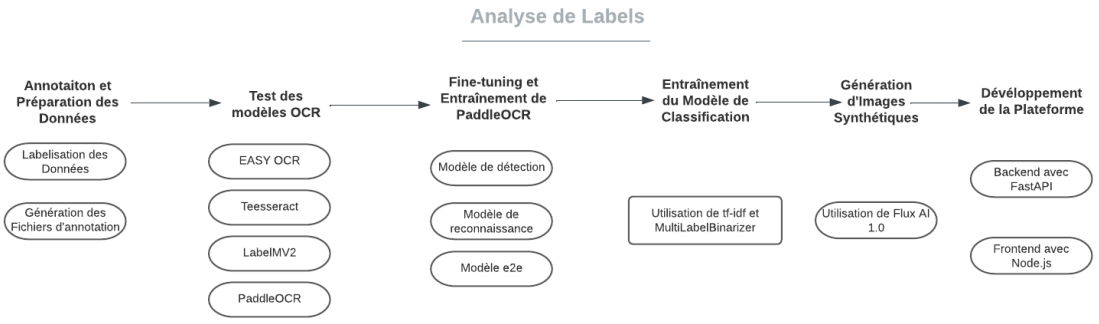
Rapport de Finetuning et Déploiement d’un Modèle OCR avec PaddleOCR

Introduction

L'objectif de ce projet était de créer un modèle prenant en entrée l’image de la plaque signalétique afin de renvoyer en sortie les 4 informations clés : Le constructeur, le numéro de modèle, le numéro de série et la date de fabrication. Pour atteindre cet objectif, nous avons d’abord pensé à utiliser les modèles les plus simples et les plus performants en matière d’OCR, notamment EASY OCR, tesseract LMV2Layout avec detectron et PaddleOCR. Les résultats seront discutés en détails au fil de notre rapport, on a procédé après à utiliser des modèles plus complexes, le modèle PaddleOCR étant celui le plus net , nous avons procédé à fine-tuner les différents modèles disponibles avec toutes les étapes nécessaires pour ce process. La phase de détection étant aboutie et améliorée, il était question d’extraire les données reconnues et les classifier en utilisant des méthodes NLP pour trouver les 4 éléments nécessaires. Pour enrichir la base de données qui était vraiment très petite, nous avons aussi penser à un modèle de stable diffusion et génération d’images : Flex Ai 1.0.



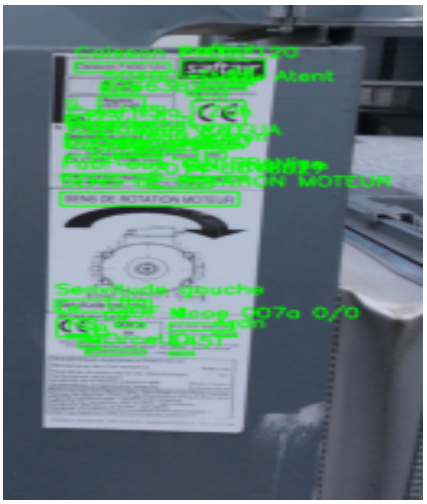
1. Comparaison des modèles

Dans le cadre de ce projet, plusieurs modèles de reconnaissance optique de caractères (OCR) ont été testés pour identifier celui qui conviendrait le mieux à la tâche d'extraction d'informations sur les plaques signalétiques. **EASYOCR** a été l'une des premières solutions explorées. Ce modèle est simple à utiliser et fonctionne rapidement pour des tâches basiques, mais il a montré des limitations en termes de précision sur des textes industriels complexes. Ensuite, **Tesseract**, bien connu pour sa robustesse, a été testé. Il a montré des performances intéressantes en matière de détection de texte, notamment sur des images avec des caractères petits ou peu contrastés. Cependant, Tesseract avait tendance à "sur-détecter" en identifiant des éléments non pertinents ou en confondant des lignes de texte, ce qui a nécessité un post-traitement conséquent pour filtrer les résultats.

