglet dédié.
J'ai utilisé le *JupyterLab* inclus dans l'outil *Vertex AI*, car il est directement connecté aux autres outils de GCP.
R Studio m'a principalement servi à rédiger les rapports destinés au métier en Rmarkdown.
Excel m'a été utile pour organiser les jeux de données au format `csv`.
La majorité des codes sont au format `ipynb`, de plus, j'ai utilisé *Visual Studio Code* pour mieux structurer mon code et pour la partie MLOps (détaillée dans la suite du rapport).

4.4 Image Matching

La première méthode pour résoudre ce cas d'usage consiste à utiliser un algorithme basé sur des descripteurs de caractéristiques, couplé à un réseau de neurones profond de type GNN nommé *Super Glue* [22].
L'idée principale est de trouver des points similaires entre les deux photos (des correspondances), ainsi plus il y a de points en commun, plus les deux photos ont de chances d'avoir des contextes similaires.

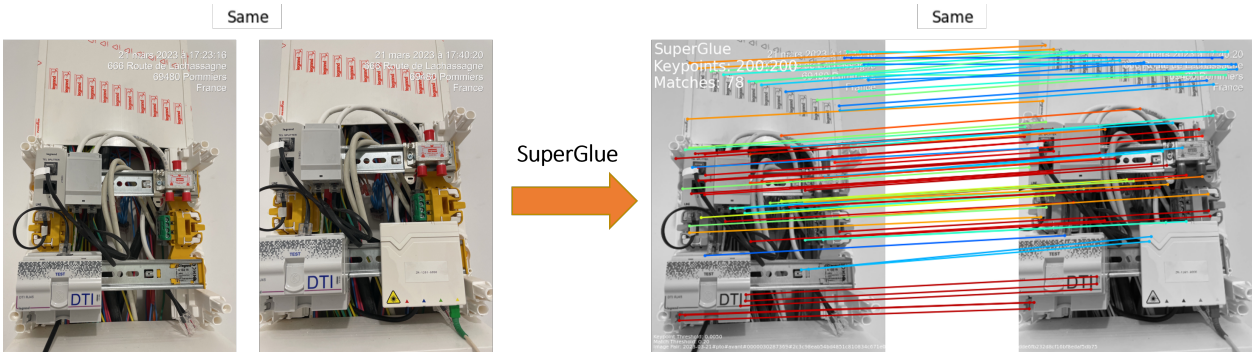


Figure 4 : Exemple d'utilisation de l'algorithme Super Glue

Cet algorithme possède plusieurs hyperparamètres personnalisables, dont un redimensionnement des images fixé à (640×640), le nombre maximum de correspondances fixé à 200 et le seuil de confiance minimum pour considérer deux points comme étant en correspondance, fixé à 20%.

On pourrait imaginer un système comptant le nombre total de correspondances entre deux photos, ou évaluant une paire de photos en fonction du score de confiance des correspondances, pour classer ces photos en **Same** ou **Not same**.
Cependant, on se heurte rapidement à un problème : les photos à disposition possèdent souvent des informations textuelles qui sont générées au moment de la prise de vue.
Ces informations incluent les adresses, les villes, les départements où les interventions se passent, ainsi que les dates et les heures.
L'algorithme se concentre énormément sur ces informations textuelles au détriment du reste de l'image, ce qui crée un nombre d'erreurs non négligeable à l'échelle du jeu de données entier.

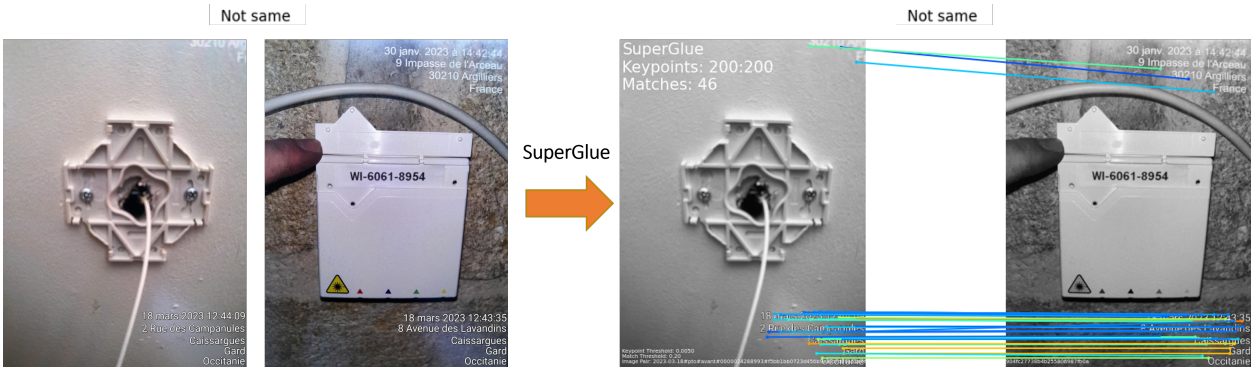


Figure 5 : Exemple d'utilisation de l'algorithme Super Glue

L'exemple ci-dessus illustre une situation où la paire de photos est labellisée comme **Not same**, cependant *Super Glue* trouve 46 correspondances, qui se trouvent toutes au niveau des informations textuelles.
L'algorithme ne prend pas en compte le reste des deux photos, alors que dans ce cas précis, la photo avant intervention n'est pas au même endroit que la photo après intervention.
Étant donné que le nombre de photos comportant des informations textuelles n'est pas négligeable dans le jeu de données, et est difficile à calculer pour les nouvelles interventions car il dépend du technicien (si la localisation de son téléphone est active ou pas), cette méthode n'a pas été retenue pour ce cas d'usage, malgré son potentiel.

4.5 Image Similarity

4.5.1 Libraries et modèles

La librairie principale de Computer Vison utilisée pour les modèles pré-entraînés dans ce cas d'usage est *PyTorch* [18], avec son outil *torchvision* [30].
PyTorch est une plateforme populaire d'apprentissage profond qui offre une approche flexible et dynamique pour la création de modèles d'apprentissage automatique.
L'utilisation de *PyTorch* simplifie l'exploitation du cas d'usage car il est intégré à *Vertex AI*, ce qui facilite la mise en œuvre et le déploiement des modèles.
torchvision nous donne accès à une multitude de modèles de classification d'images pré-entraînés sur le célèbre jeu de données *ImageNet* [13], qui comprend plus de 1200000 images réparties en 1000 classes.
Ces modèles sont principalement basés sur des réseaux de neurones convolutifs (CNN).

La librairie principale utilisée pour la classification finale dans la méthode Image Similarity est *scikit-learn* [15].
scikit-learn propose un large éventail d'algorithmes d'apprentissage supervisé et non supervisé, ainsi que des outils pour prétraiter les données, évaluer les modèles et accomplir des tâches courantes en apprentissage automatique.
Des modèles pré-entraînés de *PyTorch*, ainsi que plusieurs algorithmes de classification de *scikit-learn*, sont utilisés dans la méthode Image Similarity.

4.5.2 CNN et Image Embedding

La deuxième méthode pour résoudre ce cas d'usage repose sur trois phases :

- Utiliser un modèle de réseaux de neurones convolutifs [6] (CNN) pré-entraîné sur le jeu de données *ImageNet*.
  Le but est d'extraire des vecteurs caractéristiques en prenant l'avant dernière couche du modèle en tant que couche de sortie pour chacune des deux photos d'une intervention, ce qui constitue l'embedding des images.

- Les vecteurs caractéristiques seront utilisés pour calculer des distances spatiales dans un espace à n dimensions (ici $n = 2048$).

- Les distances calculées seront utilisées comme données d'entrée pour un classifieur qui déterminera si les images sont classées en tant que **Same** ou **Not same**.

