HESTIM
ENGINEERING & BUSINESS SCHOOL

5A IIIA

Rapport de Projet 3pH « PIC »

Sujet :

Détection d'Anomalies par Inspection Visuelle Industrielle

Réalisé par :
AKAKPO-DJAKPATA Mathieu
AGBEVIADE J. Komivi
ADJETE Kenneth Benedict

Membre de jury :
Pr. LAHLOU
Pr. GILCHRIS DANNON

Année universitaire
2025/2026

Table des matières

Liste des Figures

Liste des Tableaux

Introduction

Dans un contexte industriel en pleine mutation numérique, la qualité des produits manufacturés constitue un enjeu stratégique majeur. Les systèmes de contrôle qualité traditionnels, basés sur l'inspection visuelle humaine, atteignent aujourd'hui leurs limites face aux exigences croissantes de précision, de rapidité et de traçabilité. L'avènement du Deep Learning ouvre de nouvelles perspectives pour automatiser intelligemment ces processus critiques.

Ce document de cadrage présente un projet ambitieux visant à développer un système de détection automatique d'anomalies visuelles pour l'industrie 4.0. En s'appuyant sur le dataset MVTec AD, référence académique et industrielle reconnue, ce projet explore les approches supervisées et non-supervisées du Deep Learning pour répondre aux défis complexes de l'inspection automatisée.

L'objectif est double : d'une part, maîtriser les techniques de pointe en vision par ordinateur et en apprentissage profond ; d'autre part, déployer une solution opérationnelle, depuis l'entraînement des modèles jusqu'à une application web interactive permettant l'inférence en temps réel.

Ce rapport structure le projet en neuf sections : le contexte et les objectifs, la description détaillée des données, les fondements de la détection d'anomalies, les approches Deep Learning envisagées, les métriques d'évaluation, les technologies et outils, le planning prévisionnel, l'analyse des risques, et enfin une conclusion synthétisant les apports attendus.

1 Contexte et Objectifs du Projet

Introduction

La transformation numérique de l'industrie impose de repenser les processus de contrôle qualité. Cette première section établit le contexte dans lequel s'inscrit le projet, en identifiant les limitations des méthodes traditionnelles et en définissant précisément les objectifs à atteindre pour répondre aux enjeux de l'Industrie 4.0.

1.1 Contexte

Dans le cadre de l'Industrie 4.0, la transformation numérique des processus industriels nécessite des systèmes intelligents capables d'automatiser le contrôle qualité. Le contrôle visuel traditionnel, effectué manuellement par des opérateurs humains, présente plusieurs limitations : coûts élevés, fatigue humaine, manque de reproductibilité et difficulté à détecter des défauts subtils.

La détection automatique d'anomalies visuelles représente un enjeu majeur pour :

- Réduire les coûts liés aux défauts non détectés et aux rebuts
- Améliorer la qualité des produits de manière constante
- Accélérer les cadences de production sans compromettre la qualité
- Optimiser les processus en identifiant rapidement les sources de défauts

1.2 Objectifs

Ce projet a pour objectif de développer un système intelligent de détection d'anomalies visuelles basé sur le Deep Learning, avec les objectifs spécifiques suivants :

1. Préparer et analyser le dataset MVTec AD, reconnu comme référence dans les domaines académique et industriel.
2. Concevoir, entraîner et optimiser des modèles de détection d'anomalies, en utilisant des approches supervisées et non supervisées.
3. Évaluer de manière rigoureuse les performances des modèles à l'aide de métriques pertinentes telles que l'Accuracy, l'AUC-ROC et l'IoU.
4. Visualiser et interpréter les zones anormales détectées grâce à des techniques comme les heatmaps et les masques de segmentation.
5. Déployer une application web interactive capable d'effectuer l'inférence en temps réel.

Conclusion

Cette section a posé les fondations du projet en identifiant clairement les besoins industriels et les objectifs techniques. La suite du document détaillera les données sur lesquelles nous nous appuierons pour atteindre ces objectifs ambitieux.

2 Description des Données

Introduction

La qualité des données constitue le socle de tout projet de Machine Learning. Cette section présente en détail le dataset MVTec AD, ses caractéristiques techniques, sa composition et les types d'anomalies qu'il contient. Cette compréhension approfondie est essentielle pour adapter nos approches méthodologiques aux spécificités des données industrielles.

2.1 Provenance et Structure

Le MVTec Anomaly Detection (MVTec AD) Dataset est un dataset académique développé par MVTec Software GmbH, une entreprise spécialisée dans les solutions de vision industrielle. Il constitue le premier dataset complet et multi-objets destiné à la détection d'anomalies, avec des annotations précises au niveau des pixels.

Caractéristiques principales:

- 5 354 images haute résolution, allant de 700×700 à 1024×1024 pixels.
- 15 catégories, comprenant 5 textures et 10 objets.
- Images en couleurs (RGB) ou en niveaux de gris, selon les catégories.
- Acquisition professionnelle réalisée avec un capteur industriel 2048×2048 pixels équipé de lentilles télécentriques bilatérales.