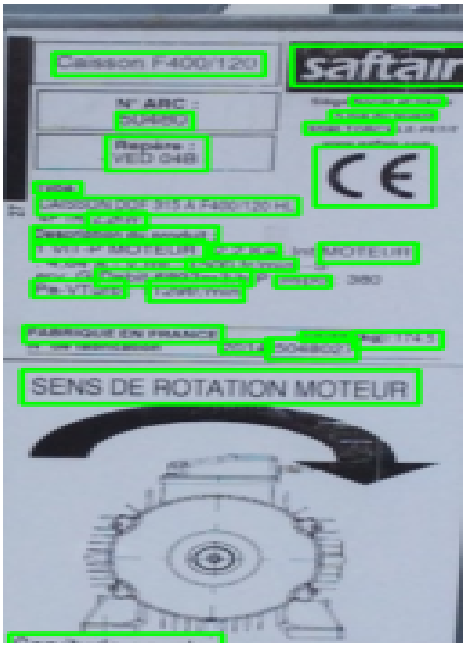
Le modèle **LabelMVv2** (Layout LMv2 avec Detectron) a été testé pour sa capacité à reconnaître non seulement le texte mais aussi les structures de documents, mais il s'est avéré trop complexe pour le type d'images industrielles de ce projet.

Enfin, **PaddleOCR** s'est avéré être le plus adapté pour nos besoins. PaddleOCR offre un bon compromis entre simplicité et performance. Il détecte avec précision les informations cruciales tout en évitant les surcharges de détection, ce qui le rend particulièrement efficace pour des images complexes mais structurées comme celles des plaques signalétiques. Grâce à sa flexibilité et aux possibilités de fine-tuning, PaddleOCR a été retenu pour l'entraînement final du modèle.

Les captures suivantes présentent les différents tests réalisés avec chacun de ces modèles, illustrant leurs performances respectives et les raisons du choix final.



Tesseract



Easy OCR

1: Schneider Ref: NSYS3D7525 0.926
2: Schneider Ref: NSYS3D7525 0.926
3: SARLL 0.933
4: SARLL 0.933
5: SElectric 0.982
6: SElectric 0.982
7: A-868 0.986
8: A-868 0.986
9: FR 0.997
10: FR 0.997
11: US 0.837
12: US 0.837
13: INDUBTRIAL CONTROL PANELENCLOSURE 0.936
14: INDUBTRIAL CONTROL PANELENCLOSURE 0.936
15: LISTED 0.988
16: LISTED 0.988
17: Enclosure Type:1,2,3.3R,44X.5,12and13. 0.823
18: Enclosure Type:1,2,3.3R,44X.5,12and13. 0.823
19: matal the cmronmental typ
e ralng.all condult fnge and other componenle 0.796
20: matal the cmronmental typ
e ralng.all condult fnge and other componenle 0.796
21: ed In opeelngs made In the
enelosure muel have le same ratlnge and only wall 0.806
22: ed In opeelngs made In the
enelosure muel have le same ratlnge and only wall 0.806
23: e12 0.617
24: e12 0.617
25: 4mm (1/e-114in) diameler i
n ihe ioweet parl of thebottom wall. 0.794
26: 4mm (1/e-114in) diameler i
n ihe ioweet parl of thebottom wall. 0.794
27: 1436 0.985
28: 1436 0.985
29: 17878734 0.997
30: 17878734 0.997

