17 mars, 2025

Data challenge - Amélioration du contrôle qualité industriel grâce au Computer Vision

Eliot Morard

Table des matières

I / Introduction 3
I - 1 / Contexte du challenge 3
I - 2 / Objectifs du challenge 3
I - 3 / Métrique de coût 3
I - 4 / Le jeux de données 4
II / Méthodologie adoptée 5
II - 1 / Pipeline 5
III / Résolution du problème 7
III - 1 / CNN 7
III - 1 . 1 / CNN basique 7
III - 1 . 2 / CNN plus profond 8
III - 2 / Transfert learning 10
III - 3 / Resnet50 11
III - 3 . 1 / Matrice des coûts personnalisée 13
III - 3 . 2 / Synthèse 15
III - 4 / Architecture Resnet101 17
III - 4 . 1 / Synthèse 21
IV / Conclusion 22

CONTEXTE DU CHALLENGE

I / Introduction

I - 1 / Contexte du challenge

Les opérations de contrôle qualité intégrées aux chaines de production permettent à la fois de garantir la qualité finale du produit, d'éviter la propagation de pièce défaillantes vers les process suivants et de détecter au plus tôt les dérives pour en corriger les causes.

Une technique courante est d'utiliser une caméra pour réaliser une ou plusieurs prises de vues du produit et d'y appliquer des algorithmes spécifiques pour controler certaines caractéristiques. Si le système identifie un défaut, il est courant de demander un recontrole humain ce qui constitue une charge de travail supplémentaire et un risque d'erreur inhérent au contrôle humain.

L'utilisation de modèles de computer vision permet de rejuger automatiquement certains types d'images avec des performances supérieures à celles des opérateurs.

I - 2 / Objectifs du challenge

L'objectif de ce challenge est de développer un modèle de classification d'images dans un contexte industriel. Concrètement, il s'agit d'identifier, à partir de photographies de composants, celles qui présentent des défauts afin d'optimiser la satisfaction client et de prévenir les dysfonctionnements au sein de la chaîne de production. Le jeu de données comporte une classe dédiée aux pièces conformes, cinq classes correspondant aux types de défauts préalablement identifiés, ainsi qu'une classe « drift » destinée à regrouper les anomalies non prévues, identifiées en phase d'inférence.

Pour évaluer la robustesse du modèle, nous utiliserons une métrique basée sur une matrice de coût personnalisée, conçue pour pénaliser de manière disproportionnée les erreurs relatives à la non-détection des images anormales.

I - 3 / Métrique de coût

Pour ce défi, nous utilisons une matrice de coût personnalisée associée à la métrique Penalty Weighted Accuracy **(PWA)**. La **PWA** quantifie la performance du modèle en comparant le coût total des erreurs de classification aux pires erreurs possibles, en pondérant spécifiquement les non-détections d'images anormales.

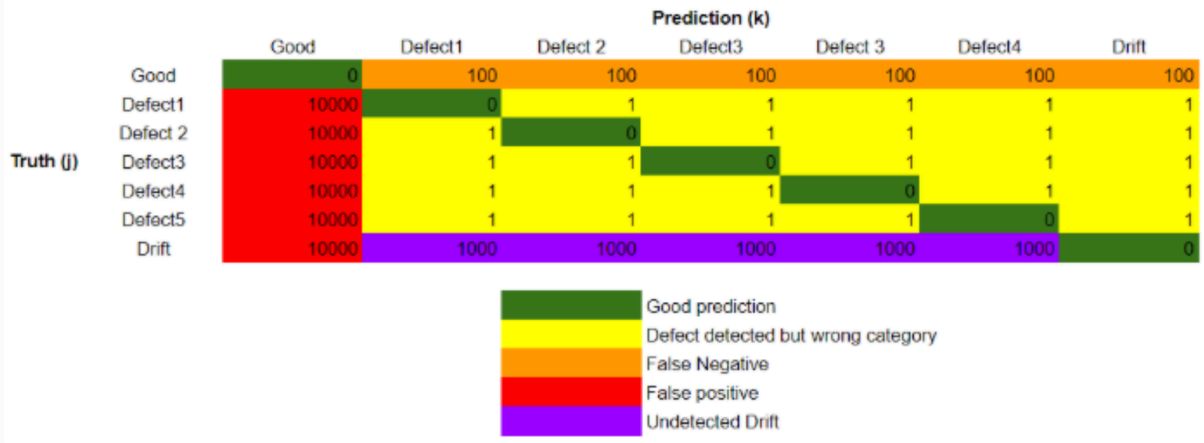

Fig. 2. – Matrice des coûts pour la **PWA**

Pendant la phase de test, le jeu de données comprend 1085 images. Pour illustrer la sensibilité et les limites de la **PWA**, nous proposons d'étudier un scénario trivial : classer systématiquement toutes

les images dans la même classe de défaut, choisie de manière à minimiser le coût de classification global. C'est à dire en optant pour la classe de défaut qui engendre la pénalité la plus faible, sans jamais identifier correctement les images conformes.

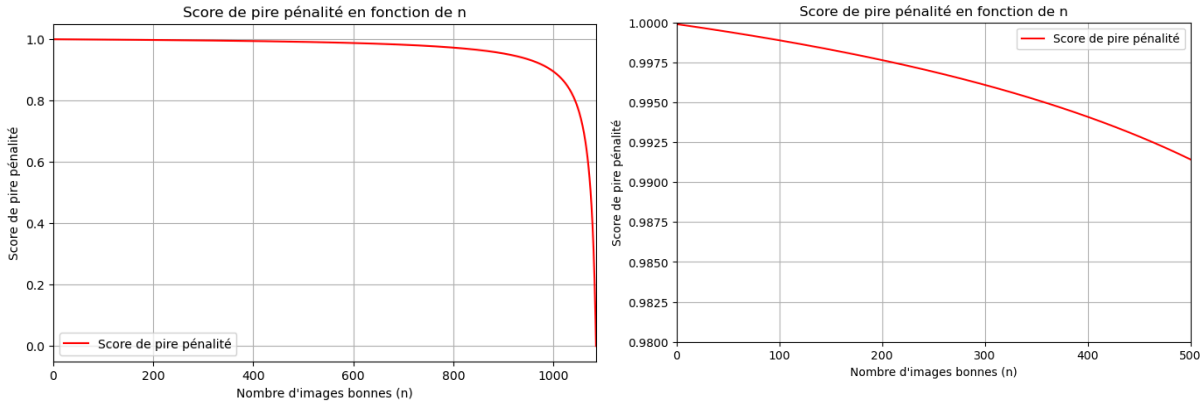

Fig. 3. – **PWA** en fonction du nombre n d'image de la classe GOOD

Nous pouvons conclure que lorsque le nombre d'échantillons est très faible, la métrique de **PWA** perd de sa pertinence, car sa sensibilité aux erreurs de classification est limitée dans ce contexte. Il semble que l'adoption de cette métrique soit justifiée par un taux de défaut relativement bas dans le jeu de données, ce qui permet de mieux distinguer les erreurs critiques. Cependant, il est important de noter qu'environ 15% des images de l'ensemble d'entraînement sont classées comme bonnes. Cette minorité de la classe "bonnes images" peut biaiser le classement **PWA**, car le modèle risque de privilégier ce classement trivial en excluant toute bonne image et en la classant comme défectueuse.

