Document scientifique

Gestion de situation de crise

Jiek RUAN
Adrien FORMOSO
Ibrahim ALKARDO
Guillaume LE DEZ

DIA3 HETIC 2025

Sommaire

1 . Spécification et Conception de la Solution

1.1 Analyse des besoins et objectifs du projet
1.2 État de l’art des outils et approches méthodologiques
1.3 Choix technologiques et architecturaux

2. Développement de la Solution

2.1 Méthodologie et gestion du cycle de vie du projet
2.2 Conception et implémentation de l’architecture logicielle
2.3 Développement et intégration des composants
2.4 Optimisation et gestion des erreurs

3. Tests, Validation et Déploiement

3.1 Stratégie de tests et validation du modèle
3.2 Évaluation des performances et comparaison avec l’existant
3.3 Déploiement et maintenance évolutive

4. Résolution de Problèmes et Améliorations

4.1 Analyse et correction des défaillances
4.2 Optimisation continue et perspectives d’évolution

5. Conclusion

1. Introduction

1.1 Analyse des besoins et objectifs du projet

Une grande entreprise de fabrication de vis souhaite automatiser son contrôle qualité à l'aide d'un système de vision basé sur une caméra. L'objectif est de détecter automatiquement les vis non conformes afin d'optimiser la cadence de production et d'assurer un niveau de qualité constant.

Une première application a été développée en interne par l'équipe data, mais elle ne fonctionne pas correctement : elle classifie toutes les vis comme « conformes », rendant le contrôle qualité inefficace. Face à cet échec, une nouvelle équipe de quatre personnes a été constituée pour concevoir une solution plus performante.

L'enjeu principal est donc de concevoir et entraîner un modèle de réseau de neurones capable d'analyser un dataset d'images et d'améliorer significativement la détection des vis non conformes. Cette solution devra être fiable, scalable et s'intégrer efficacement dans le processus industriel existant.

1.2 État de l'art des outils et approches méthodologiques

Modèles existants pour l'analyse d'image

L'inspection visuelle automatisée repose sur l'apprentissage profond, en particulier les réseaux de neurones convolutifs (CNN). Différentes approches sont utilisées en fonction des contraintes du projet :

- CNN classiques (ResNet, EfficientNet) : Réseaux performants pour la classification d'images et l'extraction de caractéristiques.
- Auto-encodeurs : Utilisés pour la détection non supervisée d'anomalies par reconstruction d'image.

Les benchmarks de référence incluent MVTec AD Dataset et Magnetic Tile Defects Dataset, utilisés pour évaluer la performance des modèles sur des défauts industriels.

Notre approche :

1. Prétraitement des données : Normalisation, augmentation et segmentation des images.
2. Sélection du modèle : Comparaison des performances entre CNN et auto-encodeurs.
3. Entraînement : Optimisation des hyperparamètres et validation croisée.
4. Évaluation : Utilisation des métriques de précision, rappel et AUC.
5. Déploiement : Intégration dans la chaîne de production avec surveillance continue et mises à jour périodiques.

1.3 Choix technologiques et architecturaux

Le développement de la solution repose sur un ensemble d'outils et de bibliothèques éprouvés pour l'analyse d'images et l'apprentissage automatique. Le choix de ces technologies s'appuie sur leur robustesse, leur flexibilité et leur adoption dans l'industrie.

Technologies utilisées

- Python : Langage principal pour le développement, choisi pour sa richesse en bibliothèques dédiées à l'IA et son écosystème mature.
- Keras : API de haut niveau facilitant l'implémentation et l'entraînement de réseaux de neurones profonds via TensorFlow.
- Scikit-learn : Outils complémentaires pour la pré-analyse des données, l'ingénierie des caractéristiques et l'évaluation des modèles.
- Pandas & Seaborn : Manipulation, exploration et visualisation des données pour l'analyse des tendances et la compréhension des distributions.
- Streamlit : Déploiement rapide d'une interface utilisateur interactive pour la visualisation des résultats et l'évaluation du modèle en temps réel.