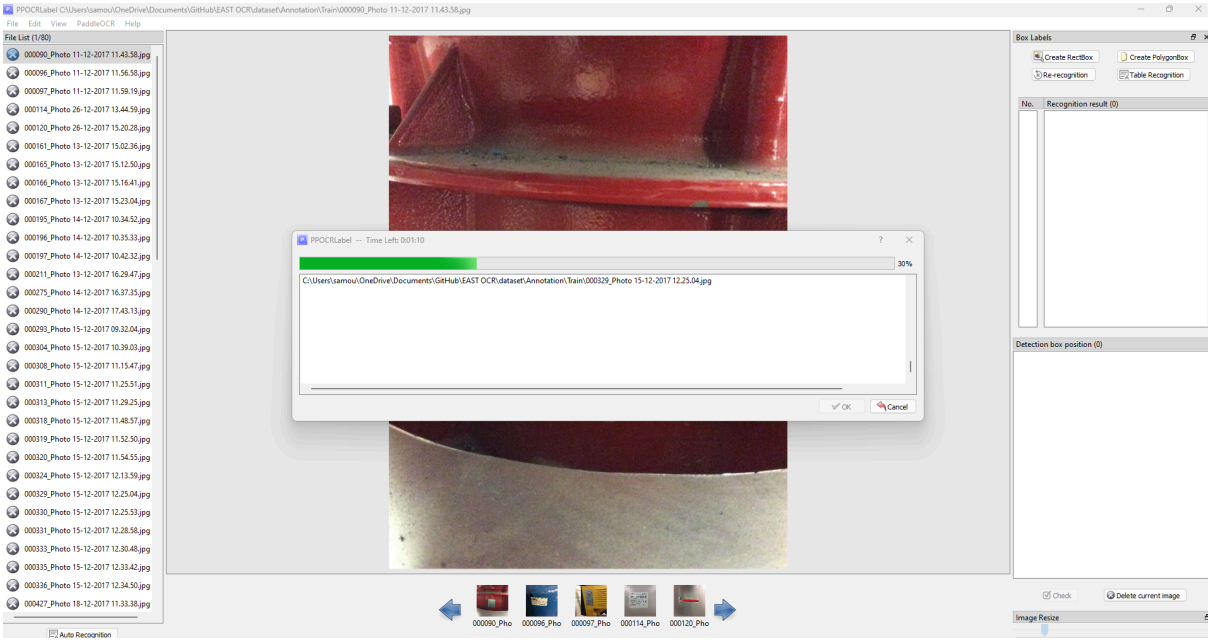
Paddle OCR

2. **Préparation des Données :**

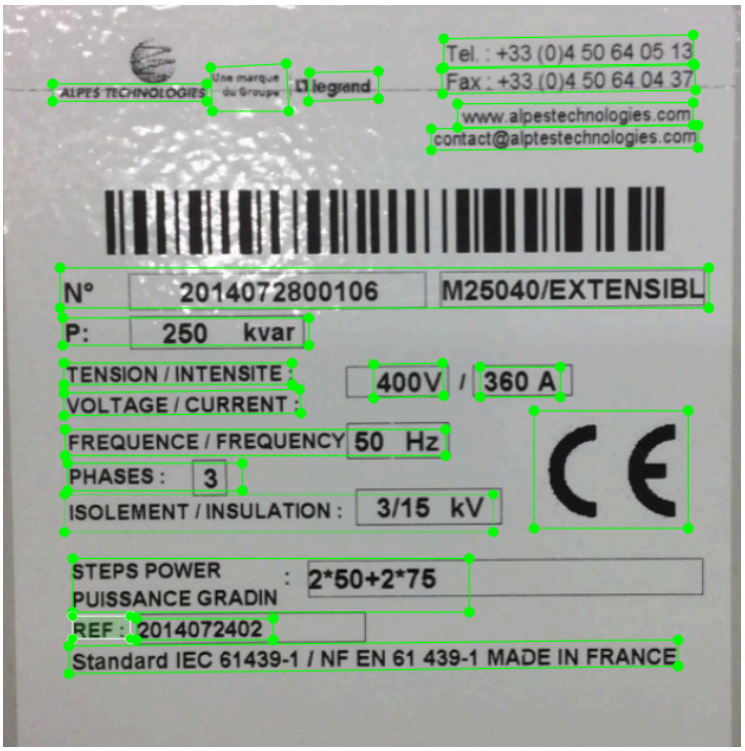
Pour annoter les données nécessaires à l'entraînement de notre modèle OCR, nous avons utilisé **PPOCRLabelv2**, un outil d'annotation semi-automatique spécialement conçu pour le domaine de l'OCR. **PPOCRLabelv2** est une bibliothèque en Python, basée sur PyQT5, qui permet d'effectuer des annotations graphiques sur des images. L'outil intègre le modèle **PP-OCR** pour détecter automatiquement les zones de texte et reconnaître leur contenu. Il prend en charge plusieurs modes d'annotation, notamment pour les boîtes rectangulaires, les tables, les textes irréguliers et les informations clés.

L'utilisation de PPOCRLabel est simple et intuitive. L'outil ouvre une fenêtre où les images sont chargées, avec des détections automatiques affichées sous forme de boîtes rectangulaires. Ces boîtes contiennent les transcriptions détectées par **PaddleOCR**. L'utilisateur a alors la possibilité de redessiner les rectangles de détection, de corriger ou d'ajuster le contenu de la transcription, puis de sauvegarder ces annotations. Toutes les annotations sont ensuite exportées dans un fichier **label.txt**, contenant les coordonnées des boîtes ainsi que les textes correspondants. Ce fichier sert directement de jeu de données pour l’entraînement des modèles de détection et de reconnaissance de PP-OCR.