Plusieurs modèles pré-entraînés ont été testés, parmi lesquels figurent *MobileNet* [21], *EfficientNet* [28] ainsi que *VGG16* [24], cependant, pour ce cas d'usage, le choix s'est porté sur *ResNet152* [12].
Les distances calculées comprennent les distances Cosine, Braycurtis, Euclidienne, Chebyshev et Canberra.
Quant aux classifieurs utilisés, ils font partie de la librairie *scikit-learn* (SVM, Random Forest...) ainsi que *XGBoost* [4] et *LightGBM* [14].
De plus, il est à noter que la taille des images en entrée de *ResNet152* est de (224×224) pixels.

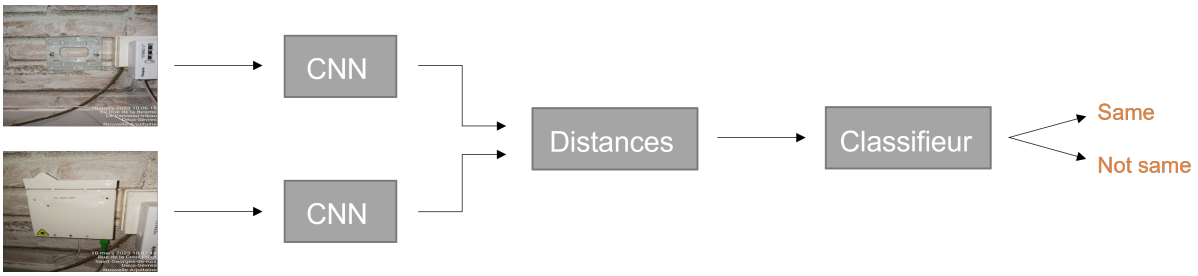


Figure 6 : Architecture de la méthode Image Similarity

Les classifieurs de *scikit-learn* qui ont produit les résultats les plus intéressants sont le *RidgeClassifier* [20], l'*ExtratreesClassifier* [7] ainsi que le *SVC* [27], cependant, chacun d'entre eux a présenté des résultats différents.
Ci-dessous, vous trouverez les trois matrices de confusion des modèles appliqués au jeu de données de test :

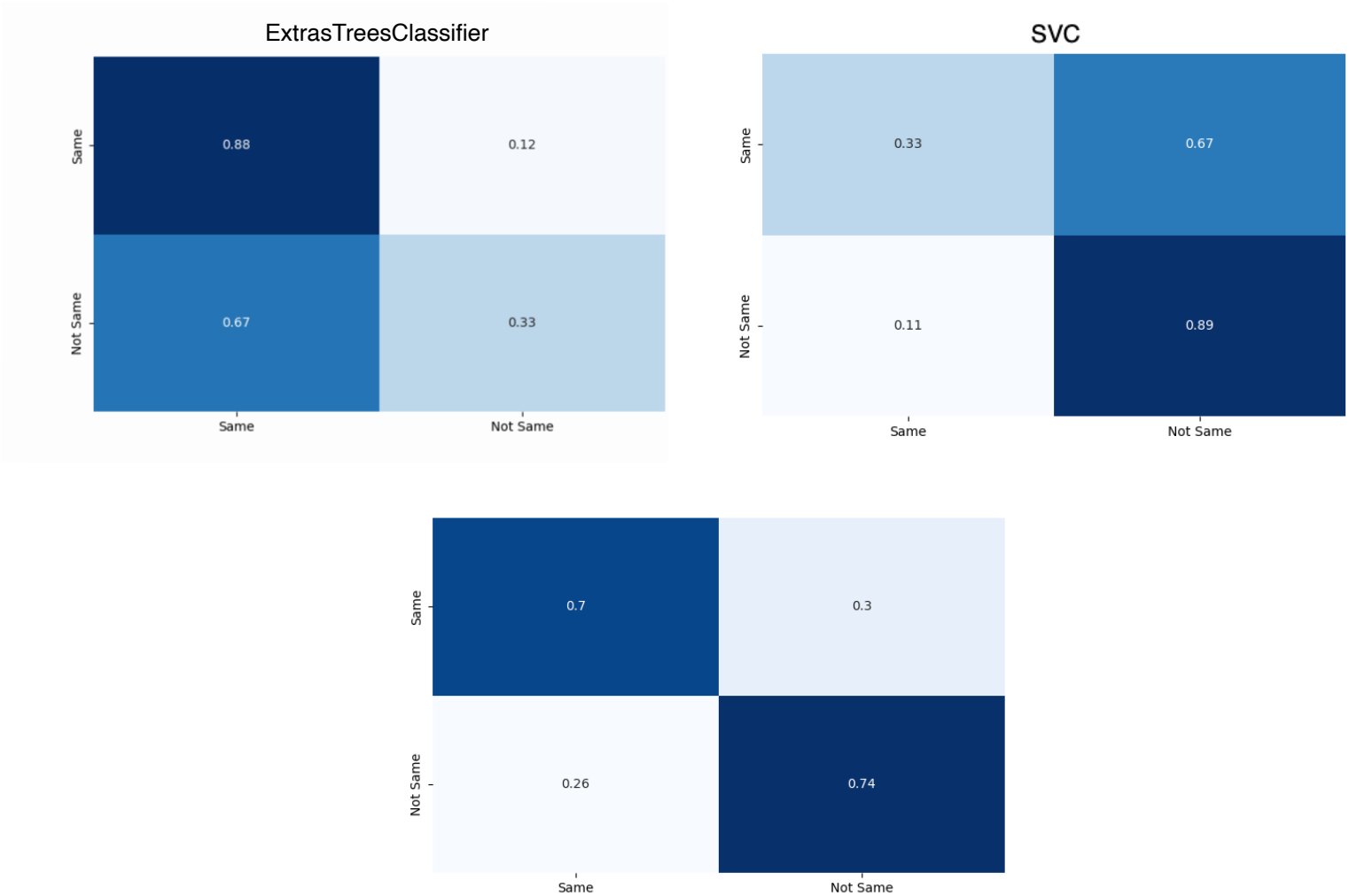


Figure 7 : Matrices de confusion avec la méthode Image Similarity avec RidgeClassifier, ExtratreesClassifier et SVC

On rappelle que le jeu de test est composé de 1941 paires de photos dont 150 paires classées **Not same**.
L'algorithme *ExtraTreesClassifier* présente un très bon pourcentage pour la classe **Same** avec 88 % de prédictions correctes, mais il n'est pas du tout performant pour la classe **Not same**, avec seulement 33 % de prédictions correctes.
À l'inverse, l'algorithme *SVC*, qui est en fait un SVM, présente un très bon taux de réussite pour la classe **Not same**, avec 89 % de prédictions justes, mais il n'est pas du tout performant pour la classe *Same*, avec seulement 33 % de prédictions correctes.
Enfin, l'algorithme *RidgeClassifier* a globalement de bonnes performances sur les deux classes, mais cela n'est pas suffisant pour répondre aux objectifs de ce cas d'usage.
La méthode d'Image Similarity pourrait éventuellement être pertinente en fonction des besoins métier et des objectifs spécifiques de ce cas d'usage.
Il est possible que l'on cherche plutôt à détecter les cas **Not same** de manière drastique, même si cela entraîne de mauvaises performances sur la classe **Same**, ou inversement.