I - 4 / Le jeux de données

Le jeu de données utilisé est constitué des classes suivantes :

- GOOD (Label = 0) : 1235 images
- Boucle plate (Label = 1) : 71 images
- Lift-off blanc (Label = 2) : 270 images
- Lift-off noir (Label = 3) : 104 images
- Missing (Label = 4) : 6472 images
- Short circuit MOS (Label = 5) : 126 images

Les images ont été capturées sous quatre angles de vue différents (DIE01, DIE02, DIE03, DIE04). Compte tenu du nombre limité d'exemples disponibles, nous avons choisi de ne pas différencier explicitement ces angles de vue lors de l'entraînement. L'hypothèse est que le modèle pourra, par le biais de l'apprentissage automatique, extraire et généraliser les caractéristiques discriminantes essentielles pour la classification, sans nécessiter une séparation explicite par angle.

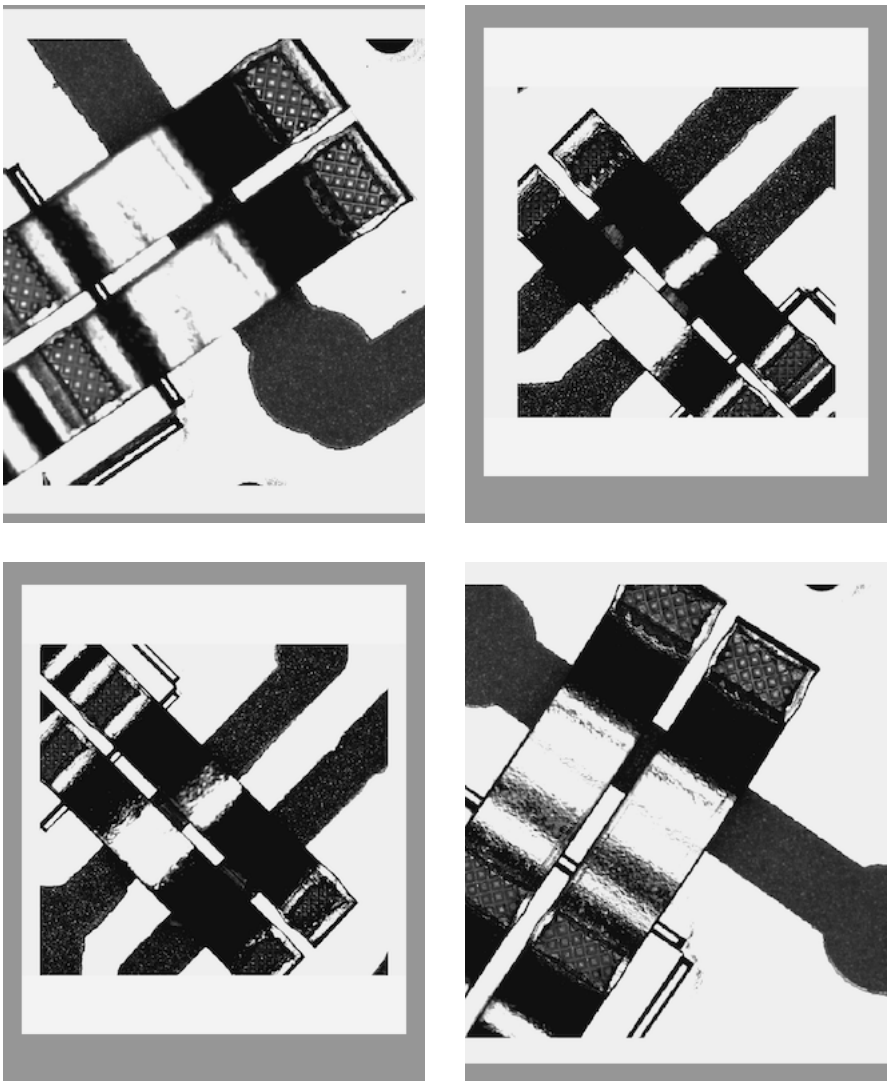

Fig. 4. – Images sous quatre angles de vue différents (DIE01, DIE02, DIE03, DIE04)

II / Méthodologie adoptée

II - 1 / Pipeline

Dans ce projet, nous proposons une méthode en plusieurs étapes pour classifier les images industrielles afin de détecter des défauts connus et inconnus (classe « drift »). Nous réalisons d'abord un prétraitement consistant en une rotation spécifique selon l'angle de vue et le composant, puis un recadrage centré sur la bande centrale, conformément au benchmark proposé.

Une étape de détection préalable utilisant la méthode PADIM permet ensuite d'identifier les anomalies non prévues (« drift »). L'objectif est d'éviter que le réseau classifie incorrectement ces cas hors distribution.

Enfin, nous utilisons des réseaux de neurones convolutionnels (**CNN**), testés selon différentes profondeurs et hyperparamètres, pour effectuer la classification finale des images en classe « GOOD » ou en catégories de défauts préalablement définies.

L'étude vise ainsi à évaluer l'effet combiné du prétraitement, de la détection d'anomalies et du CNN, tout en tenant compte des limites intrinsèques à la métrique utilisée (**PWA**).

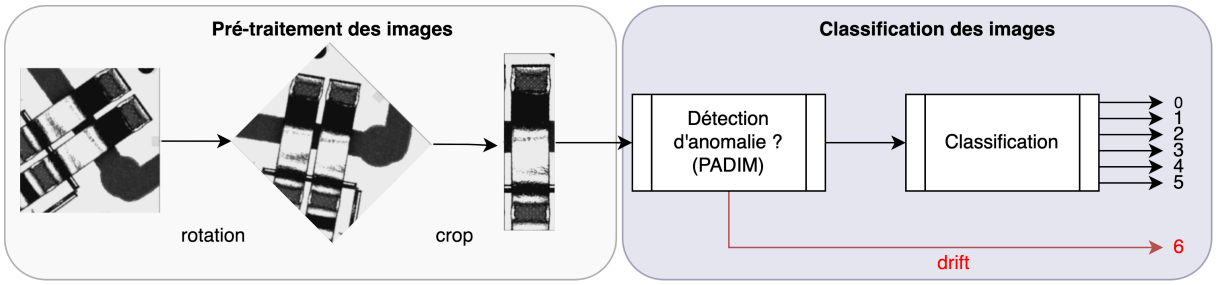

Fig. 5. – Pipeline de projet

Phase d'entraînement

III / Résolution du problème

Dans le cadre de ce projet, et compte tenu des ressources GPU limitées disponibles, nous avons fixé le nombre d'époques d'entraînement à 10. Ce choix permet d'obtenir un bon compromis entre la performance du modèle et le temps de calcul nécessaire.