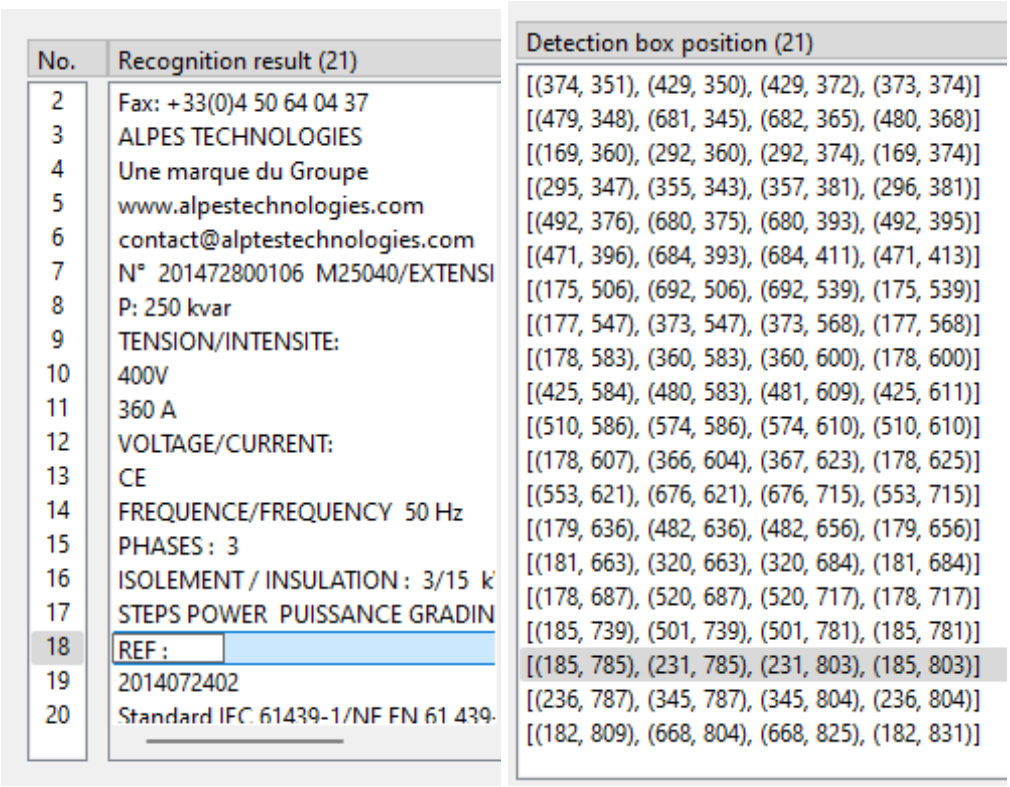
Grâce à PPOCRLabel, nous avons pu enrichir notre base de données avec des annotations précises et adaptées, facilitant ainsi le fine-tuning du modèle et l’amélioration des résultats.



PPOCRLabel en détection



Recadrage customisé



Modification des Labels et transcriptions

Voici un exemple de la syntaxe du fichier Label.txt :

```
'000090_Photo 11-12-2017 11.43.58.jpg
[
  {
    "transcription": "P8.7kW",
    "points": [[476,635],[498,635],[498,738],[ 476,738]],
    "difficult": false},
  {
    "transcription": "nmin 380.min-1",
    "points": [[498,628],[516,634],[508,746],[494,741]],
    "difficult": false
  },
```

3. Entraînement et Fine-Tuning du Modèle :

PaddleOCR propose plusieurs façons d'entraîner et de fine-tuner des modèles OCR selon l'objectif visé, qu'il s'agisse de la détection de texte, de la reconnaissance ou d'un modèle end-to-end (E2E). Voici un aperçu des différentes approches possibles pour entraîner ou adapter ces modèles :

a. Entraînement d'un Modèle de Détection

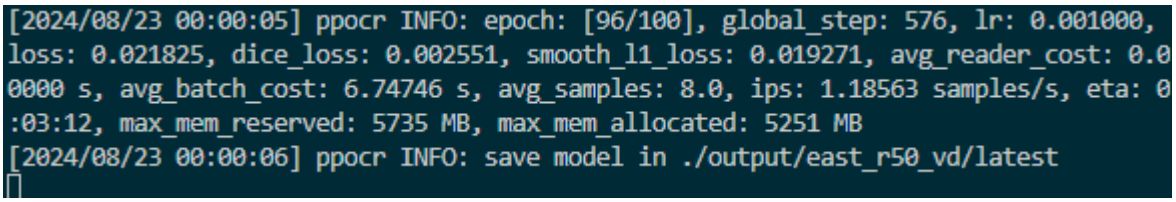
La première étape consiste à détecter les zones de texte dans une image. PaddleOCR offre plusieurs algorithmes pour ce faire, comme EAST, DB et SAST. Pour configurer l'entraînement d'un modèle de détection, vous devez préparer les annotations associées aux images. Ces annotations incluent les coordonnées des boîtes englobantes des textes. Une fois les données prêtes, un fichier de configuration YAML permet de spécifier les hyperparamètres, le modèle pré-entraîné utilisé, les transformations des images, etc.