4.6 AutoML

La troisième méthode pour résoudre ce cas d'usage consiste à utiliser un classifieur d'images du module *AutoML* de l'outil `Training` de *Vertex AI* sur GCP.
La phase d'entraînement est entièrement gérée par Google, ce qui signifie que le seul travail à effectuer est de préparer un jeu de données au format approprié (CSV), d'effectuer un prétraitement des photos et de régler quelques paramètres.
Le format [2] demandé par Google est le suivant : une colonne `set` qui indique l'ensemble auquel appartient l'image (`TRAIN`, `TEST` ou `VALIDATION`), une colonne `gs_uri` qui contient l'URL de l'image stockée dans le `Cloud Storage`, et une colonne `label` qui représente le label de l'image, soit **Same** ou **Not Same**.

training	gs://ofr-vqi-data-pipeline-models-dev/notebook/ldir/AUTO_ML_PAIRE_PTO_V2/data/29546047_11157.jpeg	Same
training	gs://ofr-vqi-data-pipeline-models-dev/notebook/ldir/AUTO_ML_PAIRE_PTO_V2/data/17997103_3725.jpeg	Same
training	gs://ofr-vqi-data-pipeline-models-dev/notebook/ldir/AUTO_ML_PAIRE_PTO_V2/data/29488159_10884.jpeg	Not same
training	gs://ofr-vqi-data-pipeline-models-dev/notebook/ldir/AUTO_ML_PAIRE_PTO_V2/data/29551570_11199.jpeg	Same
training	gs://ofr-vqi-data-pipeline-models-dev/notebook/ldir/AUTO_ML_PAIRE_PTO_V2/data/23979910_9034.jpeg	Same
training	gs://ofr-vqi-data-pipeline-models-dev/notebook/ldir/AUTO_ML_PAIRE_PTO_V2/data/18027341_4256.jpeg	Same
training	gs://ofr-vqi-data-pipeline-models-dev/notebook/ldir/AUTO_ML_PAIRE_PTO_V2/data/17992720_3606.jpeg	Same
training	gs://ofr-vqi-data-pipeline-models-dev/notebook/ldir/AUTO_ML_PAIRE_PTO_V2/data/23843145_8278.jpeg	Same
training	gs://ofr-vqi-data-pipeline-models-dev/notebook/ldir/AUTO_ML_PAIRE_PTO_V2/data/23262196_6574.jpeg	Same
training	gs://ofr-vqi-data-pipeline-models-dev/notebook/ldir/AUTO_ML_PAIRE_PTO_V2/data/30264433_11535.jpeg	Same
training	gs://ofr-vqi-data-pipeline-models-dev/notebook/ldir/AUTO_ML_PAIRE_PTO_V2/data/29521620_11003.jpeg	Same
training	gs://ofr-vqi-data-pipeline-models-dev/notebook/ldir/AUTO_ML_PAIRE_PTO_V2/data/23270952_6801.jpeg	Same
training	gs://ofr-vqi-data-pipeline-models-dev/notebook/ldir/AUTO_ML_PAIRE_PTO_V2/data/29565061_11349.jpeg	Same
training	gs://ofr-vqi-data-pipeline-models-dev/notebook/ldir/AUTO_ML_PAIRE_PTO_V2/data/11231890_1129.jpeg	Same
training	gs://ofr-vqi-data-pipeline-models-dev/notebook/ldir/AUTO_ML_PAIRE_PTO_V2/data/9183208_650.jpeg	Not same
training	gs://ofr-vqi-data-pipeline-models-dev/notebook/ldir/AUTO_ML_PAIRE_PTO_V2/data/23762567_7119.jpeg	Same

Figure 8 : Exemple de jeu de données en entrée de l'AutoML

Les paramètres réglés sont le nombre de `Node Hours` fixé à 50, ainsi que le choix de l'accuracy fixé à `Higher Accuracy`.
Le prétraitement réalisé consiste en la concaténation des photos avant et après intervention en utilisant la hauteur maximale entre les deux photos comme hauteur commune.
En fin de compte, il n'y aura qu'une seule image en entrée de l'*AutoML*.

Figure 9 : Exemple d'une image en entrée de l'AutoML composée des photos avant et après intervention concaténées

Par ailleurs, afin d'équilibrer les classes, une augmentation de données a été effectuée dans le jeu d'entraînement pour la classe **Not same**.
Finalement, le jeu de données d'entraînement comprend 9533 paires d'images, contre aux 6309 initialement.
La proportion de paires d'images classées **Same** passe de 93 % à 68 %, tandis que celle des paires d'images classées **Not same** augmente de 7 % à 32 % dans le jeu d'entraînement.

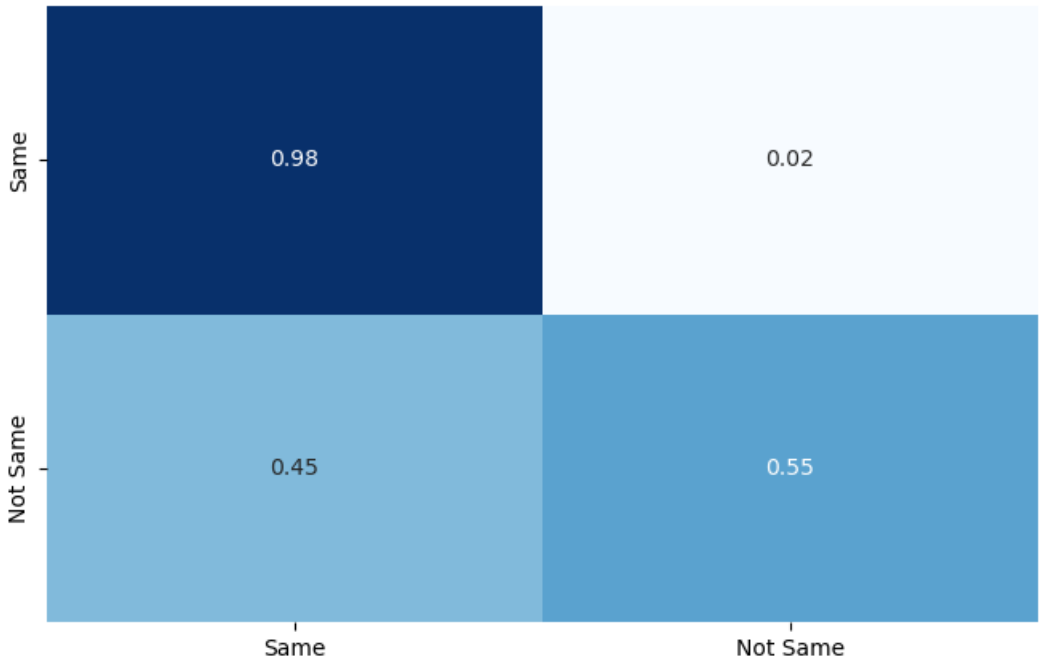


Figure 8 : Matrice de confusion du modèle AutoML

Le modèle *AutoML* présente des résultats assez satisfaisants avec 98 % de bonnes prédictions sur les images classées **Same** et 55 % de bonnes prédictions sur les images classées **Not same**.
Étant donné le déséquilibre marqué entre les classes dans le jeu de test, ces résultats sont rassurants.
Le modèle est enregistré dans `Model Registry` de *Vertex AI*, où l'on peut notamment retrouver la matrice de confusion ainsi que les graphiques de précision et de recall du modèle.
Cette méthode semble être une solution pertinente pour le cas d'usage.

4.7 AutoML + Image Similarity

La dernière méthode pour résoudre ce cas d'usage consiste à combiner l'approche d'Image Similarity et l'AutoML afin de tirer parti des avantages de chacune de ces deux méthodes pour affiner les résultats.
La combinaison des deux méthodes se fait au moyen d'un système de vote similaire à un petit arbre de décision.
D'un côté, il favorise l'AutoML pour la classe **Same** en raison de ses excellentes performances, et d'un autre côté, il privilégie l'approche d'Image Similarity pour la classe **Not same**, en particulier avec l'utilisation de l'algorithme *SVC* qui présente de bonnes performances pour cette classe.