2.2 Composition du Dataset

Type	Catégories
Textures	Carpet, Grid, Leather, Tile, Wood
Objets	Bottle, Cable, Capsule, Hazelnut, Metal Nut, Pill, Screw, Toothbrush, Transistor, Zipper

Table 1: Composition du dataset

2.3 Statistiques Clés

Ici, nous présentons les statistiques clés du dataset, mettant en évidence la répartition des images d'entraînement et de test, la diversité des anomalies, ainsi que l'étendue des annotations pixel-précises.

Distribution globale du dataset :

- Images d'entraînement : 3629, toutes sans défaut.
- Images de test : 1725, dont 467 normales et 1258 présentant des anomalies.
- Types de défauts : 73 catégories différentes.
- Régions annotées : 1900 masques avec annotations précises au niveau des pixels.

Exemples de statistiques par catégorie

Catégorie	Train	Test (bon)	Test (défaut)	Types de défauts
Carpet	280	28	89	5
Hazelnut	391	40	70	4
Pill	267	26	141	7
Toothbrush	60	12	30	1

Table 2: Exemples de statistiques par catégorie

Les défauts présents dans le dataset couvrent une large variété de situations industrielles réelles :

- **Défauts de surface** : rayures, bosses, contaminations.
- **Défauts structurels** : déformations, parties cassées.
- **Défauts par absence** : composants manquants.
- **Variations de couleur** : décolorations, taches.

Particularités du dataset :

- Certaines anomalies ont été générées manuellement afin de reproduire des scénarios réalistes.
- Les anomalies présentent des tailles très variables, allant de quelques pixels à plusieurs centaines.
- Les conditions d'illumination sont contrôlées, avec cependant des variations intentionnelles pour simuler différents environnements industriels.

Conclusion

Le dataset MVTec AD offre une base solide et réaliste pour développer des systèmes de détection d'anomalies. Sa diversité (textures et objets), ses annotations précises et son acquisition professionnelle en font une ressource idéale pour évaluer rigoureusement nos

approches. La section suivante présentera les concepts théoriques de la détection d'anomalies visuelles.

3 Approches Deep Learning Envisagées

Introduction

Cette section décrit les architectures de réseaux de neurones prévues pour le projet. Chaque approche répond à des objectifs spécifiques : expérimentation rapide, robustesse via le transfer learning, ou détection non-supervisée pour une généralisation maximale.

3.1 Approche Supervisée: CNN Custom

Architecture prévue

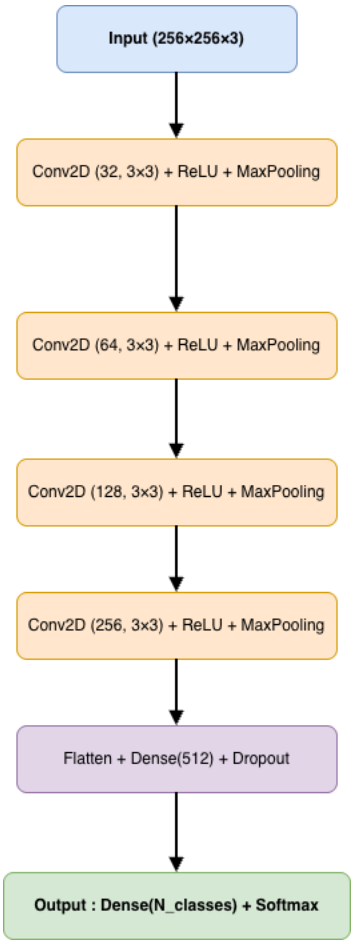

Figure 1: Architecture_CNN

Classification multi-classe

Le modèle prédit à la fois la présence d'anomalies et leur type spécifique (ex. rayure, contamination, déformation). Cette approche facilite la traçabilité et l'analyse des causes racines en production. Par exemple :

- **Classe 0** : Normal (aucun défaut)

- **Classe 1** : Rayure
- **Classe 2** : Contamination
- **Classe 3** : Déformation
- **Classe N** : Autres types de défauts

Cette approche multi-classe permet une traçabilité fine des défauts et facilite l'analyse des causes racines en production.

L'architecture choisie est simple afin de permettre l'établissement d'une baseline fiable. Elle comprend trois à quatre couches convolutionnelles, jugées suffisantes pour extraire efficacement les caractéristiques locales, telles que les textures et les défauts. Un mécanisme de dropout est intégré pour régulariser le modèle et prévenir le surapprentissage (overfitting). Enfin, la couche de sortie est adaptée au nombre de classes, incluant les différentes catégories de défauts ainsi que la classe normale.

3.2 Approche Supervisée: Transfer Learning

Cette approche consiste à utiliser le transfer learning, qui consiste à réutiliser des modèles pré-entraînés sur de grandes bases de données pour accélérer l'apprentissage sur notre dataset spécifique.