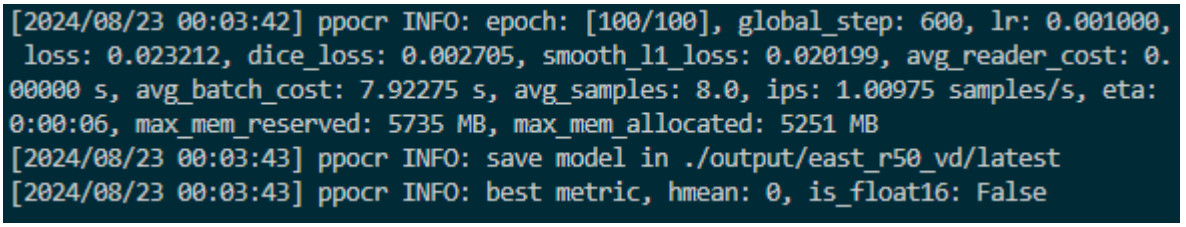
Un exemple de configuration typique pourrait inclure la définition des paramètres tels que :

- **Prétraitement des images** : redimensionnement, normalisation, etc.
- **Transformations pour augmenter la robustesse du modèle** : rotation, étirement.
- **Chemin des données d'entraînement et d'évaluation**.
- **Chemin du modèle pré-entraîné** (si applicable pour le fine-tuning).

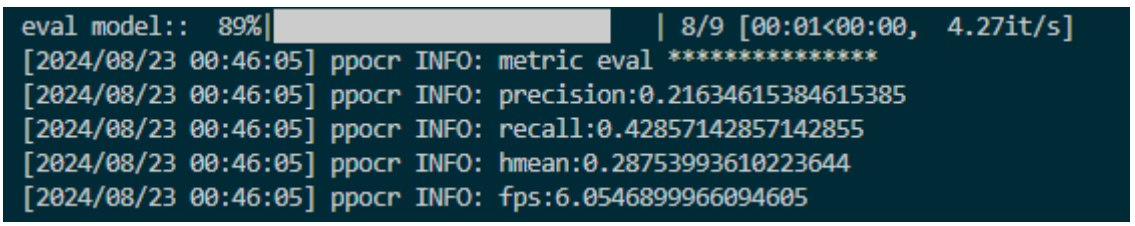
La commande d'entraînement ressemblerait à :

```
$ python tools/train.py -c configs/det/det_mv3_east.yml
```

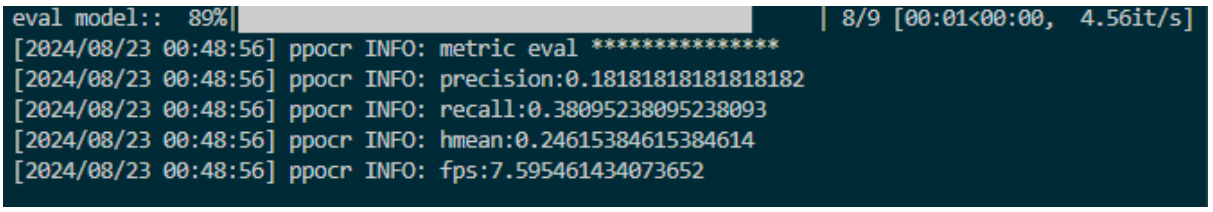




eval resnet



eval mobile:



b. Entraînement d'un Modèle de Reconnaissance

Une fois les textes détectés, l'étape suivante est de les reconnaître. PaddleOCR permet d'entraîner des modèles de reconnaissance sur différentes langues et jeux de caractères. Ici encore, un fichier de configuration YAML guide l'entraînement, précisant les chemins d'accès aux données, les paramètres des modèles, les dictionnaires de caractères, etc.

Les principaux points de configuration incluent :

- **Chemin des images d'entraînement et d'évaluation**.
- **Fichier de dictionnaire contenant les caractères à reconnaître**.
- **Modèle pré-entraîné pour le fine-tuning**.