III - 1 / CNN

III - 1 . 1 / CNN basique

Dans un premier temps, nous avons mis en place un réseau de neurones convolutionnel simple, sans architecture spécifique pré-entraînée. Celui-ci est composé de deux couches de convolution suivies d'une couche fully-connected finale. L'objectif de cette première étape est d'évaluer les performances de base en classification ainsi que la capacité initiale d'apprentissage du modèle. Les résultats obtenus sont analysés à l'aide de la **PWA**.

```python
# Bloc 1
x = Conv2D(32, (3, 3), activation='relu', padding='same')(inputs)
x = BatchNormalization()(x)
x = Conv2D(32, (3, 3), activation='relu', padding='same')(x)
x = BatchNormalization()(x)
x = MaxPooling2D((2, 2))(x)
x = Dropout(0.25)(x)

# Bloc 2
x = Conv2D(64, (3, 3), activation='relu', padding='same')(x)
x = BatchNormalization()(x)
x = Conv2D(64, (3, 3), activation='relu', padding='same')(x)
x = BatchNormalization()(x)
x = MaxPooling2D((2, 2))(x)
x = Dropout(0.25)(x)

# Partie Fully Connected
x = Flatten()(x)
x = Dense(128, activation='relu')(x)
x = Dropout(0.5)(x)
outputs = Dense(num_classes, activation='softmax')(x)

model = Model(inputs, outputs, name="cnn_basic")
model.compile(optimizer=Adam(),
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

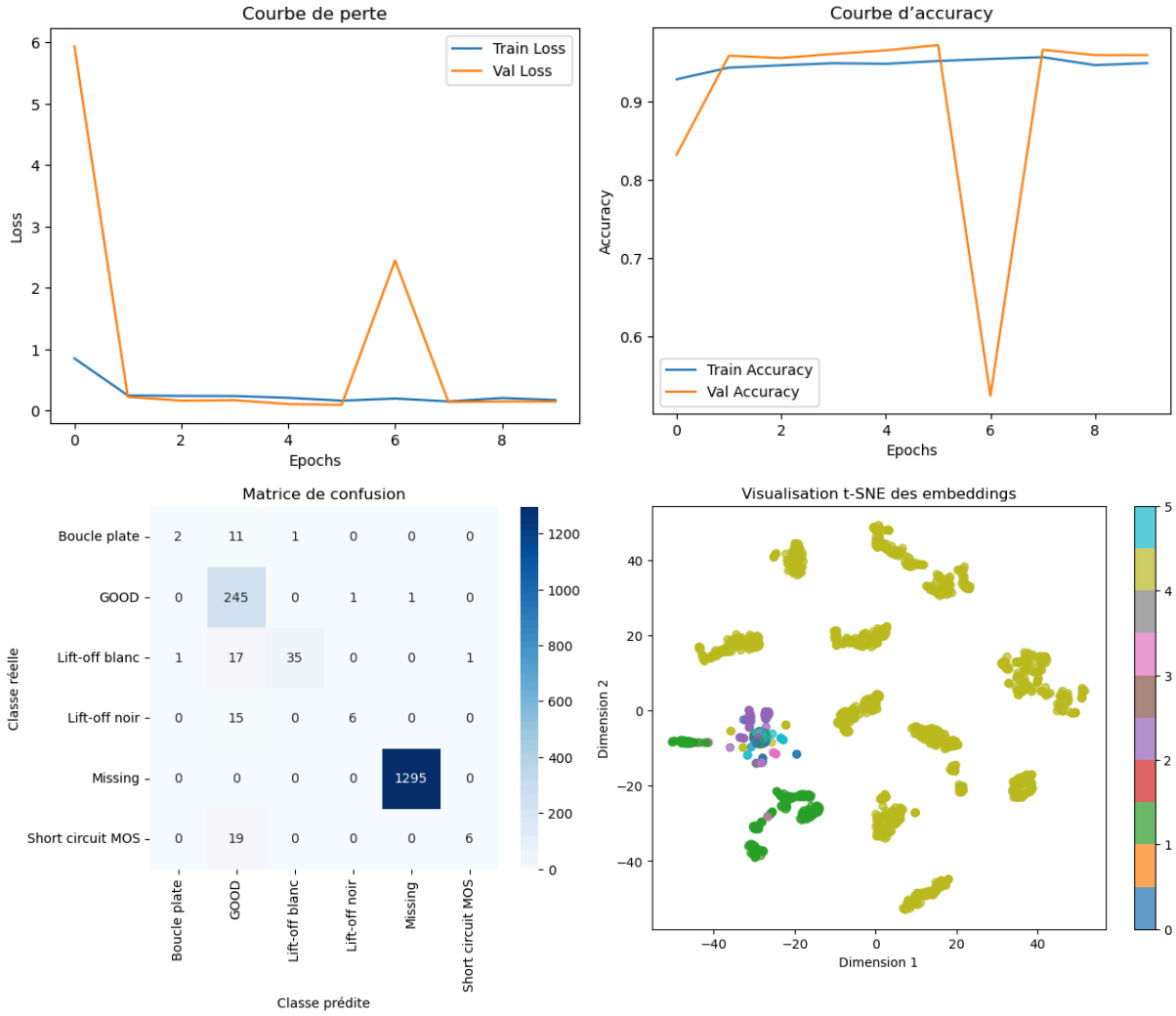

Fig. 7. – Dashboard des performance du CNN avec 3 couches

Score PWA : 0.9561

Le modèle actuel donne rapidement une bonne capacité à classifier correctement les classes majoritaires (« GOOD » et « Missing »), comme le montre l'évolution rapide de la courbe d'accuracy. Cependant, la matrice de confusion montre que il y a des difficultés importantes pour identifier correctement les classes minoritaires telles que « Boucle plate », « Lift-off noir », ou « Short circuit MOS ». Ce phénomène s'explique par le déséquilibre significatif entre les différentes classes du jeu de données d'entraînement.

La visualisation t-SNE des embeddings confirme l'analyse précédente, illustrant clairement une bonne séparation des classes majoritaires, mais aussi un grand chevauchement entre les clusters des classes minoritaires, expliquant ainsi leur faible précision de classification.

Afin d'améliorer les performances sur ces classes plus rares, nous envisageons d'approfondir l'architecture du réseau, en augmentant le nombre de couches convolutionnelles, pour permettre au modèle d'extraire des caractéristiques plus fines et mieux différenciées.

III - 1 . 2 / CNN plus profond

Nous avons augmenté la profondeur du modèle en ajoutant deux blocs de convolution supplémentaires, respectivement composés de couches Conv2D à 128 puis 256 filtres. Ce changement

augmente fortement le nombre de paramètres, permettant théoriquement au modèle d'extraire des caractéristiques plus complexes des images.

```python
# Bloc 1
x = Conv2D(32, (3, 3), activation='relu', padding='same')(inputs)
x = BatchNormalization()(x)
x = Conv2D(32, (3, 3), activation='relu', padding='same')(x)
x = BatchNormalization()(x)
x = MaxPooling2D((2, 2))(x)
x = Dropout(0.25)(x)

# Bloc 2
x = Conv2D(64, (3, 3), activation='relu', padding='same')(x)
x = BatchNormalization()(x)
x = Conv2D(64, (3, 3), activation='relu', padding='same')(x)
x = BatchNormalization()(x)
x = MaxPooling2D((2, 2))(x)
x = Dropout(0.25)(x)

# Bloc 3
x = Conv2D(128, (3, 3), activation='relu', padding='same')(x)
x = BatchNormalization()(x)
x = Conv2D(128, (3, 3), activation='relu', padding='same')(x)
x = BatchNormalization()(x)
x = MaxPooling2D((2, 2))(x)
x = Dropout(0.25)(x)

# Bloc 4
x = Conv2D(256, (3, 3), activation='relu', padding='same')(x)
x = BatchNormalization()(x)
x = Conv2D(256, (3, 3), activation='relu', padding='same')(x)
x = BatchNormalization()(x)
x = MaxPooling2D((2, 2))(x)
x = Dropout(0.25)(x)