Approche architecturale

Bien qu'aucune architecture complexe ne soit mise en place, le projet est structuré de manière modulaire :

1. Prétraitement des données : Chargement, nettoyage et augmentation des images via Pandas et Keras.

2. Modélisation : Définition et entraînement du réseau de neurones avec Keras

3. Évaluation : Analyse des performances via Scikit-learn et visualisation des résultats avec Seaborn.

4. Déploiement : Interface interactive via Streamlit pour faciliter l'exploitation des prédictions.

Ce choix technologique garantit une solution agile, efficace et facilement itérable en fonction des résultats obtenus et des ajustements nécessaires.

2. Développement de la Solution

2.1 Méthodologie et gestion du cycle de vie du projet

Le projet ayant été réalisé sur une période de 4 jours, nous avons choisi d'adopter une approche itérative et pragmatique, en nous concentrant sur les fonctionnalités essentielles afin d'optimiser l'efficacité du développement

Organisation du travail

Pour structurer notre progression, nous avons réparti le travail en quatre étapes clés, correspondant à chaque journée :

- Jour 1 : Définition des besoins, répartition des tâches et mise en place de l'environnement de travail.
- Jour 2 : Développement du modèle CNN pour la classification binaire et premiers tests.
- Jour 3 : Intégration du modèle dans l'application et optimisation des performances.
- Jour 4 : Tests finaux, corrections des erreurs et préparation de la présentation.

Outils et suivi

Pour organiser nos tâches et suivre notre avancement, nous avons utilisé Google Docs, tandis que Github nous a permis de gérer le code de manière collaborative et d'assurer un bon suivi des versions.

2.2 Conception et implémentation de l'architecture logicielle

L'architecture logicielle du projet est conçue pour être modulaire et évolutive, permettant une gestion claire des différentes étapes du processus de développement, de l'analyse des données à la mise en production.

L'architecture se divise en plusieurs modules interconnectés, chacun dédié à une étape clé du développement.

1. Prétraitement des données : La première étape consiste à charger les images à partir des sources de données, à les nettoyer et à les augmenter pour enrichir le jeu d'entraînement, ce qui améliore la robustesse du modèle. L'augmentation est réalisée à l'aide de bibliothèques comme Keras et Pandas.

2. Modélisation du réseau de neurones : Le modèle est construit avec Keras, qui fournit une interface hautement flexible et facile à utiliser pour la définition du réseau de neurones. TensorFlow, en arrière-plan, assure la gestion des calculs et l'entraînement du modèle. La structure du réseau est ajustée selon les besoins spécifiques du projet.
   - La dernière couche utilise la fonction d'activation Sigmoid, permettant de produire une probabilité de conformité/non-conformité.
   - La fonction de perte choisie est Binary Crossentropy, adaptée aux problèmes de classification binaire.

3. Entraînement et validation : Une fois le modèle défini, il est entraîné sur les données avec des techniques telles que l'early stopping et la réduction du taux d'apprentissage pour optimiser l'entraînement. Les performances sont évaluées en utilisant des métriques pertinentes comme la précision, le rappel et le F1-score,

calculées grâce à Scikit-learn.

4. Visualisation et évaluation des performances : Seaborn et Pandas sont utilisés pour analyser et visualiser les résultats des tests de performance, ce qui permet d'identifier les zones d'amélioration du modèle et d'itérer rapidement si nécessaire.

5. Déploiement et interface utilisateur : L'interface interactive pour la visualisation des résultats est déployée avec Streamlit, permettant aux utilisateurs de charger des images, de visualiser les prédictions et de tester différentes configurations du modèle en temps réel.

2.3 Développement et intégration des composants

Le développement a commencé par la préparation des données, avec un processus de data augmentation sur les images de vis non conformes, comprenant notamment des rotations pour accroître la diversité du jeu de données et renforcer la robustesse du modèle.

Le dataset a ensuite été divisé en ensembles d'entraînement et de validation, avec un split des données et un shuffle pour garantir la variabilité et éviter tout biais. Les images ont été normalisées pour que les valeurs des pixels soient dans une gamme adaptée à l'apprentissage.

Pour l'entraînement du modèle, nous avons utilisé des techniques d'optimisation comme Early Stopping pour éviter le surapprentissage et ReduceLROnPlateau pour ajuster le taux d'apprentissage en fonction des performances du modèle.

