

SAE QTQUICKDETECT

Veille Technologique

Réalisé par Gatien Da Rocha,
Jean-Loup Mellion et Jules Nicolas

Client: M. Le Lain
Responsable projet : Mt. Pham

Contexte

Rappel de Contexte :

Notre projet vise à développer un système avancé de détection d'objets à partir d'images et de vidéos. Le cahier des charges, qui guide notre développement, met en lumière plusieurs exigences clés que nous résumons comme suit:

1. Interface Utilisateur :

- Une interface conviviale et intuitive destinée à faciliter l'interaction de l'utilisateur.
 - Importation d'images et vidéos depuis divers supports.
 - Lancement de la détection d'objets en un clic.
 - Visualisation instantanée des résultats sur l'interface.
 - Sauvegarde des détections.
 - Sélection parmi divers modèles pré-chargés de détection.
 - Section dédiée à l'ajustement des hyper-paramètres pour une personnalisation.
 - Possibilité pour les experts d'intégrer leurs propres modèles.

2. Traitement d'Images et de Vidéos :

- Au-delà d'une interface conviviale, le système doit garantir un traitement efficace et précis des médias :
 - Capacité à traiter indifféremment images et vidéos.
 - Utilisation des technologies de pointe pour une détection d'objets fiable.

3. Documentation et Guides :

- Pour assurer la prise en main du système par tous types d'utilisateurs :
 - Fourniture d'une documentation complète, accessible à tous.
 - Proposition d'un guide détaillé pour les utilisateurs souhaitant approfondir la personnalisation.
 - Présentation d'une démonstration vidéo pour illustrer l'utilisation concrète de l'outil.

Ce projet, à la croisée de la technologie et de l'expérience utilisateur, ambitionne de fournir un outil complet et efficient pour la détection d'objets à partir de différents médias.

Liste des technologies

Interface utilisateur :

- Electron
- QT
- TKinter
- JavaFX

Deep learning :

- ScikitLearn
- Tensorflow
- PyTorch
- ONNX

Modèles :

- YOLOv8
- Faster-R-CNN
- SSD

Langages :

- C++
- Python
- Java
- Javascript





Le langage sélectionné dépendra du framework de deep learning et du toolkit IHM.

Comparatif des technologies

Interface utilisateur :

- **Electron** : Electron est un cadre de développement populaire pour la création d'applications de bureau multiplateformes. Il combine la puissance de Chromium (pour l'affichage) et de Node.js (pour le backend), permettant ainsi aux développeurs de construire des applications en utilisant des technologies web telles que HTML, CSS et JavaScript.
 - **Avantages** : moderne et support multiplateforme.
 - **Inconvénients** : consommation élevée de ressources et nécessite une API pour une communication entre Python et JavaScript.
 - **Licence** : MIT - Permissif.
- **Qt** : Qt est un cadre de développement d'interfaces graphiques puissant et multiplateforme. Bien qu'il soit écrit en C++, des bindings sont disponibles pour plusieurs autres langages, dont Python.
 - **Avantages** : intégration facile avec Python, support multiplateforme.
 - **Inconvénients** : choix entre PyQt (plus documenté) et PySide (licence plus flexible).
- **Tkinter** : Tkinter est la bibliothèque standard d'interfaces graphiques pour Python. Elle offre une méthode simple et directe pour créer des fenêtres, des boutons et d'autres éléments d'interface.
 - **Avantages** : pas besoin d'installation supplémentaire, license permissive (BSD).
 - **Inconvénients** : moins flexible et modernité limitée.
- **JavaFX** : JavaFX est un framework de développement d'interfaces graphiques pour Java. Il offre une gamme d'outils pour créer des applications riches et modernes.
 - **Inconvénients** : nécessite une API pour la communication entre Python et Java et est sous licence GPL, limitant le choix de la licence commerciale.