Modèles pré-entraînés envisagés :

1. ResNet-50 (He et al., 2016)
    - Profondeur : 50 couches avec connexions résiduelles
    - Pré-entraînement : ImageNet (1,2 million d'images)
    - Avantage : extraction de features robustes, évite le vanishing gradient
2. EfficientNet-B0 (Tan & Le, 2019)
    - Scaling optimal (profondeur, largeur, résolution)
    - Performances state-of-the-art avec moins de paramètres
    - Adaptable à différentes résolutions d'images
3. VGG-16 (Simonyan & Zisserman, 2014)

- Architecture simple et interprétable
- Baseline classique en vision industrielle

Classification Multi-Classe avec Transfer Learning

Pour chaque architecture pré-entraînée, nous remplacerons la couche de classification finale par une nouvelle couche Dense adaptée au nombre de types d'anomalies du dataset MVTec AD. Cette approche permet de bénéficier des features génériques apprises sur ImageNet tout en spécialisant le modèle sur nos catégories spécifiques de défauts.

Stratégie de Transfer Learning

Nous utiliserons une approche fine-tuning optimale :

Gel des couches de base : Les premières couches convolutionnelles (features génériques) restent figées.
Entraînement des dernières couches : Seules les 3-5 dernières couches sont réentraînées pour s'adapter aux défauts industriels.
Nouveau classificateur : Remplacement de la couche finale par notre classificateur multi-classe.
Learning rate faible : Utilisation d'un taux d'apprentissage réduit (1e-4 au lieu de 1e-2) pour les couches dégelées.

3.3 Approche Non-Supervisée: Autoencoders

Architecture Convolutional Autoencoder (CAE) :

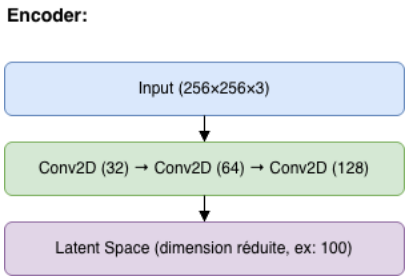

Figure 2: Encoder CAE

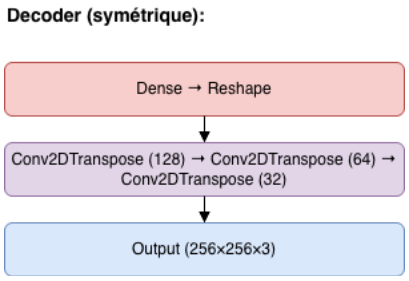

Figure 3: Decoder CAE

Fonction de perte

La reconstruction est optimisée en combinant une **perte L2 (MSE)**, qui évalue l'erreur pixel à pixel, et une **perte SSIM**(Structural Similarity Index) pour mieux préserver les structures visuelles importantes.

Détection d'anomalies

Les anomalies sont identifiées en calculant l'erreur de reconstruction pour chaque pixel. Un seuil est défini à partir du jeu de validation contenant uniquement des images normales, et des heatmaps d'anomalies peuvent être générées pour visualiser les régions problématiques.

Variantes avancées (optionnelles)

Pour améliorer la modélisation, il est possible d'utiliser des **Variational Autoencoders (VAE)** qui introduisent un aspect probabiliste dans l'espace latent, ou des **Adversarial Autoencoders** qui régularisent le latent space via une approche adversariale.

Conclusion

Ces trois approches complémentaires permettront une évaluation exhaustive : le CNN custom établira une baseline, le transfer learning exploitera des connaissances pré-acquises pour la classification multi-classe des types de défauts, et l'autoencoder testera la viabilité de la détection non-supervisée. Les résultats comparatifs guideront le choix de l'architecture finale pour le déploiement.

4 Métriques d'Évaluation

Introduction

L'évaluation rigoureuse d'un système de détection d'anomalies nécessite des métriques adaptées à plusieurs niveaux : classification globale, segmentation précise des défauts, et analyse par catégorie. Cette section présente les métriques retenues, leur interprétation et leur importance dans le contexte industriel.

4.1 Métriques de Classification (Image-Level)

4.1.1 AUC-ROC (Area Under the Receiver Operating Characteristic Curve)

$$\text{AUC-ROC} = \int_0^1 \text{TPR}(t)\, d\text{FPR}(t)$$

Principe : Mesure la capacité du modèle à séparer classes normales/anormales, indépendamment du seuil de décision

Interprétation :

- AUC = 0.5 : Performance aléatoire (modèle inutile)
- AUC = 1.0 : Séparation parfaite
- AUC > 0.8 : Performance généralement considérée comme bonne
- AUC > 0.9 : Excellente performance

  Avantages :
- Robuste au déséquilibre de classes (crucial car défauts rares en production)
- Évalue la qualité du scoring sur tout le spectre de seuils possibles
- Permet comparaison objective entre modèles différents

  Importance : Métrique principale dans le benchmark MVTec AD et standard industriel pour détection d'anomalies

Pourquoi AUC-ROC et pas Accuracy ?

En production, les défauts représentent souvent <1% des pièces. Un modèle prédisant toujours normal aurait 99% d'accuracy mais serait inutile. L'AUC-ROC évalue réellement la capacité discriminante du modèle.