Les performances ont été suivies à travers les métriques loss, accuracy et la recall. Le réseau de neurones convolutionnel (CNN) a été conçu et entraîné avec Keras, permettant de classifier efficacement les vis conformes et non conformes. Une interface interactive a été mise en place via Streamlit, permettant aux utilisateurs de soumettre des images et d'obtenir des prédictions en temps réel.

2.4 Optimisation et gestion des erreurs

Lors de l'entraînement du modèle, plusieurs stratégies ont été appliquées pour optimiser les performances et éviter l’overfitting (surapprentissage). L'une de ces méthodes est l'**Early Stopping** qui interrompt l'entraînement si la performance sur les données de validation cesse de s'améliorer, ce qui permet de conserver le meilleur modèle.

Parallèlement, **ReduceLROnPlateau** ajuste le taux d'apprentissage lorsque les performances sur la validation stagnent, facilitant ainsi la convergence du modèle en évitant des ajustements trop brusques. Ces techniques combinées permettent d'améliorer la robustesse du modèle tout en garantissant un apprentissage plus efficace.

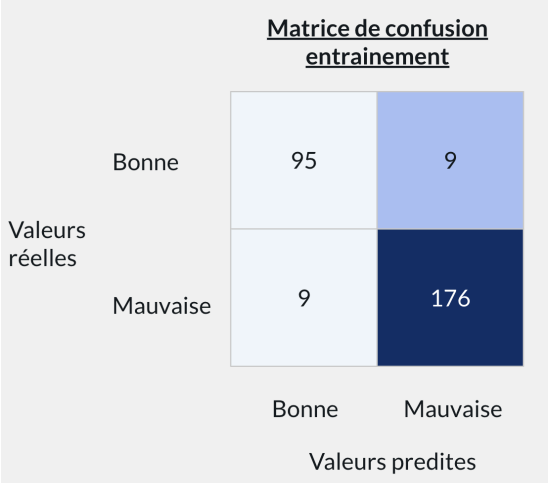
3.Tests, Validation et Déploiement

3.1 Stratégie de tests et validation du modèle

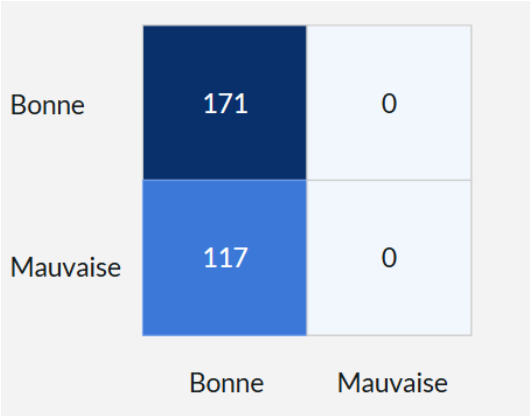
L'évaluation de la performance du modèle repose sur des métriques quantitatives et des outils d'analyse approfondis. Nous utiliserons l'accuracy et la loss pour mesurer respectivement la capacité du modèle à classifier correctement les vis et l'évolution de son apprentissage au fil des itérations.

En complément, une matrice de confusion sera exploitée pour analyser la répartition des prédictions et identifier les éventuelles erreurs (faux positifs et faux négatifs). Cet outil permettra d'évaluer la fiabilité du modèle en comparant ses prédictions à la réalité, et d'orienter d'éventuels ajustements afin d'améliorer sa robustesse.

Voici notre matrice avec des résultats convaincant sur l'entraînement du modèle :



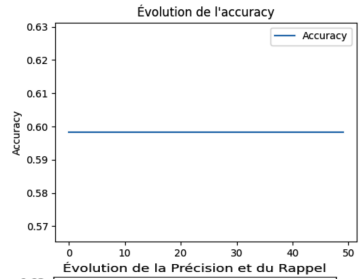
En comparaison les mauvaises prédictions par le modèle initial :