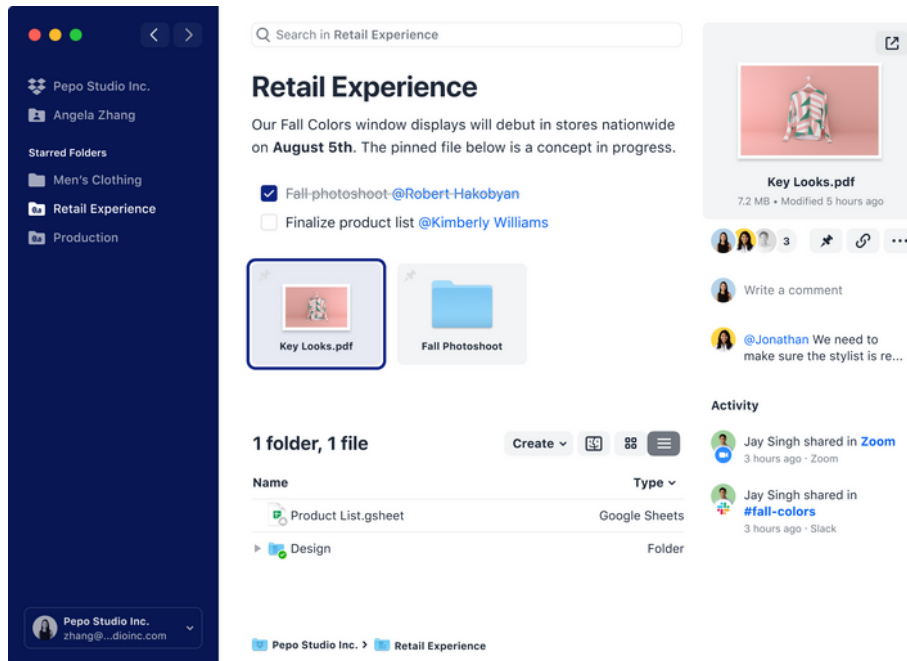
Comparatif des technologies

	 Electron	 Qt	 Tkinter	 JavaFX
Langage(s)	Web (Javascript, html, css, WebAssembly)	C++, Python, ...	Python	Java
Qualité de l'interface utilisateur	+++	+++	-	+++
Empreinte ressources	+++	+	-	++
License	MIT	GPL (PyQT) IGPL (PySide)	BSD	GPL

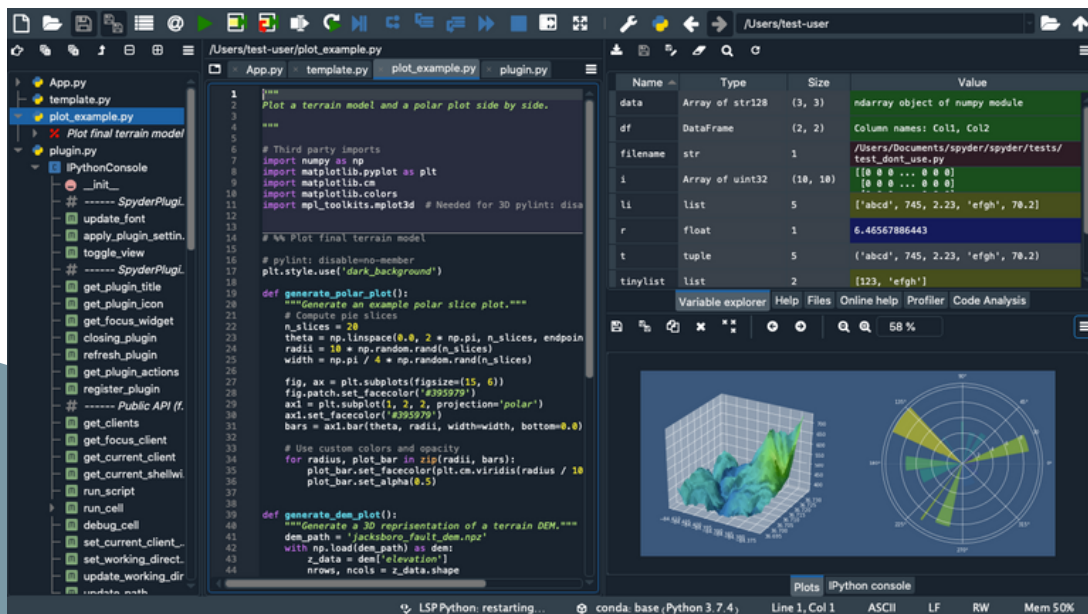
Décision : Utilisation de Qt avec PyQt pour sa documentation et sa stabilité.

Exemples d'usage de PyQt

Dropbox desktop :



Spyder IDE Python :





Comparatif des technologies

Modèles de détection :

- **YOLOv8** : YOLOv8 (You Only Look Once) est la huitième itération de la série YOLO, qui est réputée pour sa détection d'objets en temps réel. YOLO adopte une approche unique en traitant la détection d'objets comme une régression, contrairement à d'autres modèles qui l'abordent comme un problème de classification.
 - **Avantages** : Bon tooling, à jour, performant en rapidité et précision.
 - **Inconvénients** : Consommation élevée de ressources matérielles, complexité de l'entraînement.
 - **Licence** : AGPL-3.0
- **Faster R-CNN** : Faster R-CNN est un modèle de détection d'objets qui utilise une combinaison de réseaux de propositions de régions (RPN) et de réseaux neuronaux convolutifs profonds pour détecter et classer les objets.
 - **Avantages** : Maturité du modèle, flexibilité avec personnalisation des backbones.
 - **Inconvénients** : Architecture plus ancienne, performances potentiellement inférieures à YOLOv8.
 - **Licence** : Implementation dans **Detectron** : Apache 2.0
- **SSD (Single Shot Detection)** : SSD est un modèle de détection d'objets qui vise à effectuer la détection en une seule passe, éliminant ainsi la nécessité de multiples étapes intermédiaires.
 - **Avantages** : Performant en vitesse, flexibilité avec ajustement de la taille des boîtes englobantes.
 - **Inconvénients** : Commence à dater, bibliothèques moins avancées, précision potentiellement sacrifiée.
 - **Licence** : Implementation dans **ssd.pytorch** : MIT