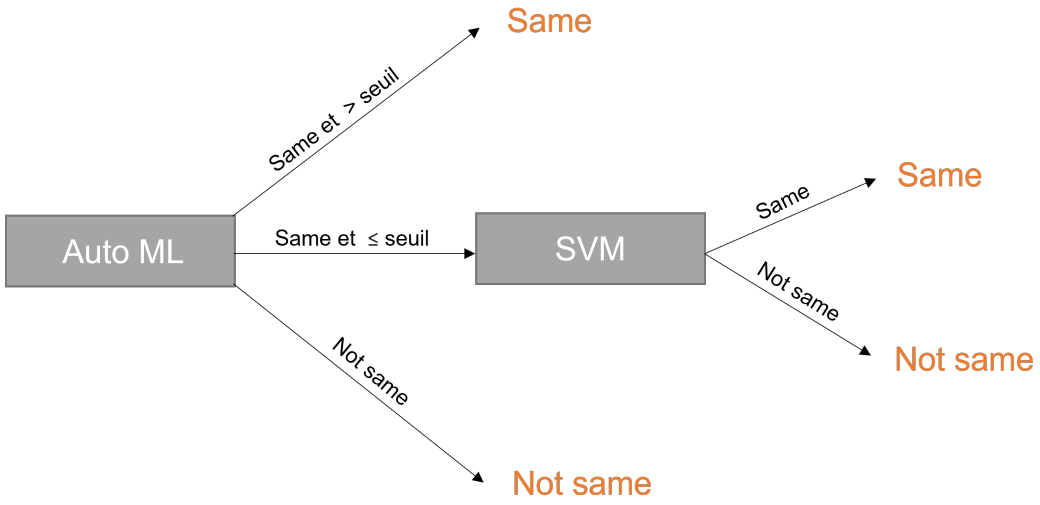


Figure 10 : Sytème de vote pour la combinaison AutoML + Image Similarity

Soit VP_{score} la moyenne des scores des bonnes prédictions pour la classe **Same** et FP_{score} la moyenne des scores des mauvaises prédictions pour la classe **Same**.
Le seuil de discrimination est défini comme suit :

$$\frac{VP_{score} + FP_{score}}{2}$$

Voici la matrice de confusion du modèle AutoML + Image Similarity appliqué au jeu de données de test :

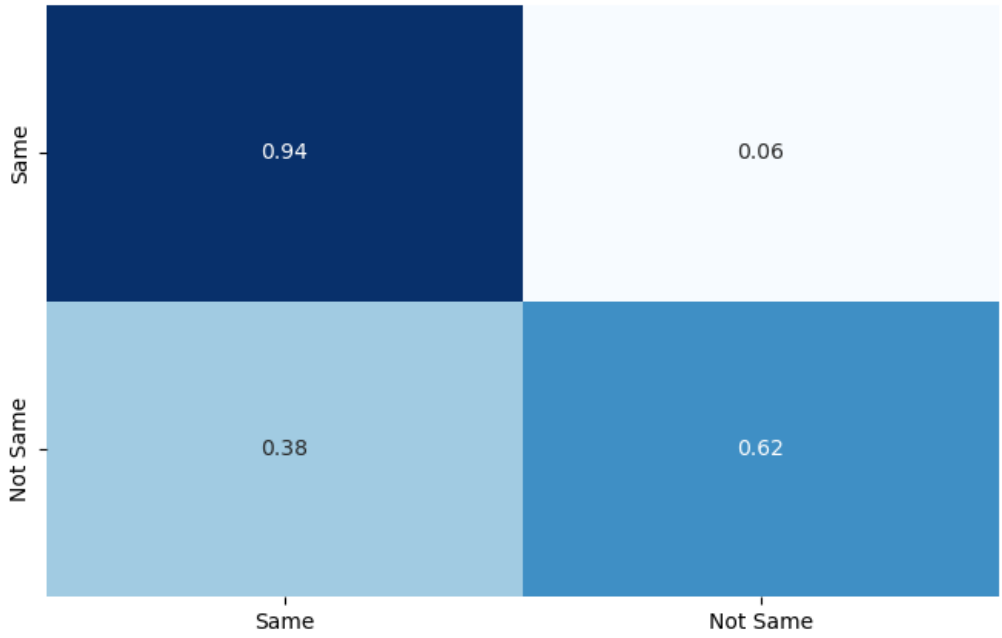


Figure 11 : Matrice de confusion du modèle AutoML + Image Similarity

En se basant sur la matrice de confusion pour la méthode AutoML, pour une diminution de 4 % de bonnes performances sur la classe **Same**, on observe une augmentation de 7 % de bonnes performances sur la classe **Not same**.
Cette méthode peut être une bonne solution pour le cas d'usage, cependant, il faut faire preuve de prudence. Étant donné que la classe **Same** est sur-représentée, une diminution de 1 % dans les performances de cette classe a un impact plus important qu'une augmentation de 1 % des performances de la classe **Not same**.

4.8 Méthode retenue

Les trois méthodes utilisées ont été présentées au métier, et la méthode retenue est l'AutoML.
Cette méthode offre d'excellentes performances sur la classe **Same**, tout en maintenant des performances assez bonnes sur la classe **Not same**.
Étant donné que l'objectif du cas d'usage est de détecter d'éventuels cas de fraudes, il est impératif de maintenir un niveau élevé de performance sur la classe **Same**, ce qui est le cas ici.
De plus, ce modèle est plus simple à entretenir car il utilise l'*AutoML* de *Vertex AI*, ce qui le rend facilement exploitable en cas de besoin.
Par conséquent, le modèle est déployé sur `Online Prediction` de *Vertex AI*, permettant ainsi de réaliser des prédictions en utilisant les `Sample Requests`.

4.9 Potentielle amélioration du modèle

Au cours de l'exploration du cas d'usage, d'autres méthodes ont été testées sur de petits jeux de données. C'est notamment le cas de la méthode *YOLO + Segmentation + AutoML*.
Les difficultés rencontrées dans ce cas d'usage sont liées aux différences de position, de zoom et de rotation entre les images.
Pour résoudre ces problèmes, une méthode a été imaginée, impliquant les étapes suivantes :

1. Utilisation du modèle de détection d'objet *YOLOv5* [33] pour appliquer un recadrage aux photos avant et après intervention.
   Le but est de conserver uniquement la partie des traces de construction pour la photo avant et la partie de la PTO sur la photo après, tout en incluant un peu de contexte autour.

2. Réduction des biais liés aux différences de position, de zoom et de rotation en effectuant des segmentations de la trace de construction et de la PTO.
   On met en place des masques noirs à la place pour conserver uniquement le contexte autour de l'intervention.
   Pour cela, on peut utiliser un algorithme *UNet* [31] ou le récent algorithme de segmentation de *Meta* appelé *Segment Anything Model* [16] (SAM).

3. Lancement d'un modèle *AutoML* en effectuant le prétraitement de concaténation des photos.

Cette approche vise à atténuer les problèmes liés aux différences de position, de zoom et de rotation en réduisant les biais potentiels.

Figure 12 : Architecture de la méthode YOLO + Segmentation + Auto ML

L'avantage majeur de cette méthode est la réduction maximale des biais liés aux différences de position, de zoom et de rotation entre les images.
Cependant, cette approche présente quelques inconvénients : Tout d'abord, le processus nécessite un nombre important de photos à annoter pour plusieurs modèles, ce qui peut demander des ressources conséquentes en temps et en efforts.
De plus, l'enchaînement de ces modèles peut éventuellement entraîner un temps d'exécution qui dépasse la limite de trois secondes.

5 Évaluation

L'étape suivante du cas d'usage consiste à évaluer le modèle retenu.
Pour ce faire, nous allons estimer les coûts et les gains potentiels que le modèle peut engendrer.
Chaque prédiction est accompagnée de plusieurs codes de retour qui confirment ou infirment les résultats des modèles lancés.

Parmi ces codes de retour, le code `EQP-0` indique que l'image a été classée par le classifieur d'équipement comme étant l'équipement attendu.
Le code `DUP-0` confirme l'absence de duplicata de l'image, tandis que le code `PDP-0` indique la détection d'une nouvelle pose de PTO.
Ces trois codes de retour valident les trois premières phases d'analyse énoncées au début du rapport.