4.1.2 F1-Score

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Precision = TP / (TP + FP) : Proportion de vraies anomalies parmi les détections

Recall = TP / (TP + FN) : Proportion de défauts réellement détectés

Utilité : Équilibre entre minimiser les fausses alarmes et maximiser la détection des défauts

Importance industrielle :

Précision élevée : Évite les arrêts de production injustifiés (coûts)

Rappel élevé : Garantit la qualité client (défauts non livrés)

F1 élevé : Optimise les deux objectifs simultanément

Avantage : Métrique unique et facilement communicable aux non-spécialistes

4.2 Métriques de Segmentation (Pixel-Level)

4.2.1 IoU (Intersection over Union) – Jaccard Index

$$\text{IoU} = \frac{\text{Intersection(Prédiction, GT)}}{\text{Union(Prédiction, GT)}}$$

Ground Truth (GT) = masques d'annotation manuels de référence fournis dans MVTec AD

Interprétation :

- IoU > 0.5 : Bonne détection (standard en détection d'objets)
- IoU > 0.75 : Excellente localisation
- IoU < 0.3 : Localisation insuffisante

Importance industrielle :

- Vérifie que la zone détectée correspond précisément au défaut réel
- Essentiel pour l'analyse des causes racines (où exactement est le problème ?)
- Permet la traçabilité et l'amélioration des processus

Avantage : Métrique géométrique intuitive, standard en segmentation

4.2.2 Per-Region Overlap (PRO)

Principe : Overlap moyen calculé sur chaque région annotée individuellement, puis moyenné

Différence avec IoU global :

- IoU global : un seul score pour toute l'image

- PRO : moyenne de scores calculés pour chaque défaut séparément

Avantage majeur : Équitable envers les petits et grands défauts

- Un grand défaut bien détecté ne "masque" pas les petits défauts manqués
- Chaque région compte autant dans l'évaluation

Importance : Métrique officielle du benchmark MVTec AD pour la segmentation

Contexte industriel : Les petits défauts (rayures fines, micro-fissures) sont souvent les plus critiques et difficiles à détecter

4.3 Stratégie d'Évaluation multi-Niveaux

Niveau 1 - Classification (Image-Level) :

- **AUC-ROC** : Métrique principale pour comparer les modèles
- **F1-Score** : Métrique opérationnelle à un seuil choisi
- Matrice de confusion pour analyse qualitative

Niveau 2 - Segmentation (Pixel-Level) :

- **IoU** : Précision de localisation
- **PRO** : Évaluation équitable multi-régions

Niveau 3 - Classification Multi-Classe :

- Matrice de confusion (types de défauts)
- F1-Score par classe (macro-average)
- Analyse des confusions entre types de défauts

Conclusion

Cette batterie complète de métriques permettra une évaluation rigoureuse et multidimensionnelle des modèles. L'AUC-ROC servira de métrique principale pour la comparaison globale, tandis que les métriques de segmentation (IoU, PRO) évalueront la précision de localisation, essentielle pour l'utilisabilité industrielle.

5. Technologies et Outils

Introduction

Le choix des technologies constitue un facteur déterminant pour la réussite technique du projet. Cette section présente et justifie les outils retenus pour le développement, l'entraînement, le déploiement et la mise à disposition de l'interface utilisateur, en précisant leurs avantages respectifs dans le contexte du projet.

5.1 Framework Deep Learning: TensorFlow / Keras

Justification du choix:

- **Écosystème mature** : Documentation exhaustive, communauté très active comptant plusieurs millions de développeurs, et résolution rapide des problèmes techniques.
- **Keras API** : Interface haut niveau intuitive permettant un prototypage rapide, réduisant jusqu'à 80 % du code par rapport à PyTorch en bas niveau.
- **TensorFlow 2.x** : Exécution immédiate (eager execution) activée par défaut, facilitant le débogage et l'expérimentation, avec une intégration native de Keras.
- **Déploiement simplifié** : Compatibilité directe avec TensorFlow Serving (API REST), TensorFlow Lite (appareils mobiles et edge computing) et **TensorFlow.js** (exécution dans le navigateur).
- **Support industriel** : Développé et maintenu par Google, adopté massivement dans l'industrie (Airbnb, Coca-Cola, Intel, etc.).
- **Performance optimisée** : Intègre le compilateur XLA, le support natif des GPUs NVIDIA et des TPUs Google, garantissant un entraînement accéléré et efficace.

Importance dans le projet

TensorFlow offre le meilleur compromis entre facilité de développement (Keras) et capacités de production (TensorFlow Serving, optimisations). Sa compatibilité avec AWS SageMaker et Lambda est un atout majeur pour notre architecture cloud.

Alternatives envisagées

PyTorch : Excellente pour recherche et prototypage, mais TensorFlow préféré pour déploiement production et intégration AWS

5.2 Cloud Computing: Amazon Web Services (AWS)

AWS offre une infrastructure cloud complète et intégrée, permettant de couvrir l'ensemble du cycle de vie ML (données → entraînement → déploiement → monitoring) sans friction entre services.

Services AWS utilisés

5.2.1 AWS SageMaker (Entraînement)

Plateforme managée dédiée à l'entraînement, à l'optimisation et à la gestion du cycle de vie des modèles d'apprentissage automatique.

Fonctionnalités clés

- **Notebooks managés** : Environnements Jupyter préconfigurés avec support GPU, facilitant le prototypage et l'expérimentation.
- **Training Jobs** : Exécution distribuée et scalable des tâches d'entraînement sur clusters GPU.
- **Hyperparameter Tuning** : Optimisation automatique des hyperparamètres via recherche bayésienne.
- **Model Registry** : Gestion centralisée du versioning, de la traçabilité et du déploiement des modèles.