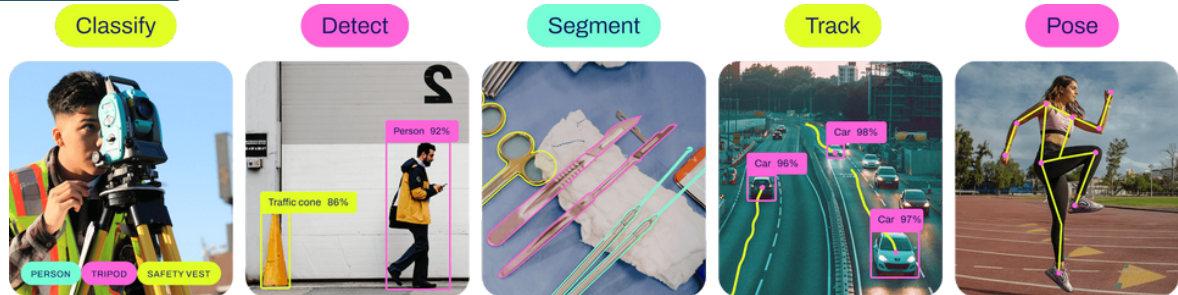
Comparatif des technologies

	 YOLOv8	 Faster R-CNN (Detectron)	SSD (Single Shot Detection) (ssd.pytorch)
Framework Deep learning	PyTorch, ONNX	Caffe2	PyTorch
Précision	++	+++	+
Vitesse d'inférence	+++	-	+
License	AGPL-3.0	Apache 2.0	MIT

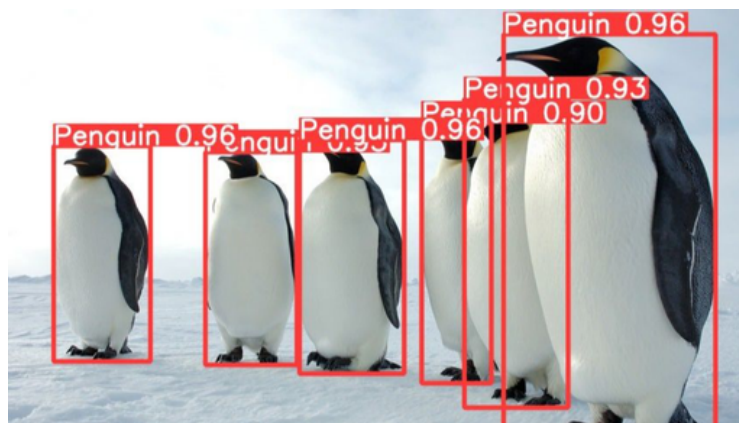
Décision : Utilisation de YOLOv8 pour sa rapidité, sa précision, et les outils bien conçus qu'il propose. D'autres modèles seront rajoutés par la suite.

Exemples d'usage de YOLOv8

Support de différentes tâches :







Détection sur image







Comparatif des technologies

Deep Learning :

- **ScikitLearn** : Bibliothèque de référence en matière d'apprentissage automatique pour Python. Elle fournit une vaste gamme d'outils supervisés et non supervisés, ainsi que des outils pour la sélection de modèles et le prétraitement des données.
 - **Avantages** : Facilité d'utilisation, vaste gamme d'algorithmes.
 - **Inconvénients** : Moins optimisé pour le deep learning.
 - **Licence** : BSD.
- **Tensorflow** : Bibliothèque de deep learning open source développée par Google. Elle est conçue pour fournir un cadre flexible pour construire et déployer des modèles de deep learning à grande échelle.
 - **Avantages** : Hautes performances, adaptabilité, TensorBoard pour la visualisation.
 - **Inconvénients** : Courbe d'apprentissage plus élevée.
 - **Licence** : Apache 2.0.**TensorFlow**
- **PyTorch** : Framework de deep learning open source développé par Facebook's AI Research lab. Il est conçu pour être flexible et permettre une recherche en deep learning plus intuitive grâce à son approche impérative.
 - **Avantages** : Bon support pour l'accélération matérielle, largement utilisé, bien documenté.
 - **Inconvénients** : Fonctionnalités de déploiement limitées.
 - **Licence** : BSD.PyTorch
- **ONNX** : Format ouvert pour représenter les modèles de deep learning. Il vise à fournir une interopérabilité entre les différentes bibliothèques et outils de deep learning.
 - **Avantages** : Interopérabilité entre divers outils de ML.
 - **Inconvénients** : Juste un format, nécessite des outils compatibles.
 - **Licence** : MIT.
ONNX