# Partie Fully Connected
x = Flatten()(x)
x = Dense(256, activation='relu')(x)
x = Dropout(0.5)(x)
outputs = Dense(num_classes, activation='softmax')(x)

model = Model(inputs, outputs, name="cnn_deep")
model.compile(optimizer=Adam(),
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

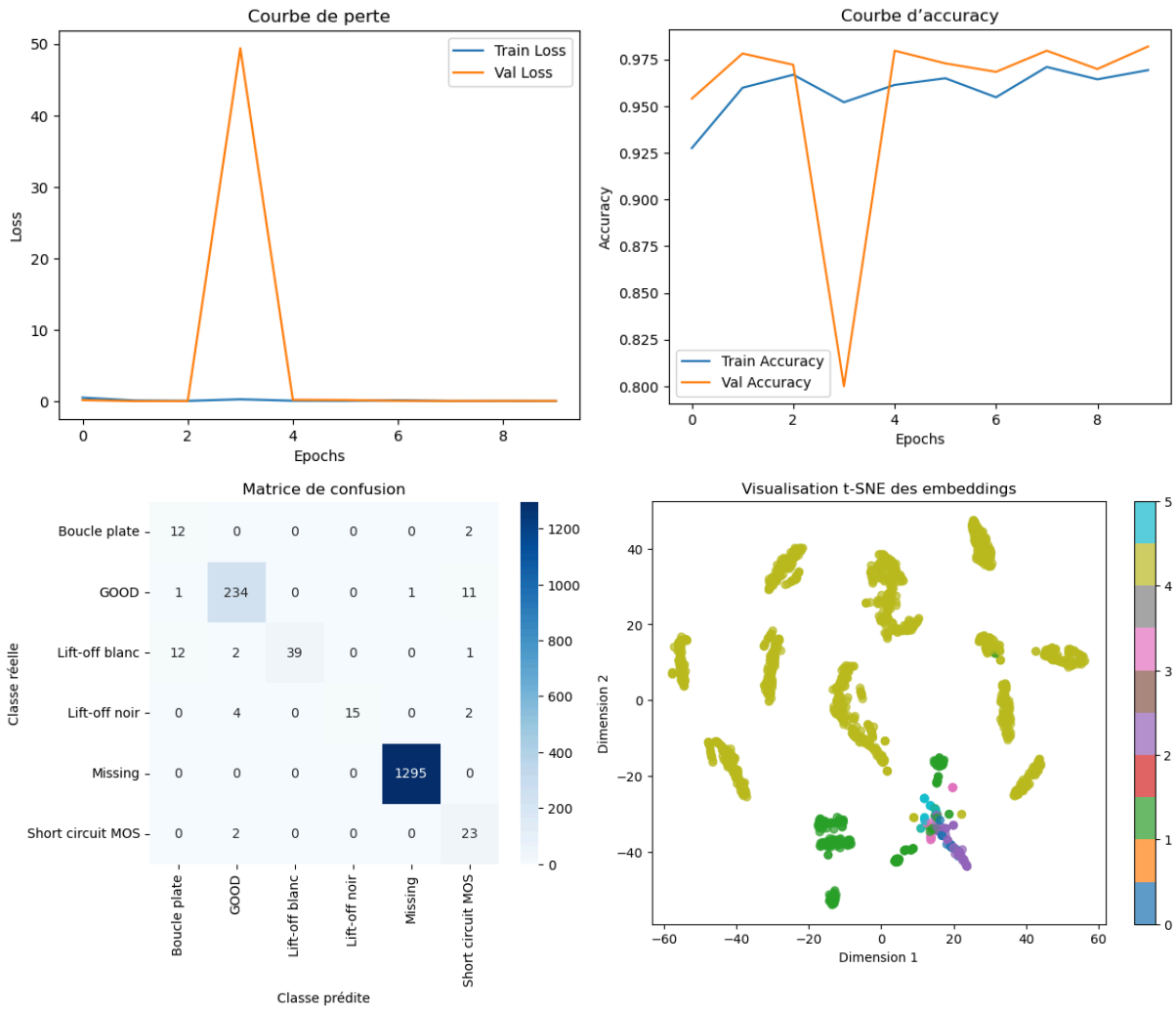

Fig. 8. – Dashboard des performance du CNN avec 5 couches

Score PWA : 0.0.9942

Cette architecture plus profonde montre effectivement une amélioration notable des performances globales, comme en témoigne la courbe d'accuracy qui atteint rapidement des niveaux très élevés. La matrice de confusion montre une amélioration claire dans la reconnaissance des classes minoritaires, notamment « Boucle plate » et « Short circuit MOS », avec une réduction sensible des erreurs de classification.

La visualisation t-SNE confirme ces améliorations avec une meilleure séparation des clusters pour ces classes auparavant problématiques, malgré quelques chevauchements résiduels. Cette séparation plus nette montre une meilleure performance du réseau profond à capturer des caractéristiques distinctives et complexes.

Lee score PWA atteint une valeur particulièrement élevée de 0.9942, ce qui montre la pertinence d'approfondir le réseau.

III - 2 / Transfert learning

Nous abordons maintenant une approche différente en utilisant le transfert learning. Cette méthode consiste à réutiliser des architectures pré-entraînées sur de grands ensembles de données, comme ImageNet, afin d'exploiter directement les caractéristiques déjà apprises pour notre propre tâche. Nous évaluerons ainsi différentes architectures, notamment ResNet50 et ResNet101, en comparant leurs

performances et en observant l'impact du dégel progressif des couches finales sur notre problème de classification.

III - 3 / Resnet50

Nous avons réalisé un entraînement du modèle ResNet-50 en transférant les poids issus d'ImageNet et en dégelant uniquement les six dernières couches. Ce choix vise à profiter des caractéristiques génériques apprises précédemment tout en adaptant légèrement le modèle à notre jeu de données spécifique.