Avantages spécifiques

- Accès à des instances GPU performantes *(*p3.2xlarge : 1× V100 16 Go ; p3.8xlarge : 4× V100*)*.
- Paiement à l'usage, sans investissement matériel initial.
- Intégration native avec Amazon S3 pour le stockage et le chargement de grands ensembles de données.
- Possibilité d'utiliser des Spot Instances pour réduire jusqu'à 70 % les coûts d'entraînement.

Importance

SageMaker permet d'expérimenter rapidement avec différentes architectures de réseaux de neurones tout en assurant une montée en charge transparente, sans contrainte matérielle locale.

5.2.2 AWS Lambda + API Gateway (Inférence)

Fournir un service d'inférence serverless hautement scalable pour l'application web.

Avantages

- **Serverless** : Aucune gestion d'infrastructure, mise à l'échelle automatique.
- **Auto-scaling** : Adaptation instantanée à la charge (de 1 à plus de 10 000 requêtes/s).
- **Coût optimisé** : Facturation à la requête, sans frais en cas d'inactivité.
- **Faible latence** : Temps de réponse inférieur à 500 ms pour une prédiction d'image.

Importance

Cette approche simplifie considérablement le déploiement du modèle et garantit une disponibilité élevée, même sans expertise DevOps, tout en optimisant les coûts opérationnels.

5.2.3 Amazon S3 (Stockage)

Assurer le stockage objet distribué et sécurisé des données, modèles et journaux d'entraînement.

Usages :

- Hébergement du dataset MVTec AD (5 354 images, 2 Go).
- Stockage et versioning des modèles entraînés (formats .h5 et SavedModel).
- Conservation des logs, métriques et artefacts de prédiction.

Avantages :

- Durabilité de 99.999999999 % (11 nines).
- Scalabilité illimitée pour la croissance des données.

- Intégration native avec SageMaker et Lambda, facilitant les échanges entre les services AWS.

5.2.4 Amazon ECR (Container Registry)

Registre Docker privé pour la gestion et le stockage des images de conteneurs nécessaires au déploiement du projet.

Usages :

- Hébergement des images Docker destinées à AWS Lambda (intégrant TensorFlow et ses dépendances).
- Garantie de la reproductibilité et de la cohérence des environnements entre développement, test et production.

Importance

ECR facilite la mise en place d'une infrastructure conteneurisée fiable, assurant une continuité entre les environnements de développement et de déploiement.

Configuration type recommandée

- **Entraînement** : SageMaker ml.p3.2xlarge (1× V100 GPU, 8 vCPUs, 61 Go RAM)
- **Inférence** : AWS Lambda – 3008 Mo RAM avec TensorFlow Lite optimisé
- **Coût estimé** : 50 $ à 100 $ pour l'ensemble du projet (1 mois d'entraînement et de déploiement)

5.3 Application Web: Streamlit

Justification du choix :

- **Simplicité** : Permet de concevoir des interfaces interactives avec le langage de programmation Python, sans recourir à des langages front-end tels que HTML, CSS ou JavaScript.
- **Rapidité** : Offre un développement extrêmement rapide — un prototype fonctionnel peut être obtenu en quelques heures, contre plusieurs jours avec Flask ou React.
- **Intégration native** : Compatibilité directe avec les principales bibliothèques de data science et de deep learning (TensorFlow, NumPy, Matplotlib, Pandas, etc.).

- **Composants visuels** : Large choix de widgets interactifs (upload de fichiers, sliders, graphiques, affichage d'images, métriques dynamiques).
- **Réactivité** : Réexécution automatique du script à chaque interaction utilisateur, garantissant une interface fluide et réactive.
- **Déploiement simplifié** : Hébergement possible sur Streamlit Cloud, ou via **conteneurisation Docker**, facilitant le déploiement sur serveur ou en environnement cloud.

Importance dans le projet

Streamlit joue un rôle central en tant que interface utilisateur du système de détection d'anomalies. Il permet de concentrer les efforts sur la logique métier et les aspects d'analyse machine learning, tout en offrant une présentation claire, intuitive et professionnelle des résultats. Cette approche réduit significativement le temps de développement et favorise la mise à disposition rapide d'un outil exploitable par les utilisateurs finaux.

Pages de l'application

- **Accueil** : Upload et prédiction instantanée avec classification du type d'anomalie
- **Analyse détaillée** : Heatmaps, scores par région, masques de segmentation
- **Statistiques** : Performance du modèle, métriques par catégorie
- **Documentation** : Guide utilisateur, exemples de défauts

5.4 Environnement de Développement

Configuration locale

IDE : VS Code / PyCharm Professional
Extensions : Python, Jupyter, TensorFlow Snippets
Debugging intégré pour TensorFlow

Environnement : Conda / venv pour isolation des dépendances
Version Control: Git / GitHub
Branching strategy: GitFlow
CI/CD : GitHub Actions pour tests automatisés

Packages Python principaux

tensorflow==2.15.0
numpy==1.24.3

```
opencv-python==4.8.1
matplotlib==3.8.0
seaborn==0.13.0
scikit-learn==1.3.2
Pillow==10.1.0
streamlit==1.28.0
boto3==1.29.0   # AWS SDK
```

Importance de l'environnement

Un environnement de développement bien structuré garantit la reproductibilité des expériences, facilite la collaboration et accélère le cycle de développement.