Comparatif des technologies

				
Type	Bibliothèque	Bibliothèque	Framework	Format
Focus principal	ML traditionnel	Deep Learning	Deep Learning	Interopérabilité
Facilité d'utilisation	+++	-	++	+
Performance	+	+++	+++	+++
Licence	BSD	Apache 2.0	BSD	MIT

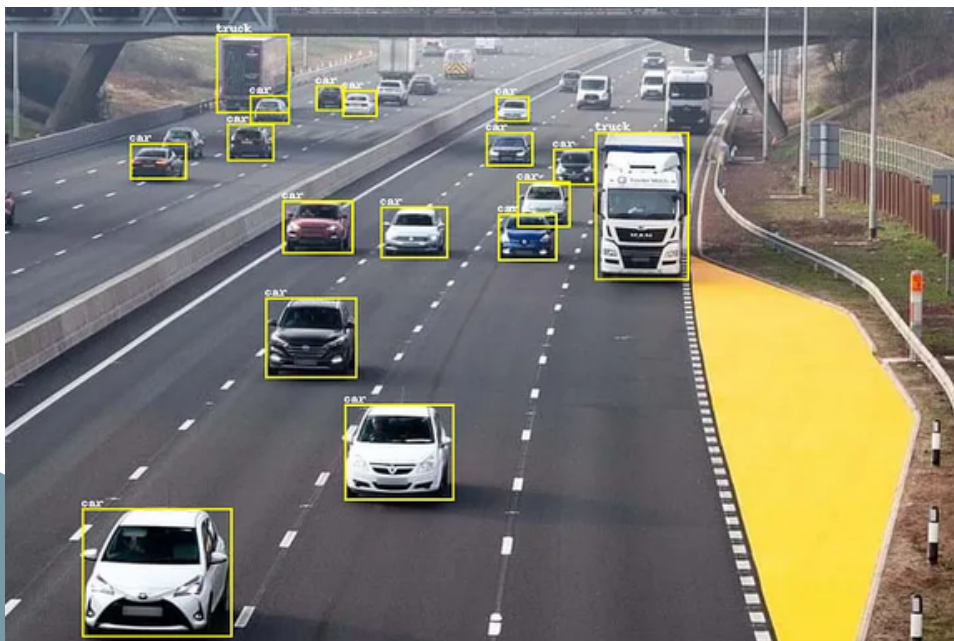
Décision : Utilisation de PyTorch pour son bon support d'accélération matérielle, sa large utilisation et sa documentation complète.

Exemples d'usage de PyTorch

Analyse de trafic (Uber) :



Detection d'objets :



Conclusion

Deep learning :

- Le projet utilisera **PyTorch**, car il est très bien documenté et dispose d'une communauté active. De plus, la grande majorité des modèles de Deep Learning tournent sur PyTorch, et il est même possible de rendre l'inférence des modèles plus rapides à l'aide de l'accélération matérielle.

Interface utilisateur :

- Nous avons retenu **PyQT** pour la réalisation de l'interface graphique. Nous avons choisi ce framework car il est très complet et permet de réaliser des interfaces graphiques modernes. De plus, il est très bien documenté et dispose également d'une communauté active.

Langages :

- Les technologies utilisées sont toutes compatibles avec **Python**, nous n'aurons donc pas besoin d'utiliser d'autres langages.

Modèles :

- **YOLOv8** est le choix retenu pour le modèle de détection. Il est performant, tant en terme de précision que de vitesse d'exécution. Il dispose également de différents modes de fonctionnement, ce qui permet l'ajout de plus de fonctionnalités au projet.

Autres :

- On utilisera **OpenCV** pour le traitement d'image, et **FFmpeg** pour pouvoir utiliser n'importe quel format de media en entrée.

Index

- Contexte • 1
- Liste des technologies • 2
- Comparatif interface utilisateur • 3, 4
- Exemple d'utilisation PyQt • 5
- Comparatif modèles de détection • 6, 7
- Exemple d'utilisation YoloV8 • 8
- Comparatif deep learning • 9, 10
- Exemple d'utilisation PyTorch • 11
- Conclusion • 12
- Index • 13