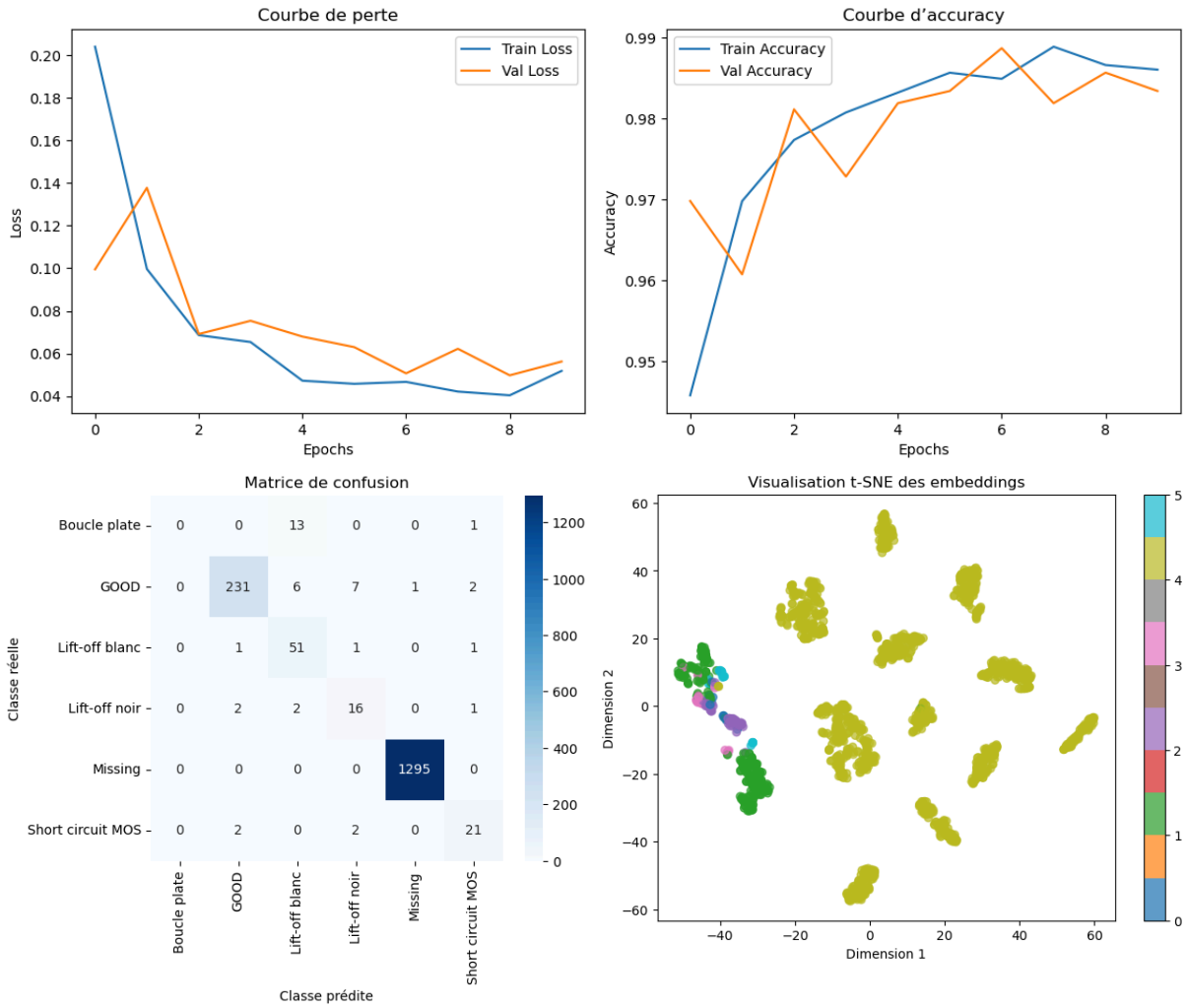

Fig. 9. – Dashboard des performance du ResNet50 avec 6 couches dégelées

Score PWA : 0.9963

Les résultats obtenus montrent un apprentissage rapide et stable, avec une précision élevée dès les premières époques, confirmée par les courbes d'accuracy et de perte. La matrice de confusion indique une excellente capacité du modèle à classifier correctement les classes majoritaires, en particulier la classe « Missing », parfaitement reconnue. Toutefois, certaines classes minoritaires telles que « Boucle plate » demeurent difficiles à prédire, illustrant les limites de cette configuration partiellement dégelée.

La visualisation t-SNE des embeddings montre une bonne séparation des classes principales avec quelques chevauchements persistants pour les classes minoritaires, ce qui souligne un potentiel d'amélioration supplémentaire en dégelant davantage de couches du modèle. Nous allons explorer cette piste en poursuivant nos expérimentations.

Maintenant, nous allons essayer de dégeler 12 puis 24 couches du modèle ResNet50 pour voir si cela permet au modèle d'apprendre de nouvelles caractéristiques à partir de nos données améliore la performance du réseau.