L'entraînement d'un modèle de reconnaissance peut être lancé avec une commande comme :

```
$ python tools/train.py -c configs/rec/rec_mv3_none_bilstm_ctc.yml
```

c. Entraînement d'un Modèle End-to-End (E2E)

PaddleOCR propose également des solutions pour les scénarios où la détection et la reconnaissance sont effectuées conjointement. Le modèle PGNet (Prototypical Graph Neural Network) est un exemple d'approche end-to-end. Ce modèle simplifie l'ensemble du processus en détectant les textes et en les reconnaissant simultanément, réduisant ainsi la complexité et l'optimisation séparées.

La configuration pour un modèle end-to-end spécifie :

- **Algorithmes de détection et de reconnaissance intégrés**.
- **Chemins des données** et modèles pré-entraînés.
- **Paramètres pour ajuster le réseau en fonction des types de textes et des longueurs maximales à reconnaître**.

Une commande pour entraîner un modèle E2E pourrait ressembler à ceci :

```
$ python tools/train.py -c configs/e2e/e2e_pgnet.yml
```

● Fine-Tuning des Modèles

Le fine-tuning s'effectue de manière similaire à l'entraînement classique, mais en important un modèle pré-entraîné comme point de départ. Le fichier de configuration YAML doit inclure

le chemin du modèle pré-entraîné, ainsi que les hyperparamètres spécifiques au domaine d'application ciblé.

Le fine-tuning est particulièrement utile pour adapter un modèle généraliste à un ensemble de données plus spécifique (par exemple, des étiquettes industrielles) en optimisant ses performances sur ces types d'images.

- ## Exécution et Évaluation

À chaque étape de l'entraînement, il est possible de surveiller les métriques de performance, telles que la précision de détection (IoU) ou le score hmean pour la reconnaissance. Une fois l'entraînement terminé, le modèle peut être utilisé pour l'inférence en batch ou en temps réel.

L'évaluation peut également être configurée pour s'exécuter périodiquement pendant l'entraînement, ce qui permet d'ajuster les paramètres de manière dynamique.

- ## Conclusion

Grâce à sa flexibilité et à ses algorithmes performants, PaddleOCR permet une adaptation à divers cas d'utilisation. Que ce soit pour la détection, la reconnaissance ou les systèmes end-to-end, PaddleOCR fournit une suite d'outils et de modèles pour répondre à une large gamme de besoins en OCR.

4. Pipeline de Détection et Classification :

Le processus de classification a été conçu pour identifier les quatre éléments clés sur les plaques signalétiques : le constructeur, le modèle, le numéro de série, et la date de fabrication. Le pipeline se déroule en plusieurs étapes :

1. **Préparation des Données :** Les données ont été initialement chargées à partir de fichiers CSV et TXT. Les fichiers TXT contiennent les transcriptions des plaques, tandis que les fichiers CSV contiennent des informations structurées comme le chemin de l'image et les annotations associées. Une correspondance est faite entre les chemins des images et les transcriptions pour enrichir le jeu de données.
2. **Prétraitement des Données :** Le texte est prétraité pour améliorer la performance du modèle de classification. Ce prétraitement inclut :
    - Conversion du texte en minuscules.
    - Suppression des caractères spéciaux et non pertinents.
    - Tokenisation du texte (découpage en mots).
    - Suppression des "stop words" (mots non significatifs comme "le", "et", etc.).
    - Reconstruction du texte prétraité.

3. **Création du Modèle :** Le modèle de classification repose sur une combinaison de TfidfVectorizer pour l'extraction des caractéristiques textuelles, et un classifieur SVM multiclasses (OneVsRestClassifier avec LinearSVC). Le TfidfVectorizer permet de transformer les textes en représentations numériques, tandis que le classifieur SVM gère la prédiction des étiquettes.
4. **Entraînement du Modèle :** Le modèle est entraîné en utilisant les données textuelles prétraitées et les annotations associées (constructeur, modèle, numéro de série, date). Les données sont divisées en ensembles d'entraînement et de test pour évaluer les performances du modèle.
5. **Évaluation du Modèle :** Une évaluation est réalisée à la fois sur l'ensemble de test et sur un ensemble de validation séparé. Le rapport de classification fournit des métriques détaillées sur la précision du modèle pour chaque catégorie.
6. **Sauvegarde du Modèle :** Les objets essentiels comme le vectorizer, le classifieur, et le binariseur multiclasse sont sauvegardés pour une utilisation ultérieure dans le processus d'inférence.