5.5 Outils de Suivi et Monitoring

TensorBoard

Rôle : Visualisation et monitoring de l'entraînement

Fonctionnalités utilisées

- Courbes d'entraînement (loss, accuracy, AUC-ROC)
- Visualisation des architectures de réseaux
- Analyse des activations et gradients
- Profiling des performances (GPU utilization, memory)

Importance

Permet de détecter rapidement les problèmes d'entraînement (overfitting, vanishing gradients) et d'optimiser les performances.

MLflow (optionnel)

Rôle : Tracking des expériences ML

Fonctionnalités

- Logging automatique des hyperparamètres

- Versioning des modèles
- Comparaison des runs (tableaux, graphiques)
- Registry centralisé de modèles

Avantage : Facilite la reproductibilité et la collaboration en équipe.

5.6 Schéma d'Architecture Global

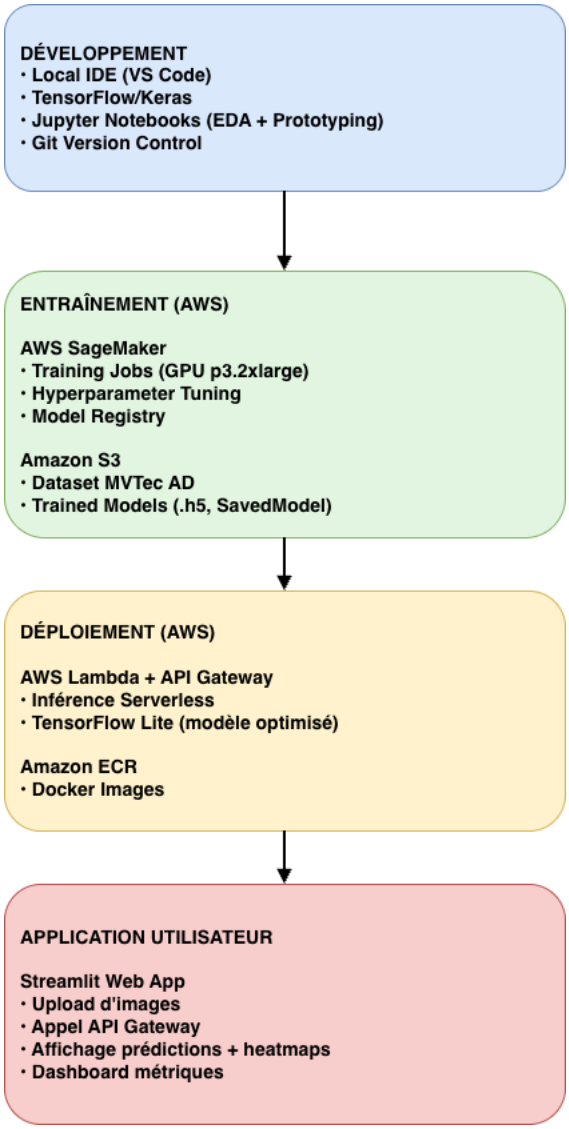

Figure 4: Architecture global

Conclusion

La stack technologique retenue (TensorFlow, AWS, Streamlit) offre un équilibre optimal entre puissance, simplicité et scalabilité. TensorFlow assure les performances d'entraînement et d'inférence, AWS fournit l'infrastructure cloud nécessaire sans gestion d'infrastructure complexe, et Streamlit permet de livrer rapidement une interface utilisateur professionnelle. Cette combinaison garantit la faisabilité technique du projet tout en minimisant les risques de déploiement.

6. Planning Prévisionnel

Introduction

Le succès du projet repose sur une planification claire des tâches et une allocation efficace du temps. Le projet est découpé en phases avec des jalons précis pour suivre l'avancement et ajuster si nécessaire.

Phase 1: Exploration et Préparation (Semaines 1-2)

Objectifs : Comprendre le dataset MVTec AD, préparer les pipelines de données, valider la faisabilité.

Tâches :

- Rédaction du document de cadrage (GO/noGO)
- Téléchargement et organisation du dataset
- Analyse exploratoire (distribution des classes, types de défauts, stats des images)
- Visualisations exemples normal/anormal
- Preprocessing : redimensionnement 256×256, normalisation, gestion RGB/grayscale
- Data augmentation : rotations, flips, zooms, ajustements luminosité/contraste

Livrables : Notebook EDA, pipeline validé, jeu de données structuré

Phase 2: Modélisation Supervisée (Semaines 3-4)

Objectifs : Établir une baseline CNN, exploiter le transfer learning, évaluer les performances initiales.