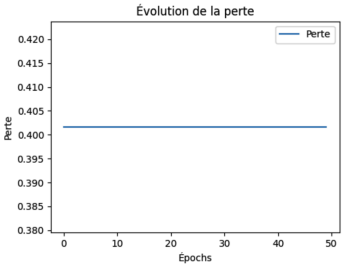


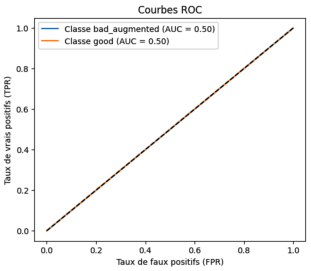
3.2 Évaluation des performances et comparaison avec l'existant

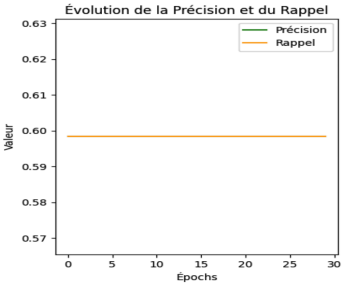
Performances modèle initial :

Accuracy : 0.5787
Loss : 0.4213
ROC : 0.5
Recall : 0.5787
Precision : 0.5787
F1 score : 0.5787



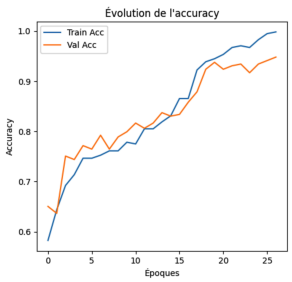


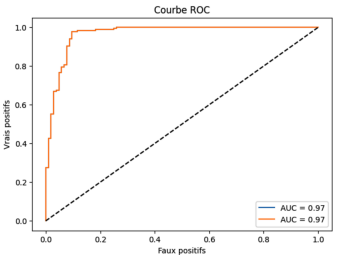


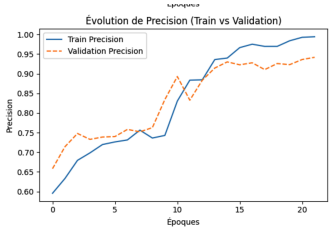


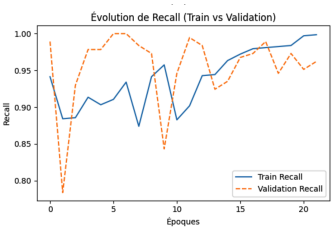
Performances nouveau modèle :

Accuracy : 0.9278
Loss : 0.2002
ROC : 0.97
Recall : 0.9730
Precision : 0.9278
F1 score : 0.9499









Paramètres modèle initial :

Layer (Type)	Output Shape	Param #
conv2d_2 (Conv2D)	(None, 62, 62, 16)	448
max_pooling2d_2 (MaxPooling2D)	(None, 31, 31, 16)	0
conv2d_3 (Conv2D)	(None, 29, 29, 32)	4 640
max_pooling2d_3 (MaxPooling2D)	(None, 14, 14, 32)	0
flatten_1 (Flatten)	(None, 6272)	0
dense_2 (Dense)	(None, 64)	401 472
dense_3 (Dense)	(None, 2)	130

Total des paramètres : 406 692 (1.55 MB)
Paramètres Optimizer: 2 (12.00 B)

Paramètres nouveau modèle :

Layer (Type)	Output Shape	Param #
Conv2D (conv2d)	(None, 222, 222, 64)	1 792
MaxPooling2D (max_pooling2d)	(None, 74, 74, 64)	0
Conv2D (conv2d_1)	(None, 72, 72, 128)	73 856
MaxPooling2D (max_pooling2d_1)	(None, 24, 24, 128)	0
Conv2D (conv2d_2)	(None, 22, 22, 256)	295 168
MaxPooling2D (max_pooling2d_2)	(None, 11, 11, 256)	0
Flatten (flatten)	(None, 30976)	0
Dense (dense)	(None, 256)	7 930 112
Dropout (dropout)	(None, 256)	0
Dense (dense_1)	(None, 1)	257

Total des paramètres : 8 301 187 (31.67 MB)
Paramètres Optimizer: 2 (12.00 B)

3.3 Maintenance de la solution

La solution est versionné sur un repository git.