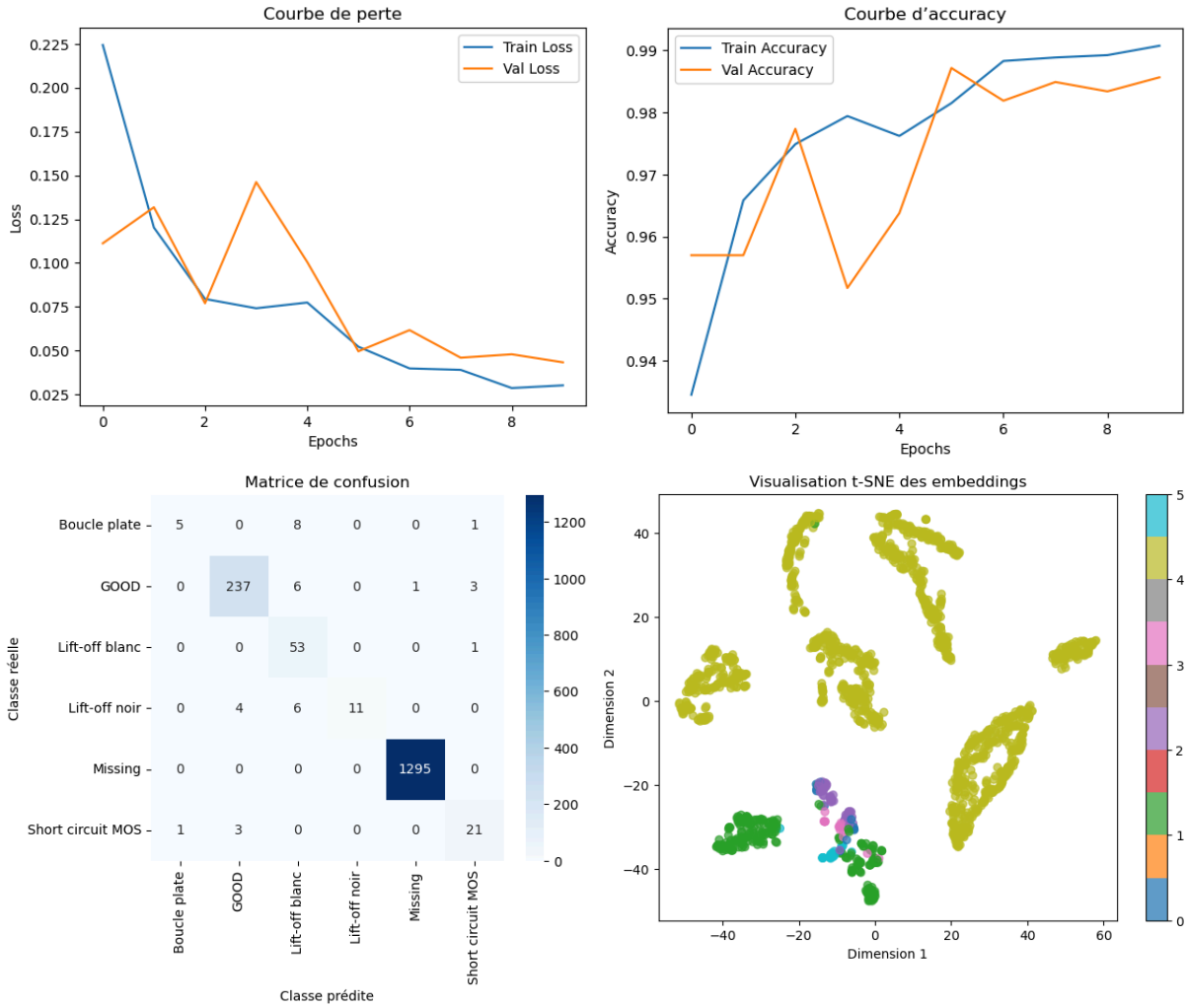

Fig. 10. – Dashboard des performance du ResNet50 avec 12 couches dégelées

Score PWA : 0.9950

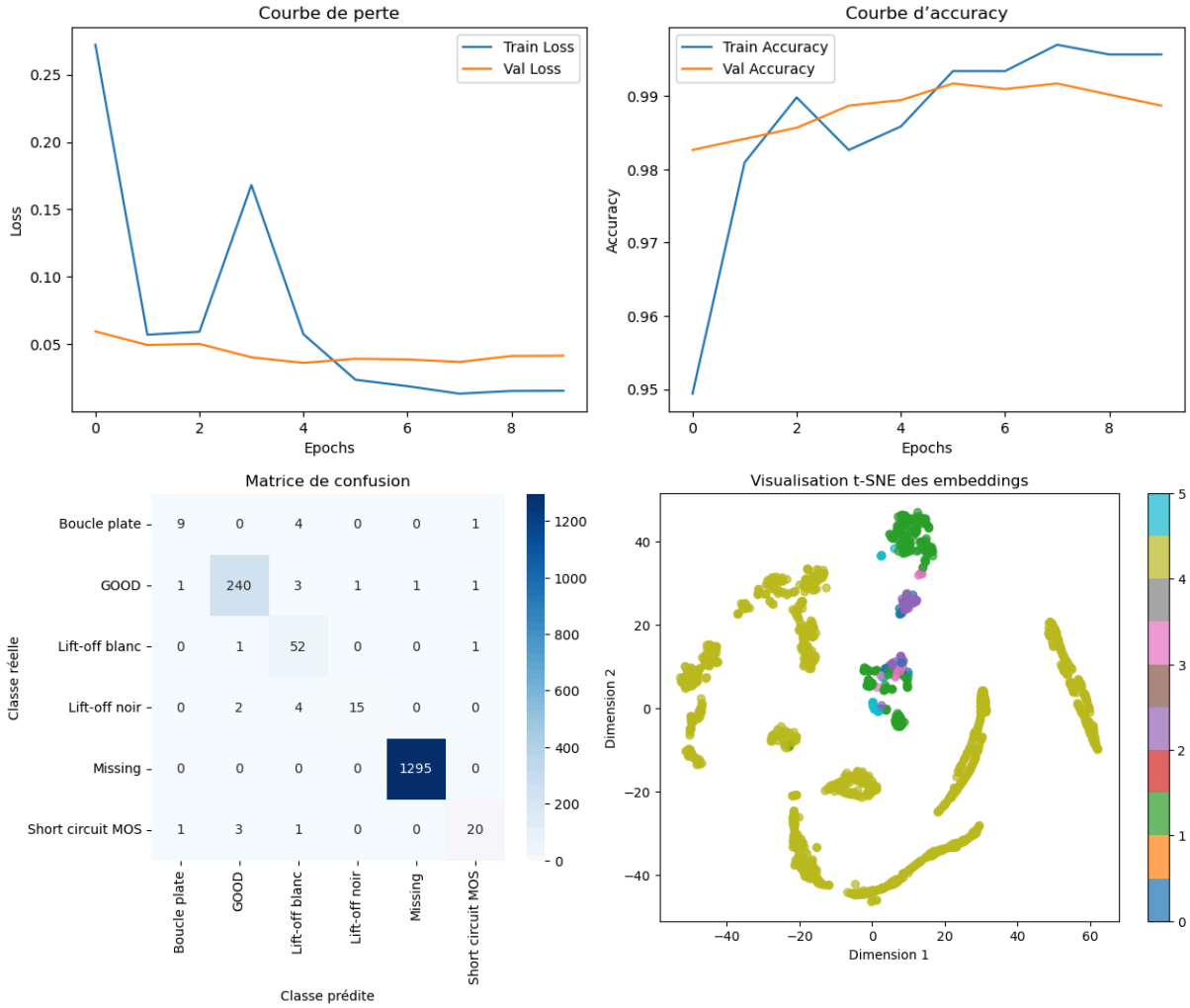

Fig. 11. – Dashboard des performance du ResNet50 avec 24 couches dégelées

Score PWA : 0.9957

- 12 couches dégelées :

Le modèle présente une légère amélioration dans la séparation des clusters, visible sur la visualisation t-SNE où les clusters apparaissent mieux définis. La matrice de confusion confirme cette tendance avec une réduction notable des erreurs pour certaines classes minoritaires comme « Lift-off blanc ». Toutefois, la classe « Boucle plate » demeure problématique, ce qui pourrait être dû à un nombre limité d'échantillons d'entraînement pour cette classe.

- 24 couches dégelées :

Le dégel supplémentaire des couches améliore légèrement la précision de classification pour les classes minoritaires, comme en témoigne la meilleure détection de la classe « Boucle plate ». Cependant, on constate également un léger surapprentissage marqué par une divergence entre les courbes d'accuracy d'entraînement et de validation sur les dernières époques.

III - 3 . 1 / Matrice des coûts personnalisée

Dans cette expérience, nous avons utilisé une matrice de coût personnalisée dans la fonction de perte afin d'améliorer la pertinence du modèle vis-à-vis des pénalités spécifiques associées à chaque erreur de classification. Cette approche visait notamment à maximiser le score **PWA**.

$$C = \begin{pmatrix} 0.0 & 10.0 & 2.0 & 2.0 & 2.0 & 2.0 \\ 5.0 & 0.0 & 5.0 & 5.0 & 5.0 & 5.0 \\ 2.0 & 10.0 & 0.0 & 2.0 & 2.0 & 2.0 \\ 2.0 & 10.0 & 2.0 & 0.0 & 2.0 & 2.0 \\ 2.0 & 10.0 & 2.0 & 2.0 & 0.0 & 2.0 \\ 2.0 & 10.0 & 2.0 & 2.0 & 2.0 & 0.0 \end{pmatrix} \tag{1}$$