La première étape de l'évaluation consiste à interroger la base de données `BigQuery` afin d'extraire toutes les images prises avant et après une intervention lors d'une journée type en semaine, comme par exemple le 11 mai 2023.
La requête SQL ci-dessous permet d'extraire ces photos :

```sql
SELECT bef.numint, bef.base_gpc, bef.gs_uri, aft.gs_uri, bef.temps_avant, aft.temps_apres
FROM ( SELECT numint, base_gpc, gs_uri,
       FORMAT_DATETIME("%Y-%m-%d %H:%M:%E6S UTC", snap_time) AS temps_avant
       FROM `ofr-ia9-vqi-prd.api_logging.raw_fact_api_analyse`
       WHERE results.conformity_equipment_clf.label = "pto_avant"
       AND conformity_response.duplicated_image_verification.code = "DUP-0"
       AND conformity_response.conformity_equipment_clf.code = "EQP-0"
       AND DATE(created_time) = "2023-05-11"
       ORDER BY created_time DESC
     ) as bef INNER JOIN ( SELECT numint, base_gpc, gs_uri,
                           FORMAT_DATETIME("%Y-%m-%d %H:%M:%E6S UTC", snap_time) AS temps_apres
                           FROM `ofr-ia9-vqi-prd.api_logging.raw_fact_api_analyse`
                           WHERE results.conformity_equipment_clf.label = "pto"
                           AND info_response.pto_info_installation.code = "PDP-0"
                           AND conformity_response.duplicated_image_verification.code = "DUP-0"
                           AND conformity_response.conformity_equipment_clf.code = "EQP-0"
                           AND DATE(created_time) = "2023-05-11"
                           ORDER BY created_time DESC
                         ) as aft ON bef.numint = aft.numint AND bef.base_gpc = aft.base_gpc
```

Avec `numint` l'identifiant unique de chaque intervention, `gs_uri` l'URL de l'image dans le `Cloud Storage`, et `temps_avant` ainsi que `temps_apres` les dates et heures de prise des photos avant et après intervention.
Sur la journée du 11 mai 2023, c'est 2019 paires de photos qui ont renvoyé les codes `EQP-0`, `DUP-0` et `PTO-0`.

Ces 2019 paires ont été concaténées et annotées sur *Label Studio*.
Parmi elles, 1937 ont été étiquetées comme **Same** tandis que 87 ont été étiquetées comme **Not Same**.
Pour évaluer ces 2019 paires de photos, nous avons utilisé les `Samples Requests` de `Online Prediction`.

Voici la matrice de confusion résultante :

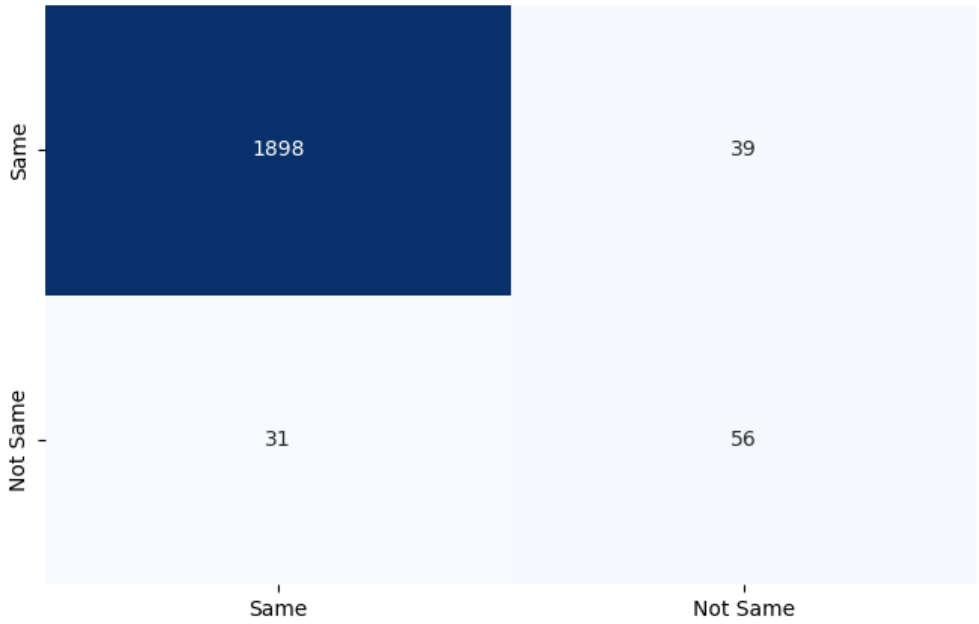


Figure 13 : Matrice de confusion pour l'évaluation de la journée du 11 Mai 2023

On peut observer que 56 paires de photos ont été correctement détectées comme **Not Same**.

Passons maintenant à la deuxième partie de l'évaluation qui implique la recherche du nombre de photos renvoyant les trois codes `EQP-0`, `DUP-0` et `PDP-0` sur une période d'un mois complet.
L'objectif est de généraliser et d'estimer les coûts et les gains potentiels du modèle sur une échelle plus large.

Pour ce faire, nous allons à nouveau requêter `BigQuery` afin de compter le nombre de photos renvoyant ces trois codes pour le mois d'avril 2023.
Voici la requête SQL correspondante :

```
SELECT DATE(created_time) AS d,
SUM(CASE WHEN info_response.pto_info_installation.code = "PDP-0" THEN 1 ELSE 0 END) AS pdp0_count,
FROM `ofr-ia9-vqi-prd.api_logging.raw_fact_api_analyse`
WHERE DATE(created_time) >= "2023-04-01"
AND DATE(created_time) <= "2023-04-30"
AND conformity_response.conformity_equipment_clf.code = "EQP-0"
AND conformity_response.duplicated_image_verification.code = "DUP-0"
GROUP BY d
ORDER BY d
```

Sur le mois d'Avril 2023, un total de 45585 paires d'images ont renvoyé les codes `EQP-0`, `DUP-0` et `PDP-0`.
Grâce au service `Billing` de GCP, il est possible d'explorer les coûts mensuels de chaque service utilisé.
Dans ce contexte, nous nous concentrons spécifiquement sur le coût du classifieur d'équipements, étant donné que nous effectuons également une classification.
En utilisant le coût du classifieur d'équipements pour un mois donné et en prenant en compte les `Node Hours` utilisés en totalité, nous pouvons estimer le coût d'un `Node Hours` pour un modèle de classification d'images.
En effectuant ce calcul, nous trouvons qu'un `Node Hours` coûte environ 1.155 $

Par conséquent, le coût mensuel du modèle *AutoML*, basé sur l'utilisation de 24 `Node Hours` par jour, serait d'environ $1.155 \times 24 \times 30 \simeq 831.60$ \$.

En généralisant le nombre de paires de photos **Not Same** détectées sur un mois, nous obtenons environ $\frac{45585 \times 56}{2019} \simeq 1264$ cas.

En prenant en compte les informations fournies par les métiers, selon lesquelles une réclamation faite à un technicien pour une intervention non conforme permet d'économiser environ 30 \$ à Orange, nous pouvons estimer les économies potentielles générées par le modèle *AutoML*.
Ainsi, l'économie potentielle mensuelle serait d'environ $1264 * 30 = 37931$ \$.

Il est important de noter que cette économie potentielle est bien supérieure au coût mensuel du modèle sur GCP, ce qui indique que le modèle est rentable et exploitable dans ce contexte.

6 Partie MLOps

La partie MLOps (Machine Learning Operations) englobe la gestion, le déploiement et la maintenance des pipelines et des infrastructures nécessaires au développement, au test et à la mise en production des modèles. L'objectif est de mettre en place des pipelines d'entraînement et de déploiement automatisés pour permettre un déploiement rapide et efficace des modèles.
De plus, les tâches récurrentes telles que l'entraînement des modèles, l'évaluation de leurs performances et la gestion des versions seront automatisées afin d'améliorer l'efficacité du processus de développement.

6.1 Vertex Pipelines

Pour rappel, l'outil `Pipelines` de *Vertex AI* est un système de gestion de workflow qui simplifie la mise en œuvre des processus d'apprentissage automatique en automatisant des étapes telles que le prétraitement des données, la formation des modèles, l'évaluation des performances et le déploiement en production.
Le repository `ofr-ia9-vqi-models-automl-master` contient la pipeline automatisée pour les modèles *AutoML*.
Dans ce repository, nous réalisons l'étape d'industrialisation du nouveau modèle, dans laquelle nous apportons des modifications à certains fichiers du repository afin d'y ajouter toutes les informations concernant le nouveau modèle, ces modifications sont effectuées à l'aide de *Git* et *GitLab*.