Tâches :

- Implémentation CNN 4 couches pour classification multi-classe
- Fine-tuning ResNet-50 et EfficientNet-B0, comparaison stratégies

- Entraînement et validation (EarlyStopping, ModelCheckpoint, TensorBoard)
- Évaluation : Accuracy, AUC-ROC, F1-Score, IoU, Dice, analyse par type de défaut

Livrables : Modèles entraînés, rapport comparatif, matrices de confusion

Phase 3 : Approche Non-Supervisée (Semaines 5-6)

Objectifs : Détection d'anomalies sans labels via autoencoder, comparaison avec supervisé.
Tâches :

- Implémentation autoencoder convolutionnel (encoder-decoder)
- Calibration des seuils de détection (ROC, validation sur images normales)
- Génération de heatmaps d'anomalies et post-processing
- Évaluation et comparaison supervisé vs non-supervisé

Livrables : Autoencoder entraîné, heatmaps, tableau comparatif

Phase 4 : Optimisation et Déploiement (Semaines 7-8)

Objectifs : Optimiser les modèles, déployer sur AWS, développer l'application Streamlit.
Tâches :

- Fine-tuning et hyperparameter tuning (SageMaker), éventuels ensembles
- Optimisation pour inférence (TensorFlow Lite, quantization, benchmark latence)
- Déploiement AWS (Lambda, API Gateway, Docker/ECR)
- Application Streamlit : upload, prédiction, visualisations, dashboard
- Tests end-to-end : fonctionnels et charge

Livrables : Modèles déployés, application fonctionnelle, documentation API

Phase 5 : Documentation et Présentation (Semaine 9)

Objectifs : Finaliser la documentation et préparer la démonstration.
Tâches :

- Rédaction du rapport final (synthèse, résultats, limites, perspectives)
- Documentation technique (guide utilisateur, API, déploiement)
- Préparation démonstration et slides, répétition pour soutenance

Livrables : Rapport final, documentation technique, présentation PowerPoint, démonstration live

Conclusion

Ce planning étalé sur 9 semaines permet une progression structurée du projet, avec des phases clairement délimitées. Chaque phase se conclut par des livrables tangibles permettant de valider l'avancement avant de passer à l'étape suivante. La flexibilité est maintenue pour ajuster les priorités en fonction des résultats intermédiaires.

7 Risques et Mitigations

Introduction

Tout projet technique comporte des risques susceptibles d'affecter sa réussite. Cette section identifie les principaux risques, évalue leur impact et probabilité, et propose des stratégies de mitigation concrètes pour chacun.

7.1 Tableau Synthétique des Risques

Risque	**Impact**	**Probabilité**	**Niveau**	**Mitigation**
Overfitting sur catégories peu représentées (ex: Toothbrush, 60 images)	Élevé	Moyenne	Critique	Data augmentation intensive, transfer learning, régularisation
Dépassement du budget AWS	Moyen	Faible	Modéré	Free Tier, suivi des coûts, optimisation des instances
Performances non-supervisées insuffisantes	Moyen	Moyenne	Modéré	Prioriser approche supervisée comme solution principale
Difficulté de déploiement Lambda (taille du modèle)	Faible	Moyenne	Mineur	TensorFlow Lite, quantization, modèles distillés
Déséquilibre des classes (types de défauts)	Moyen	Élevée	Modéré	Class weighting, focal loss, oversampling des classes minoritaires
Latence d'inférence trop élevée	Moyen	Faible	Modéré	Optimisation modèle, batch processing, caching

Table 3: Tableau synthétiques des risques

7.2 Analyse Détaillée des Risques Majeurs

Risque 1 - Overfitting sur Petits Datasets

Certaines catégories MVTec AD contiennent peu d'images (Toothbrush: 60, Transistor: 213). Le risque est que le modèle mémorise les données d'entraînement sans généraliser.

Indicateurs d'alerte

- Écart important entre accuracy train et validation (>10%)
- AUC-ROC stagnante ou en baisse

Stratégies de mitigation

1. **Data augmentation intensive** : rotations, flips, zooms, ajustements photométriques, Mixup/CutMix.
2. **Transfer Learning** : modèles pré-entraînés ImageNet, fine-tuning progressif.
3. **Régularisation**: dropout 0.3–0.5, L2 regularization, early stopping agressif.
4. **Validation croisée** : K-fold (k=5) et évaluation sur plusieurs seeds.

Risque 2 - Coûts AWS Excessifs

L'entraînement sur GPU peut rapidement générer des coûts élevés (~$3/h pour p3.2xlarge).

Stratégies de mitigation :

- **Free Tier et backup**: Lambda, S3, SageMaker gratuits, Google Colab Pro ou GPU local.
- **Optimisation des instances** : Spot Instances, arrêt automatique des notebooks, instances plus petites pour debug.
- **Monitoring** : AWS Budgets, CloudWatch, alertes en temps réel.

Risque 3 - Performances Non-Supervisées Insuffisantes

Les autoencoders peuvent produire des taux élevés de faux positifs/négatifs.

Stratégies de mitigation :

- Tester différentes architectures (CAE, VAE), loss SSIM + MSE, latent space adapté.
- Calibration rigoureuse des seuils (ROC, seuils adaptatifs par catégorie).
- Approche hybride : ensembling supervisé + non-supervisé.
- Accepter le risque secondaire : prioriser approche supervisée.