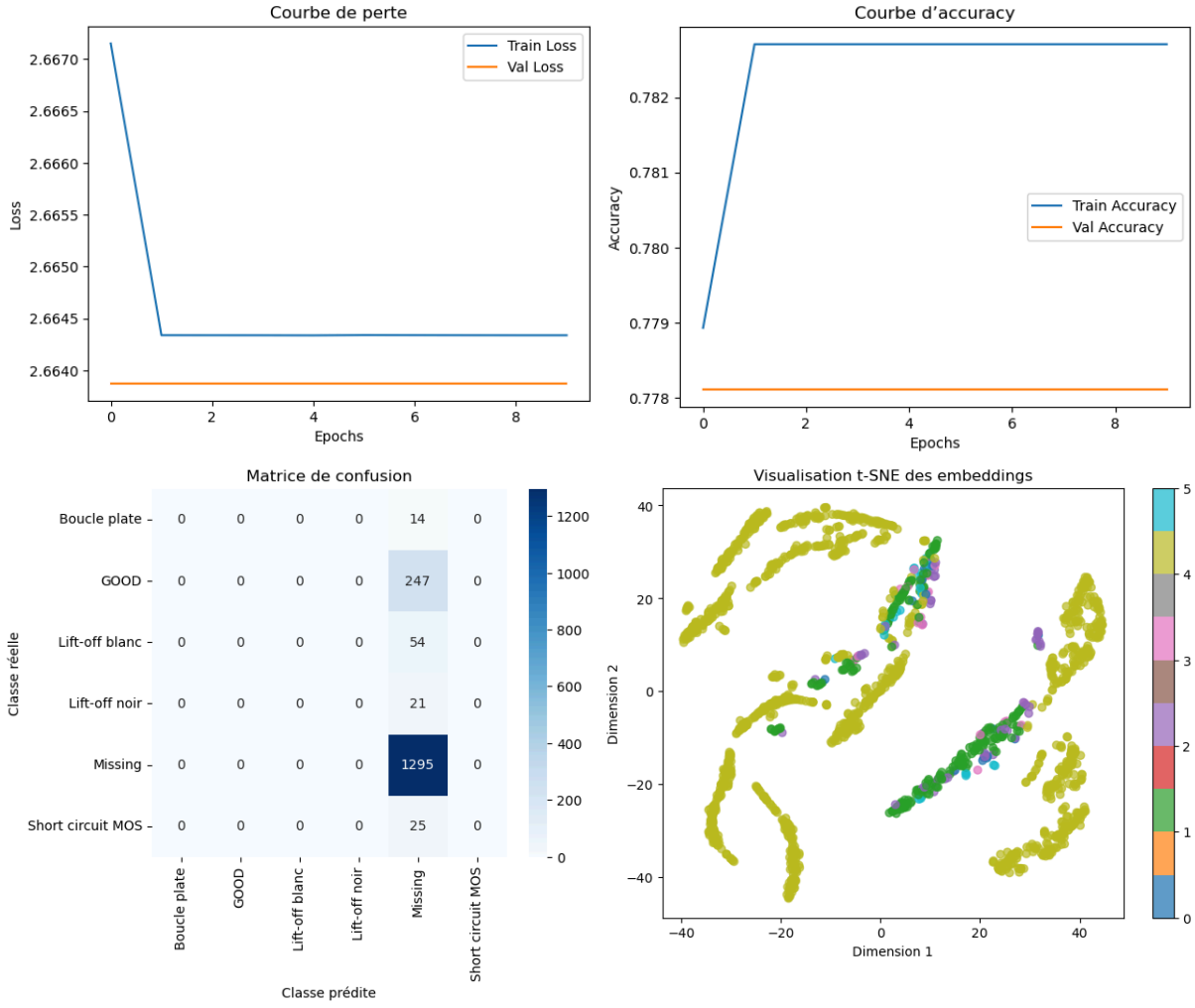

Fig. 12. – Dashboard des performance du ResNet50 avec 3 couches dégelées et matrice de coût personnalisée

Score PWA : 0.9984

Les résultats montrent un problème : le modèle subit une stagnation quasi immédiate de l'apprentissage, comme l'indiquent les courbes de perte et d'accuracy qui restent constantes après la première époque. Plus précisément, le modèle n'apprend absolument pas à distinguer les différentes classes et prédit systématiquement une classe unique (« Missing »), générant une matrice de confusion très étonnante avec toutes les prédictions concentrées sur une seule classe.

On se retrouve donc exactement dans le cas limite évoqué précédemment concernant la notation **PWA** : bien que le modèle ne discrimine aucune classe correctement, hormis la classe majoritaire, le score **PWA** est artificiellement élevé (0.9984). Ce résultat met en évidence une limite de l'utilisation d'une matrice de coûts personnalisée trop déséquilibrée lors de l'apprentissage : si les pénalités sont mal ajustées, elles peuvent empêcher la convergence du modèle vers des solutions pertinentes.

Les autres essaies effectués avec une matrice de poids personnalisée génère exactement la même matrice de confusion. Nous allons donc continuer à utiliser la loss sparse categorical crossentropy pour l'apprentissage.

Rang	Date	Participant(s)	Score public
1	14 mars 2025 20:47	lucas-versini & Gabou	0,9941
2	4 mars 2025 22:57	TomML	0,9910
3	13 mars 2025 23:57	shiwenli	0,9889
4	18 mars 2025 10:39	NicolasThiou & yacdad	0,9886
5	18 mars 2025 00:40	StephaneS	0,9883
6	14 mars 2025 22:28	emilio-picard & rosaliemillner	0,9877
7	**18 mars 2025 11:14**	**EliotM**	**0,9860**

Fig. 13. – Résultat avec l'essai trivial

III - 3 . 2 / Synthèse

L'utilisation du transfert learning avec ResNet50 pré-entraîné sur ImageNet démontre globalement une bonne performance sur le jeu de données . On observe des limites liées à la faible représentation de certaines classes minoritaires. Le dégel progressif de couches supplémentaires améliore modérément la détection de ces classes, mais introduit des risques de surapprentissage.

Configuration	PWA	Observations	Conclusion
6 couches dégelées	0.9963	Très bonne détection des classes majoritaires (Missing, good). Difficultés à détecter les classes rares.	Bonne base d'apprentissage mais limites sur classes rares.
12 couches dégelées	0.9950	Légère amélioration des classes minoritaires (Lift-off blanc). Classe boucle plate reste mal détectée	Petit bénéfice du dégel supplémentaire
24 couches dégelées	0.9957	Amélioration sur Boucle plate et Lift-off noir. Léger surapprentissage.	Léger gain qualitatif mais risque de surappprentissage
Matrice de coût personnalisée (3 couches dégelées)	0.9984	Effondrement de l'apprentissage. Toute les prédictions sont concentré sur la classe majoritaire (Missing). Score **PWA** artificiellement élevé.	Usage de matrice mal adapté à l'objectif industriel.

Tableau 1. – Synthèse des résultats de ResNet50

L'approche optimale semble être un compromis intermédiaire (12 à 24 couches dégelées) permettant d'améliorer légèrement la détection des classes rares sans trop risquer le surapprentissage. L'usage de la loss classique (sparse categorical crossentropy) est préférable à une matrice de coût personnalisée trop déséquilibrée.

III - 4 / Architecture Resnet101

Maintenant, nous allons évaluer une architecture ResNet plus profonde : ResNet-101, toujours en transfert learning. L'objectif est d'étudier si une profondeur accrue permet de mieux capturer les caractéristiques spécifiques des classes rares ou complexes, ce que l'architecture précédente (ResNet-50) n'avait pas totalement réussi à différencier. Nous analyserons ainsi les effets d'un dégel progressif de 3, 6, 12 puis 24 couches, afin d'observer l'impact de cette profondeur supplémentaire sur les performances du modèle.