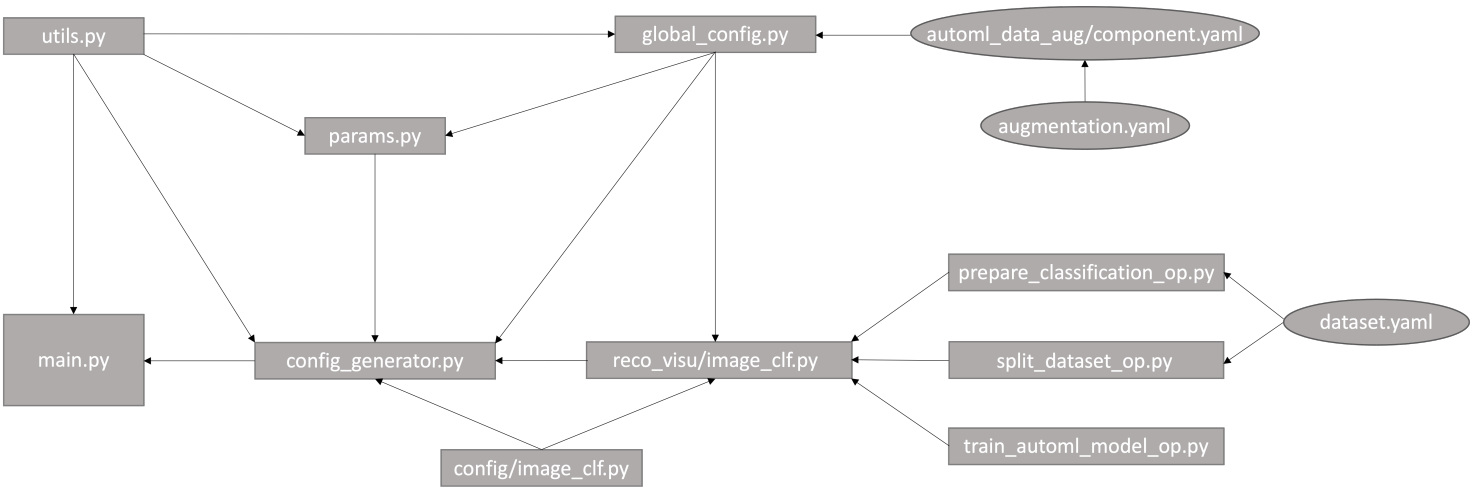


Figure 14 : Architecture global du repository 'ofr-ia9-vqi-models-automl-master'

La base des pipelines du projet VQI repose sur un fichier `csv` nommé `metadata_enriched`, qui contient toutes les données nécessaires pour l'entraînement de l'ensemble des modèles de VQI.
L'objectif de ce fichier est de centraliser toutes les informations relatives aux modèles, ce qui facilite l'exécution des pipelines et la réalisation des entraînements.

Pour intégrer les données de notre modèle *AutoML* de détection de pose de PTO dans le fichier `metadata_enriched`, nous procédons à une mise à jour du fichier.
Afin de conserver l'historique des modifications apportées au `metadata_enriched`, nous ajoutons la date et l'heure au nom du fichier, ce qui garantira son unicité.
De plus, les versions précédentes sont conservées dans le `Cloud Storage`.
Il convient de noter que la version du fichier `metadata_enriched` utilisée pour les pipelines sera toujours la plus récente.
Un aperçu du `metadata_enriched` est disponible via le lien suivant : lien

La construction de la pipeline AutoML implique les composants suivants :

- `prepare-classification-op` est le premier composant, il prend en entrée le `metadata_enriched` et génère en sortie un fichier `csv` nettoyé, nommé `formatted_dataset`.
  Le fichier `formatted_dataset` contiendra les colonnes suivantes : `INTERVENTION_ID_COL` qui est le numéro d'identification unique de l'intervention, `GCS_URI_COL` qui est l'URL de stockage de l'image dans le `Cloud Storage`, `OBJECT_COL` qui est le nom de l'image, `LABEL_COL` qui est le label associé à l'image et enfin dans le cadre du cas d'usage la colonne `PAIRE_PTO_SET_COL` qui est l'ensemble d'appartenance des photos (`TRAIN`, `VALIDATION` et `TEST`).
  Ce composant effectuera une sélection minutieuse des lignes du fichier, en ne retenant que celles qui sont pertinentes pour l'entraînement du modèle spécifique que nous souhaitons entraîner.

- `split-dataset-op` est le deuxième composant, il prend en entré le fichier `formatted_dataset` et génère en sortie un fichier `csv`, nommé `splitted_dataset`.
  Ce composant permet de découper le jeu de donnée en plusieurs ensembles : `TRAIN`, `VALIDATION` et `TEST`.
  Le fichier `splitted_dataset` contiendra des colonnes suivantes : `GCS_URI_COL`, `LABEL_COL` et `SET` qui est l'ensemble d'appartenance de l'image, dans le cadre de ce cas d'usage, les ensembles ont été définis manuellement, donc la colonne `SET` correspond à la colonne `PAIRE_PTO_SET_COL` du fichier `formatted_dataset`.
  Lorsque les ensembles ne sont pas définis manuellement, le code Python du composant `split-dataset-op` prendra en charge cette tâche.
  Le format du fichier `splitted_dataset` est conforme aux exigences de Google pour les entraînements *AutoML*.

- `data-augmentation-composant` est le troisième composant, il prend en entré le fichier `splitted_dataset` et génère en sortie un fichier `csv` nommé, `prepared_dataset`.
  Ce composant est unique car ses codes sont encapsulés dans une image Docker.
  Il est conçu pour effectuer l'augmentation de données sur l'ensemble d'entraînement `TRAIN`.
  Cela inclut des opérations telles que des rotations horizontales ou verticales, des zooms aléatoires, l'ajout de bruit aux images, etc.
  L'objectif est d'accroître la quantité de données d'entraînement et leur diversité, ce qui peut améliorer les performances du modèle.
  Les paramètres spécifiques d'augmentation de données sont ajustables via un fichier au format `yaml`, qui est lié au composant.
  Dans le contexte de ce cas d'usage, l'augmentation de données a été réalisée manuellement pendant la phase d'exploration.
  Ainsi, tous les paramètres du fichier `yaml` ont été fixés à 0 ou neutralisés.
  Le fichier `prepared_dataset` conserve les mêmes colonnes que le fichier `splitted_dataset` mais avec des données supplémentaires dans l'ensemble `TRAIN` en raison de l'augmentation de données appliquée.

- `image-dataset-create` est le quatrième composant, il prend en entrée le fichier `prepared_dataset` afin de créer un jeu de données sur la plateforme `Datasets` de *Vertex AI*.
  À cet effet, ce composant fait usage de la fonction `ImageDatasetCreateOp` [9] de la librairie *aiplatform* de Google.
  Cette fonction est conçue spécifiquement pour créer le jeu de données souhaité sur la plateforme *Vertex AI*.

- `train-automl-model-op` est le dernier composant, il prend en entrée le jeu de données crée grâce au composant `image-dataset-create` puis initie un processus d'entraînement AutoML au sein de la fonctionnalité `Training` de *Vertex AI*.
  Dans ce composant, la fonction clé utilisée est `AutoMLImageTrainingJob` [8] de la libraire *aiplatform* de Google.
  Cette fonction est spécialement conçue pour lancer des tâches d'entraînement *AutoML* sur la plateforme *Vertex AI*.