7.3 Plan de Contingence Global

- **Retards > 1 semaine** : prioriser transfer learning, réduire scope à 5 catégories, reporter déploiement AWS.
- **Performances faibles (AUC < 0.8)** : Analyse approfondie des échecs (confusion matrices), ajuster preprocessing, explorer architectures alternatives.
- **Problèmes techniques bloquants** : Support AWS (tickets techniques), Communauté TensorFlow/Stack Overflow, Pivot vers alternatives (PyTorch, Azure)

Conclusion

Identifier les risques et préparer des stratégies de mitigation permet d'anticiper les obstacles. Un suivi continu des indicateurs et l'adaptabilité du planning garantissent la réalisation des objectifs principaux sans compromettre le projet.

Conclusion

Synthèse du Projet

Ce document de cadrage a établi les fondations solides d'un projet ambitieux de détection d'anomalies visuelles industrielles par Deep Learning. En s'appuyant sur le dataset MVTec AD, référence académique reconnue, et en exploitant les technologies de pointe (TensorFlow, AWS, Streamlit), ce projet vise à développer un système intelligent capable d'automatiser le contrôle qualité industriel.

Apports Attendus

Ce projet permettra de :

1. Maîtriser les techniques state-of-the-art de détection d'anomalies visuelles

- Approches supervisées avec classification multi-classe des types de défauts
- Approches non-supervisées pour généralisation
- Techniques de transfer learning et fine-tuning

2. Comparer rigoureusement approches supervisées et non-supervisées

- Évaluation multi-niveaux (classification, segmentation)
- Analyse des forces et faiblesses de chaque paradigme
- Identification des contextes d'application optimaux

3. Déployer une solution ML complète de bout en bout

- Pipeline complet : données → entraînement → inférence → application
- Infrastructure cloud scalable et cost-effective
- Interface utilisateur professionnelle et intuitive

4. Acquérir une expertise en vision industrielle et Industrie 4.0

- Compréhension des contraintes industrielles réelles
- Maîtrise des métriques spécifiques (AUC-ROC, IoU, PRO)
- Capacité à déployer des solutions en production

Vision Finale

À l'issue de ce projet de 9 semaines, nous disposerons d'un système opérationnel de détection d'anomalies, d'une compréhension approfondie des approches Deep Learning pour l'inspection visuelle, et d'une expertise en déploiement de solutions ML sur le cloud. Ce projet constitue un tremplin vers une spécialisation en Computer Vision industrielle et en IA appliquée aux processus de fabrication.

La réussite de ce projet démontrera que l'automatisation intelligente du contrôle qualité n'est pas seulement un concept théorique, mais une réalité technologique accessible et déployable, marquant ainsi une contribution concrète à la transformation numérique de l'industrie.

Bibliographie

[1] P. Bergmann, M. Fauser, D. Sattlegger, et C. Steger, « MVTec AD – A Comprehensive Real-World Dataset for Unsupervised Anomaly Detection », Proc. IEEE CVPR,, 2019.

[2] K. He, X. Zhang, S. Ren, et J. Sun, « Deep Residual Learning for Image Recognition », Proc. IEEE CVPR, 2016.

[3] M. Tan et Q. V. Le, « EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks », Proc. ICML, 2019.

[4] K. Simonyan et A. Zisserman, « Very Deep Convolutional Networks for Large-Scale Image Recognition », Proc. ICLR,, 2014.

[5] P. Bergmann, S. Löwe, M. Fauser, D. Sattlegger, et C. Steger, « Improving Unsupervised Defect Segmentation by Applying Structural Similarity to Autoencoders », Proc. VISAPP, 2019.

[6] T. Schlegl, P. Seeböck, S. M. Waldstein, U. Schmidt-Erfurth, et G. Langs, « Unsupervised Anomaly Detection with Generative Adversarial Networks », Proc. IPMI, 2017.

Annexes

A. Liens Utiles

Dataset et Benchmarks

- Dataset MVTec AD: https://www.mvtec.com/company/research/datasets/mvtec-ad
- Papers with Code - Anomaly Detection: https://paperswithcode.com/task/anomaly-detection

Frameworks et Outils

- TensorFlow : https://www.tensorflow.org
- Keras API : https://keras.io
- Streamlit : https://streamlit.io

Cloud et Déploiement

- AWS SageMaker: https://aws.amazon.com/sagemaker
- AWS Lambda: https://aws.amazon.com/lambda
- TensorFlow Lite : https://www.tensorflow.org/lite

Communautés et Support

- TensorFlow Forum: https://discuss.tensorflow.org
- AWS Documentation : https://docs.aws.amazon.com
- Stack Overflow - Computer Vision: https://stackoverflow.com/questions/tagged/computer-vision

C. Glossaire

AUC-ROC : Area Under the Receiver Operating Characteristic Curve - métrique évaluant la capacité de séparation entre classes

Autoencoder : Architecture de réseau de neurones apprenant à compresser et reconstruire des données

Fine-tuning : Réentraînement d'un modèle pré-entraîné sur un nouveau dataset

IoU : Intersection over Union - métrique de similarité entre régions prédites et réelles

Transfer Learning : Réutilisation de connaissances apprises sur une tâche pour en résoudre une nouvelle

Heatmap : Visualisation colorée indiquant les zones d'intérêt ou d'anomalie dans une image