L'ensemble de ce pipeline permet de passer efficacement des transcriptions textuelles brutes à la classification précise des informations présentes sur les plaques signalétiques. Cela constitue une étape clé dans l'automatisation du processus de reconnaissance de texte et d'extraction d'informations.

Génération d'Images pour l'Enrichissement de la Base de Données :

Pour pallier le manque de données disponibles, nous avons intégré une étape de génération d'images basée sur un modèle de diffusion pour augmenter notre base de données. Nous avons utilisé un pipeline de génération d'images basé sur **Stable Diffusion**, via le modèle *FluxPipeline* proposé par Hugging Face.

L'objectif de cette étape était de générer des images réalistes de plaques signalétiques ou d'étiquettes industrielles, comportant des informations techniques comme les numéros de modèle, les numéros de série, les dates de fabrication ou encore les avertissements de sécurité.

Dans notre code, nous avons utilisé des configurations adaptées pour générer des images tout en optimisant les ressources disponibles (comme le GPU). Le modèle utilisé est basé sur une architecture de diffusion, capable de générer des images de haute qualité à partir de descriptions textuelles détaillées (prompts).

Justification des Paramètres

- **prompt** : Ce texte décrit précisément ce que nous souhaitons voir dans l'image générée.
- **height et width** : Taille réduite pour accélérer le processus de génération.
- **guidance_scale** : Un facteur influençant l'équilibre entre la fidélité à la description textuelle et la créativité dans la génération.
- **num_inference_steps** : Nombre d'étapes de diffusion, ajusté pour un compromis entre la qualité de l'image et le temps d'exécution.
- **generator** : Le seed permet de contrôler la reproductibilité des résultats.

Cette approche nous a permis de simuler des scénarios réalistes, générant ainsi des images supplémentaires pour améliorer l'entraînement et la robustesse du modèle OCR. Ces images ont été ensuite annotées pour être utilisées dans les phases de fine-tuning et d'évaluation.

Développement de la Plateforme

La plateforme développée dans ce projet sert d'interface utilisateur permettant de tester les différents modèles OCR, consulter les résultats de génération d'images et gérer l'ensemble des données générées. Elle a été construite principalement en utilisant **FastAPI** pour le backend, et **React.js** pour le frontend.

1. Structure du Projet

La plateforme est divisée en plusieurs modules, chacun ayant une fonction bien définie :

- **Backend (FastAPI)** : Gère les requêtes API, traite les images via les modèles OCR et génère les résultats sous forme de texte.
- **Frontend (React.js)** : Fournit une interface utilisateur intuitive permettant de charger des images, lancer des tests OCR, et visualiser les résultats.
- **Communication entre les modules** : L'API du backend expose des endpoints que le frontend utilise pour exécuter les différentes fonctionnalités, comme l'envoi d'images à analyser ou la récupération des résultats OCR.

2. Démonstration des Fonctionnalités

La plateforme permet d'effectuer les tâches suivantes :

- **Tester les modèles OCR** : Les utilisateurs peuvent charger une image, choisir un modèle OCR parmi ceux proposés (EASY OCR, Tesseract, PaddleOCR, etc.) et visualiser les résultats.
- **Consulter les résultats de la génération d'images** : La plateforme permet d'examiner les images générées par le modèle de diffusion et d'évaluer leur pertinence.
- **Annotation et gestion des données** : Un module intégré permet de vérifier les annotations générées automatiquement et de les corriger si nécessaire.

3. Instructions d’Utilisation

Les instructions détaillées pour installer et exécuter la plateforme seront partagées dans un dépôt GitHub une fois que celui-ci sera public. Cela inclura les étapes d’installation des dépendances, le lancement du serveur backend, et l’utilisation du frontend. Pour l’instant, voici un aperçu rapide :