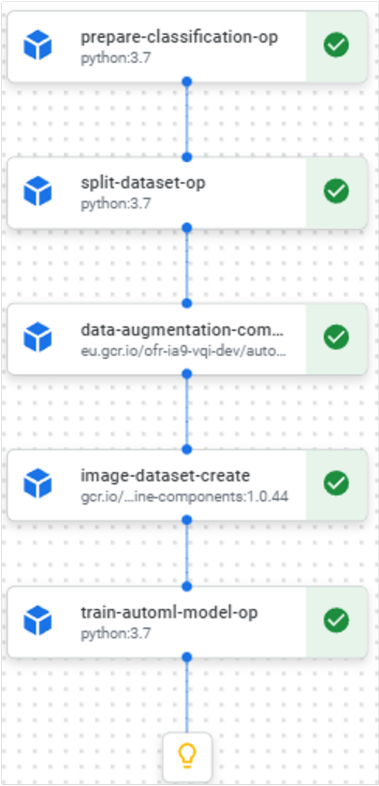


Figure 15 : Schéma de la Pipeline AutoML du modèle de détection de pose de PTO sur Vertex Pipelines

Les modifications apportées au repository `ofr-ia9-vqi-models-automl-master` concernent principalement les fichiers `prepare-classification-op` et `split-dataset-op` ainsi que les fichiers de configuration globaux annexes.

6.2 Git et GitLab

La plateforme principale utilisée est *GitLab* [11], une plateforme qui permet à l'équipe de collaborer efficacement, de suivre les versions, de tester et de déployer des applications de manière cohérente.
Les fichiers du projet VQI sont tous stockés sur *GitLab* dans des repository spécifiques en fonction de leur contenu.
Sur GCP, il existe trois projets VQI distincts : le projet `dev` (développement), où sont réalisées les explorations et les premiers entraînements des modèles, le projet `ppr` (pré-production), qui est utilisé pour effectuer divers tests de Data Engineering (CI/CD) concernant les modèles afin de confirmer leur bon fonctionnement, et enfin le projet `prd` (production), qui contient les modèles en production.

On focalise notre attention sur un repository nommé `ofr-ia9-vqi-models-automl-master`, car c'est là que nous allons intégrer le modèle *AutoML* retenu.
Ce repository contient deux branches principales : la branche `master`, qui contient la version la plus récente et à jour du repository et qui est associée au projet `ppr` sur GCP, et la branche `dev`, qui possède les mêmes fichiers que la branche `master` et sert de base pour la création de la nouvelle pipeline automatisée.
La branche `dev` est liée au projet `dev` sur GCP.
L'objectif est de créer une nouvelle branche appelée `Paire-PTO` à partir de la branche `dev`, cette nouvelle branche sera utilisée pour apporter les modifications nécessaires afin d'intégrer le nouveau modèle.
Une fois les modifications effectuées et validées, elles seront fusionnées à la branche principale pour mettre à jour le repository dans son ensemble.

Pour accomplir cela, j'ai utilisé l'environnement de développement intégré (IDE) *Visual Studio Code* [5] (VSC) ainsi que l'outil de contrôle de version *Git* [10].
Grâce à un token et une clé SSH, ainsi qu'à l'extension *GitLab Workflow* [32] de VSC, on peut établir une connexion avec le dépôt GitLab du projet VQI.
Cela m'a permis de cloner le repository directement en local sur VSC en utilisant la commande `git clone` dans un terminal Bash, puis d'apporter les modifications nécessaires.
On se place dans la branche `Paire-PTO` à l'aide de la commande `git checkout Paire-PTO`.
Une fois les modifications apportées et l'intégration du nouveau modèle effectuée dans les différents fichiers, on utilise la commande `git add *` pour préparer mes modifications, puis on effectue un commit avec la commande `git commit -m` pour appliquer les modifications à la branche Paire-PTO.
On ajoute un commentaire précis lors du commit pour documenter en détail les modifications effectuées et ainsi conserver une trace de l'historique des changements.
Enfin, on procède à un push en utilisant la commande `git push origin Paire-PTO`, ce qui permet d'envoyer les fichiers contenant les modifications vers *GitLab* dans la branche `Paire-PTO`.
Cette action a eu pour effet de mettre à jour la branche avec les changements effectués.

6.3 Processus de mise en production

La dernière partie du cas d'usage consiste à mettre en production le nouveau modèle.
Pour rappel, la branche `Paire-PTO` est une sous branche de la branche `dev`.
Après avoir modifié, mis à jour les fichiers de la pipeline Auto ML et appliqué un push vers *GitLab*, on peut passer à l'étape suivante : faire une `Merge Request`.
Une `Merge Request` (MR) est une fonctionnalité de *GitLab* qui permet de proposer des modifications de code à fusionner depuis une branche source vers une branche cible.
Les membres de l'équipe peuvent examiner le code, commenter, suggérer des modifications et approuver la fusion.
La branche source est `Paire-PTO` et la branche cible est `dev`.
Une fois la MR approuvée, on effectue la fusion de la branche `Paire-PTO` vers la branche `dev`, cela aura pour conséquence de mettre à jour la branche `dev` avec les nouvelles modifications.

La fusion effectuée déclenchera également automatiquement le déploiement de la pipeline dans le système de gestion `Pipelines` de GCP (i.e figure 14) du projet `dev`.
Par ailleurs, il est possible de lancer une pipeline en utilisant un terminal shell et un fichier `Makefile` contenu dans le repository `ofr-ia9-vqi-models-automl-master`.
Lors du lancement de la pipeline, il est possible de choisir de ne pas exécuter le composant d'entraînement (`train-automl-model-op`).
Cette option permet non seulement de vérifier le bon fonctionnement des autres composants de la pipeline, mais aussi d'éviter de lancer un entraînement long et onéreux (environ 7 heures et 145 $) en cas d'erreur ou de besoin de validation préliminaire.
Après l'exécution réussie de la pipeline sur le projet `dev`, il est essentiel de vérifier le bon déroulement de la pipeline dans la section `Pipelines` et d'évaluer les performances du modèle dans la section `Model Registry`.
Une fois cette vérification effectuée, on peut initier une `Merge Request` pour fusionner la branche `dev` vers la branche `master`.
Une fois cette merge approuvée et effectuée, la pipeline sera déployée dans le projet `ppr` sur GCP, permettant ainsi le déploiement du nouveau modèle dans l'environnement de pré-production.

Il est important de noter que les images stockées dans le `Cloud Storage` ainsi que le `metadata_enriched` doivent être transférées manuellement en utilisant *Google Cloud SDK* d'un projet GCP à l'autre, car ces projets sont indépendants et ne sont pas directement connectés.
Après l'exécution de la pipeline dans l'environnement `ppr`, il est crucial de procéder à une vérification approfondie du bon fonctionnement de la pipeline, des performances du modèle et de la validation de tous les tests automatiques.
Une fois cette vérification effectuée et confirmée, on passe au lancement de la mise en production du modèle dans le projet `prd`.
Pour cela, on crée un `Tag` sur *GitLab* à partir de la branche `master`, en le nommant avec un nouveau numéro de version en prenant en compte les anciennes versions.
Ensuite, on exécute ce `Tag`, ce qui lancera le déploiement de la pipeline dans le projet `prd`.
Après que cette tâche soit terminée et soigneusement vérifiée, le modèle est déployé en production dans le projet `prd`, prêt à être utilisé pour répondre aux besoins opérationnels.

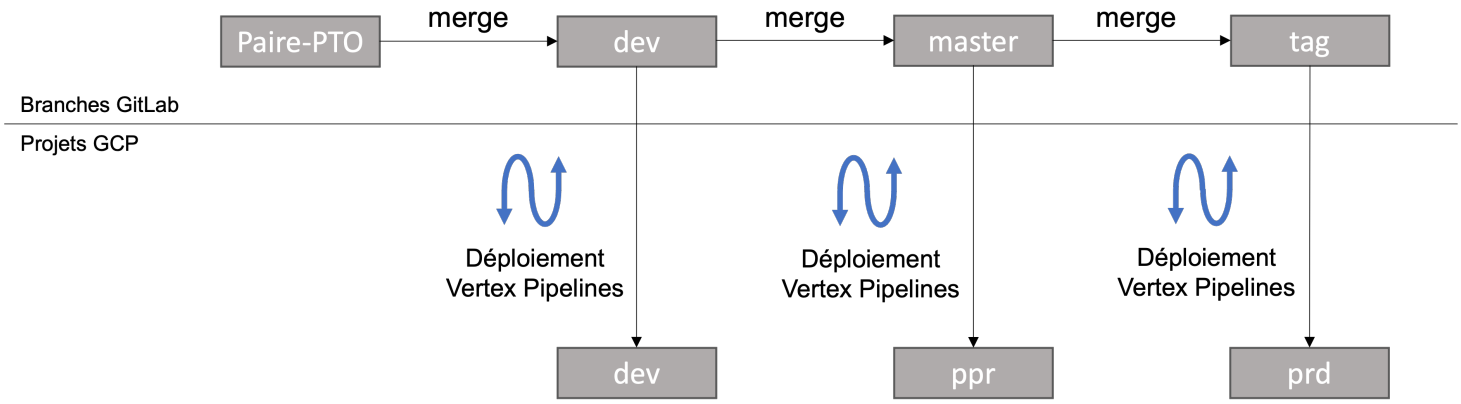


Figure 16 : Architecture du processus de mise en production du modèle

7 Conclusion

Ce stage m'a apporté une expérience très enrichissante, tant sur le plan professionnel que personnel.
Mon sujet de stage m'a permis de mener un projet concret au sein de l'entreprise, me donnant l'occasion de suivre l'intégralité du processus, de l'exploration à la mise en production.