4. Résolution de Problèmes et Améliorations

4.1 Analyse et correction des défaillances

Afin d'améliorer les performances du modèle, plusieurs ajustements ont été réalisés :

Modifications du modèle :

Fonction d'activation : Le modèle est passé de Softmax à Sigmoid pour mieux correspondre à la classification binaire des vis.

Fonction de perte : La Binary Cross-Entropy a remplacé la Mean Squared Error (MSE) pour mieux gérer les erreurs dans un contexte binaire.

Seuil de tolérance : Le seuil de tolérance a été ajusté de 0 à 0,5, ce qui signifie que toute probabilité supérieure ou égale à 0,5 est désormais considérée comme une prédiction correcte.

Courbe ROC : Utilisée pour évaluer la performance du modèle à différents seuils, offrant une analyse plus fine de la sensibilité et de la spécificité.

Améliorations/Ajouts :

LRonPlateau : Ajuste dynamiquement le taux d'apprentissage pour éviter que le modèle ne stagne.

Early Stopping : Arrête l'entraînement si les performances sur les données de validation cessent d’améliorer, afin de prévenir le surapprentissage.

Split du dataset : Le jeu de données a été divisé en ensembles d’entraînement, validation et test pour garantir une évaluation objective du modèle.

Loss/Accuracy : La loss mesure l'erreur du modèle et l'accuracy indique le taux de prédictions correctes, servant de base pour les ajustements.

Shuffle du dataset : Le mélange des données avant la séparation permet de minimiser les biais lors de l'entraînement.

Ces ajustements ont permis d'optimiser les performances du modèle, de le rendre plus robuste et de mieux gérer les données réelles.

4.2 Optimisation continue et perspectives d’évolution

Amélioration du dataset :

La collecte d'images en conditions réelles combinée à l'utilisation de GANs (Generative Adversarial Networks) pour générer des données synthétiques offrirait une solution pour équilibrer le dataset. Cela permettrait d'améliorer la robustesse et la généralisation du modèle en production, en compensant les éventuels déséquilibres dans les données et en optimisant les performances.

Optimisation des architectures :

L'exploration d'architectures alternatives et l'optimisation des hyperparamètres sont des leviers pour perfectionner la détection des défauts. Cela améliorerait également l'adaptabilité du modèle aux différentes conditions de production, rendant le système plus robuste et performant.

Extension du modèle à plusieurs classes :

Le modèle pourrait être étendu pour détecter différentes catégories de défauts en introduisant plusieurs classes. Par exemple, une classification fine des vis selon leur type de non-conformité :
Classe 0 : vis conforme
Classe 1 : vis avec impact
Classe 2 : vis avec le bout pliée
Classe 3 : vis sans bout, permettrait une détection plus précise et une gestion plus efficace des problèmes de qualité.

Amélioration de l'application Streamlit :

La refonte de l'interface de l'application visera à améliorer son ergonomie en simplifiant l'expérience utilisateur et en intégrant des visualisations plus intuitives. De plus, la possibilité de tester plusieurs photos simultanément serait ajoutée pour optimiser l'efficacité et la flexibilité de l'application.

5.Conclusion

En conclusion, La solution développée permet de détecter efficacement les vis non conformes en automatisant le contrôle qualité à l'aide de la vision par ordinateur. En partant de l'analyse des besoins et des objectifs, nous avons conçu un modèle de réseau de neurones convolutifs (CNN), soutenu par une approche modulaire et flexible pour le prétraitement, la modélisation, l’évaluation et le déploiement.

L'optimisation du modèle, grâce à des techniques telles que l’early stopping et la gestion du taux d’apprentissage, a permis d’améliorer la précision et la robustesse par rapport à l'application initiale, atteignant des performances nettement supérieures. La solution est intégrée dans un environnement de production avec une interface interactive, garantissant ainsi une évaluation en temps réel.

Des tests approfondis ont permis de valider le modèle, et des mesures sont en place pour une maintenance continue. En termes d'améliorations futures, la solution peut évoluer avec de nouveaux jeux de données et des mises à jour régulières pour maximiser son efficacité et son adaptabilité à long terme.

.