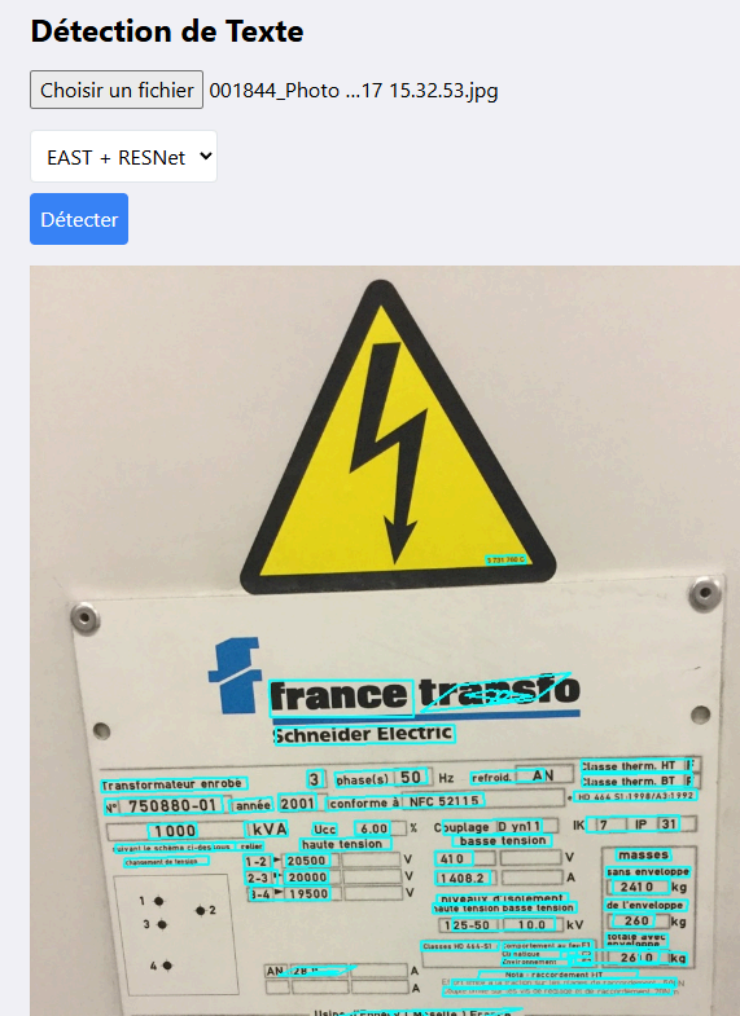
1. **Installation des dépendances** :
    - Backend : Installation via `pip` des bibliothèques nécessaires pour FastAPI et les modèles OCR.

    - Frontend : Installation via `npm` pour React.js et les dépendances associées.
2. **Lancement du serveur backend** :
    - Commande : `uvicorn main:app --reload`
3. **Lancement du frontend** :
    - Commande : `npm start` dans le dossier frontend.
4. **Utilisation de la plateforme** :
    - Une fois les deux serveurs démarrés, accéder à l'interface via `http://localhost:3000` pour interagir avec les modèles et les données.

4. Prochaines Étapes et Fonctions Futures

- **Intégration de nouvelles fonctionnalités** : En plus des fonctionnalités actuelles, nous prévoyons d'ajouter des options pour le **reinforcement learning** avec les nouvelles données collectées.
- **Améliorations de l'interface utilisateur** : Simplifier davantage l'utilisation pour permettre aux non-initiés d'interagir facilement avec les modèles OCR.

Les captures suivantes montrent quelques résultats des pages de notre application :



Conclusion

En somme, ce projet a permis d'explorer différentes techniques pour la détection et la classification d'informations textuelles sur des images de plaques signalétiques. Nous avons testé plusieurs modèles OCR, dont EASY OCR, Tesseract, LayoutLMv2 et PaddleOCR, avant de choisir PaddleOCR pour son équilibre entre précision et flexibilité. Nous avons ensuite procédé à un fine-tuning du modèle pour l'adapter spécifiquement à nos données annotées avec PPOCRLabel.

L'entraînement des modèles de détection, de reconnaissance et de classification a été réalisé avec succès, et nous avons même expérimenté la génération d'images pour enrichir notre base de données grâce à un modèle de diffusion stable. Enfin, une plateforme a été développée pour intégrer ces différentes étapes et permettre une utilisation pratique des résultats.

Cependant, il convient de noter que la page principale de la plateforme, qui vise à combiner les processus de détection et de classification en une seule interface, n'est pas encore finalisée. Néanmoins, les deux processus peuvent être testés séparément, et leur intégration complète est prévue pour la suite du développement. Cette approche démontre que chaque

composant du projet a été validé de manière indépendante avant d'être intégré, garantissant ainsi une meilleure fiabilité des résultats finaux.

Les prochaines étapes incluront l'amélioration de l'interface utilisateur, l'ajout de nouvelles fonctionnalités pour un apprentissage renforcé à partir des données collectées, ainsi que la finalisation de la page principale pour une expérience utilisateur plus fluide et intuitive.