Chaque étape du processus a été minutieusement explorée en détail, ce qui m'a permis de comprendre en profondeur son fonctionnement ainsi que sa logique.
La phase d'exploration s'est avérée particulièrement instructive, me permettant d'affiner mes compétences en recherche d'information, d'exploration de diverses méthodes et de découverte d'algorithmes novateurs et performants.
Cette approche permet de rester constamment à jour et être informé des dernières avancées dans un domaine en constante évolution.

L'exploration de toutes ces méthodes m'a permis de constituer un solide recueil de connaissances.
Par exemple, les méthodes qui n'ont pas été retenues pour mon cas d'usage pourraient se révéler utiles dans d'autres projets.
Le fait de les connaître me confère un avantage certain, renforçant ma polyvalence et ma compréhension approfondie du domaine.

Les étapes MLOps et de mise en production m'ont permis de comprendre le fonctionnement d'un projet en équipe.
En effet, en travaillant avec d'autres corps de métier tels que les Data Engineers et un Proxy Product Owners, cela a renforcé mes compétences en travail d'équipe ainsi que ma capacité à communiquer efficacement.
Cette communication est très importante, car elle favorise une collaboration efficace et assure le transfert fluide du travail entre les membres de l'équipe, lorsque cela s'avère nécessaire.
C'est pourquoi nous tenons un daily pour discuter des avancées individuelles et des prochaines étapes à venir.
Cette approche garantit une synchronisation optimale et contribue à une bonne productivité.

D'un point de vue personnel, le travail en entreprise m'a permis d'apprendre la culture de l'entreprise, travailler en open-space favorise la communication entre collègues et cela peut être très utile lorsque l'on a une question précise sur un sujet, le fait de connaître le rôle de chacun peut nous aider dans nos propres projet.
D'un point de vue personnel, le travail en entreprise m'a permis de découvrir de la culture de celle-ci.
Travailler en open-space favorise la communication entre collègues, ce qui se révèle particulièrement utile lorsqu'une question portant sur un sujet précis se pose.
L'environnement de travail est également important, j'ai eu la chance de réaliser mon stage aux côtés de personnes très à l'écoute, toujours présentes et pédagogues et cette expérience m'a aidé à gagner en confiance en moi.
Le fait de devoir constamment prendre la parole lors des réunions d'équipe m'a permis de me sentir plus à l'aise à l'oral devant un auditoire.

Enfin, travailler en entreprise est très différent de la fac, le travail est très concret et permet véritablement d'acquérir des connaissances en mettant en pratique ce que l'on a appris dans des situations réelles.
C'est à l'opposé de la fac où l'on se limite souvent à la théorie et à des exemples sur des données simulées et idéales, ce qui ne correspond pas à la réalité du terrain.

References

[1] Vertex AI. *Google*. URL: https://cloud.google.com/vertex-ai/docs/start/introduction-unified-platform?hl=fr.

[2] Format AutoML. *Google*. URL: https://cloud.google.com/natural-language/automl/docs/prepare?hl=en.

[3] Billing. *Google*. URL: https://cloud.google.com/billing/docs/concepts?hl=fr.

[4] Tianqi Chen and Carlos Guestrin. "XGBoost: A Scalable Tree Boosting System". In: *CoRR* abs/1603.02754 (). arXiv: 1603.02754. URL: http://arxiv.org/abs/1603.02754.

[5] Visual Studio Code. *Visual Studio Code*. URL: https://code.visualstudio.com/docss.

[6] Réseau neuronal convolutif. *Wikipédia*. URL: https://code.visualstudio.com/docss.

[7] ExtraTreesClassifier. *Scikit Learn*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.ExtraTreesClassifier.html.

[8] AutoMLImageTrainingJob function. *Google*. URL: https://cloud.google.com/python/docs/reference/aiplatform/latest/google.cloud.aiplatform.AutoMLImageTrainingJob.

[9] ImageDatasetCreateOp function. *Google*. URL: https://google-cloud-pipeline-components.readthedocs.io/en/google-cloud-pipeline-components-2.0.0/api/v1/dataset.html#v1.dataset.ImageDatasetCreateOp.

[10] Git. *Git*. URL: https://git-scm.com/docs.

[11] GitLab. *GitLab*. URL: https://docs.gitlab.com.

[12] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *CoRR* abs/1512.03385 (). arXiv: 1512.03385. URL: http://arxiv.org/abs/1512.03385.

[13] Imagenet. *Imagenet*. URL: https://www.image-net.org/about.php.

[14] Guolin Ke et al. "LightGBM: A Highly Efficient Gradient Boosting Decision Tree". In: (). URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf.

[15] Scikit Learn. *Scikit Learn*. URL: https://scikit-learn.org/stable/user_guide.html.

[16] Segment Anything Model. *Meta*. URL: https://github.com/facebookresearch/segment-anything.

[17] Google Cloud Platform. *Google*. URL: https://cloud.google.com/?hl=fr.

[18] PyTorch. *PyTorch*. URL: https://pytorch.org/docs/stable/index.html.

[19] Big Query. *Google*. URL: https://cloud.google.com/bigquery/docs/introduction?hl=fr.

[20] RidgeClassifier. *Scikit Learn*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.RidgeClassifier.html.

[21] Mark Sandler et al. "Inverted Residuals and Linear Bottlenecks: Mobile Networks for Classification, Detection and Segmentation". In: *CoRR* abs/1801.04381 (). arXiv: 1801.04381. URL: http://arxiv.org/abs/1801.04381.

[22] Paul-Edouard Sarlin et al. "SuperGlue: Learning Feature Matching with Graph Neural Networks". In: *CoRR* abs/1911.11763 (). arXiv: 1911.11763. URL: https://arxiv.org/pdf/1911.11763.pdf.

[23] Cloud SDK. *Google*. URL: https://cloud.google.com/sdk/docs/install?hl=fr.

[24] Karen Simonyan and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: (). arXiv: 1409.1556. URL: https://arxiv.org/abs/1409.1556.

[25] Cloud Storage. *Google*. URL: https://cloud.google.com/storage/docs/discover-object-storage-console?hl=fr.

[26] Label Studio. *Label Studio*. URL: https://labelstud.io/templates/.

[27] SVC. *Scikit Learn*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html.

[28] Mingxing Tan and Quoc V. Le. "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks". In: *CoRR* abs/1905.11946 (). arXiv: 1905.11946. URL: http://arxiv.org/abs/1905.11946.

[29] TensorFlow. *TensorFlow*. URL: https://www.tensorflow.org/learn?hl=fr.

[30] torchvision. *PyTorch*. URL: https://pytorch.org/vision/stable/index.html.

[31] UNet. *UNet*. URL: https://github.com/milesial/Pytorch-UNet.

[32] GitLab WorkFlow. *GitLab*. URL: https://marketplace.visualstudio.com/items?itemName=GitLab.gitlab-workflow.

[33] YOLOv5. *Ultralytics*. URL: https://github.com/ultralytics/yolov5.