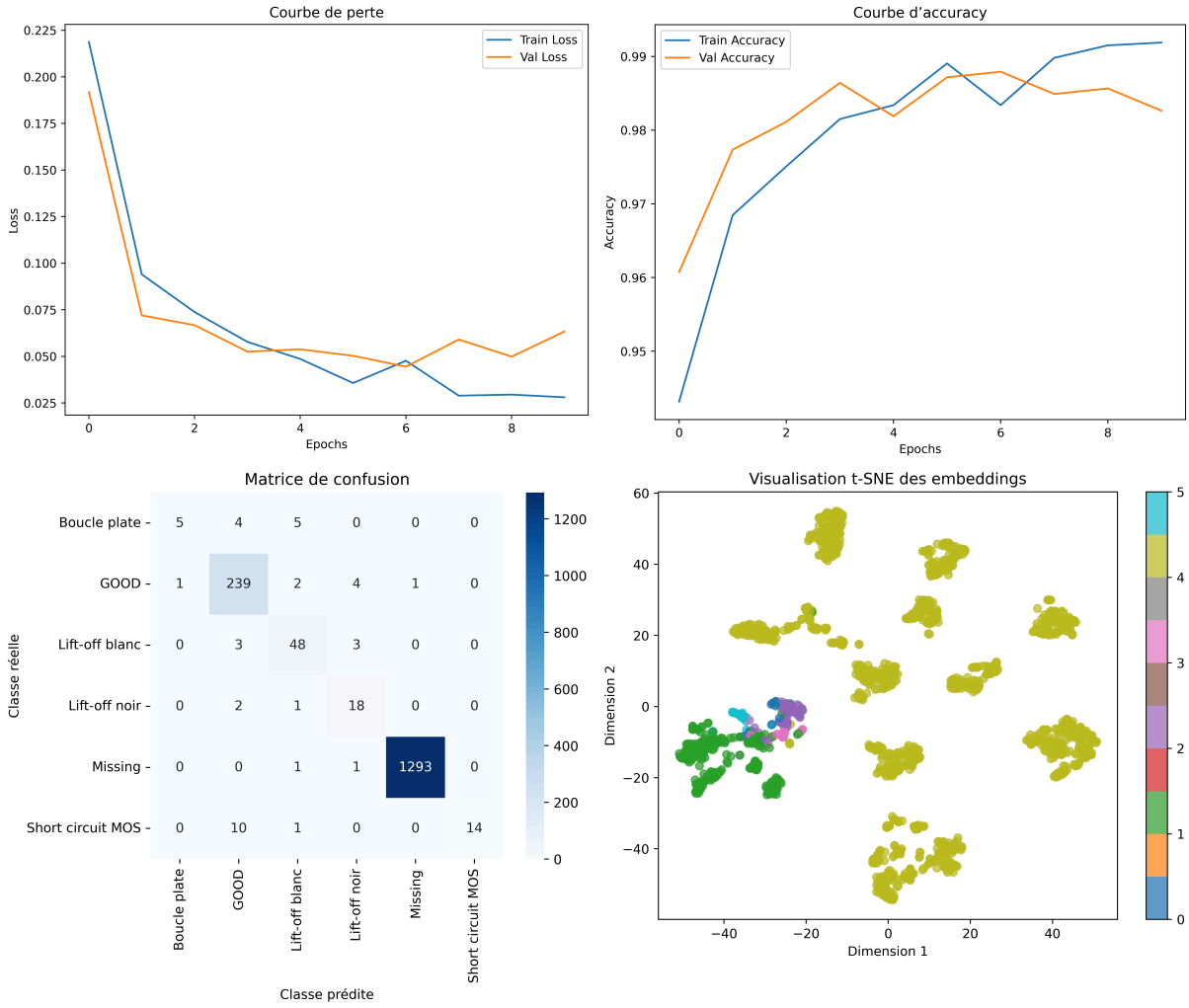

Fig. 14. – Dashboard des performance du ResNet101 avec 3 couches dégelées

Score PWA : 0.9881

- 3 couches dégelées :

Le modèle atteint rapidement une bonne performance, mais certaines confusions persistent, notamment pour la classe minoritaire « Boucle plate ». La visualisation t-SNE révèle des clusters relativement bien formés, mais quelques chevauchements subsistent. Le score **PWA** est élevé (0.9881), mais reste améliorable.

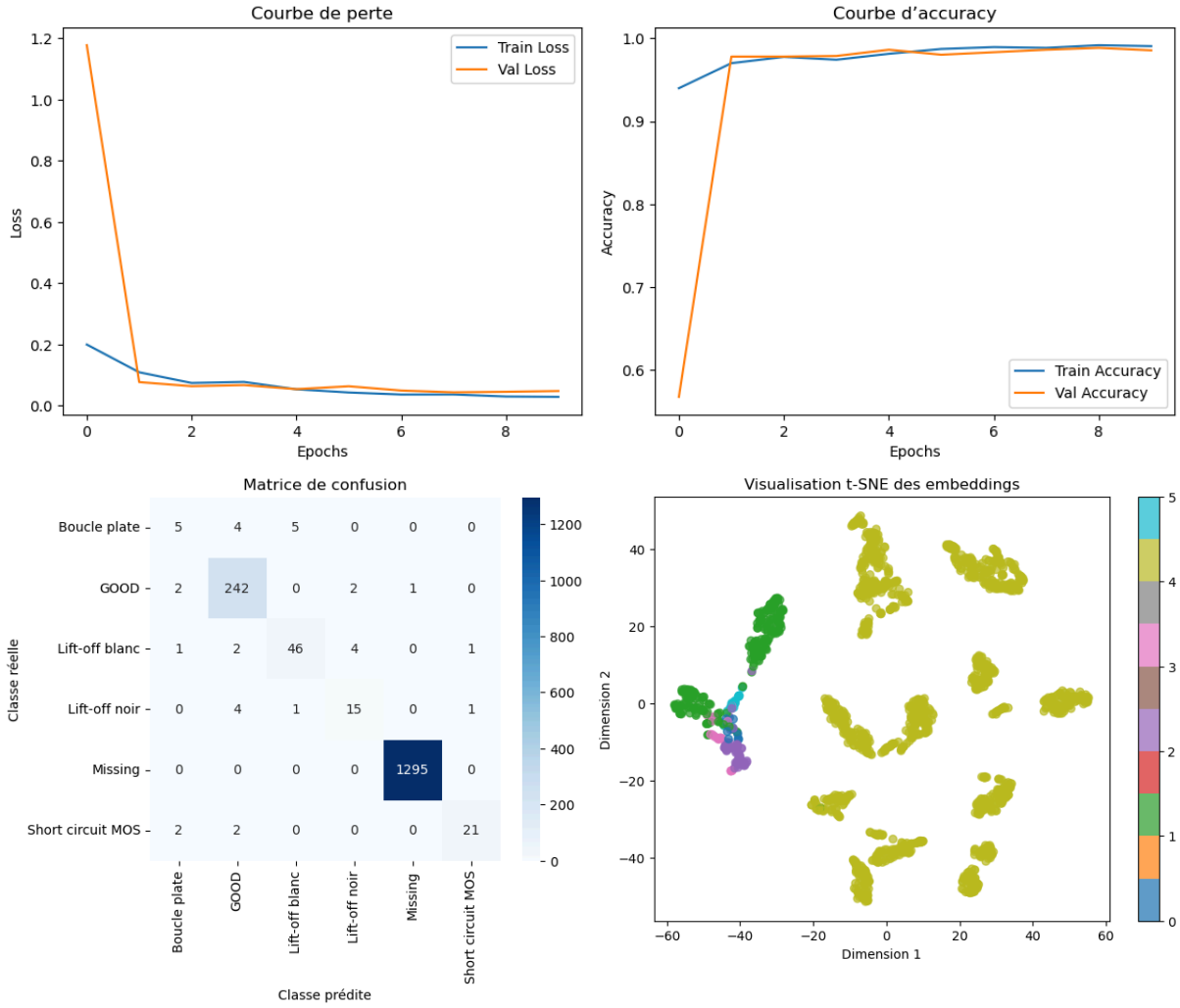

Fig. 15. – Dashboard des performance du ResNet101 avec 6 couches dégelées

- 6 couches dégelées :

Score PWA : 0.9915

Avec six couches dégelées, le modèle converge très rapidement vers une solution stable avec une meilleure précision globale. La matrice de confusion indique une nette amélioration, notamment pour les classes intermédiaires comme « Lift-off noir ». La séparation des classes dans la visualisation t-SNE est meilleure, confirmant une discrimination plus marquée. Le **PWA** augmente à 0.9915.

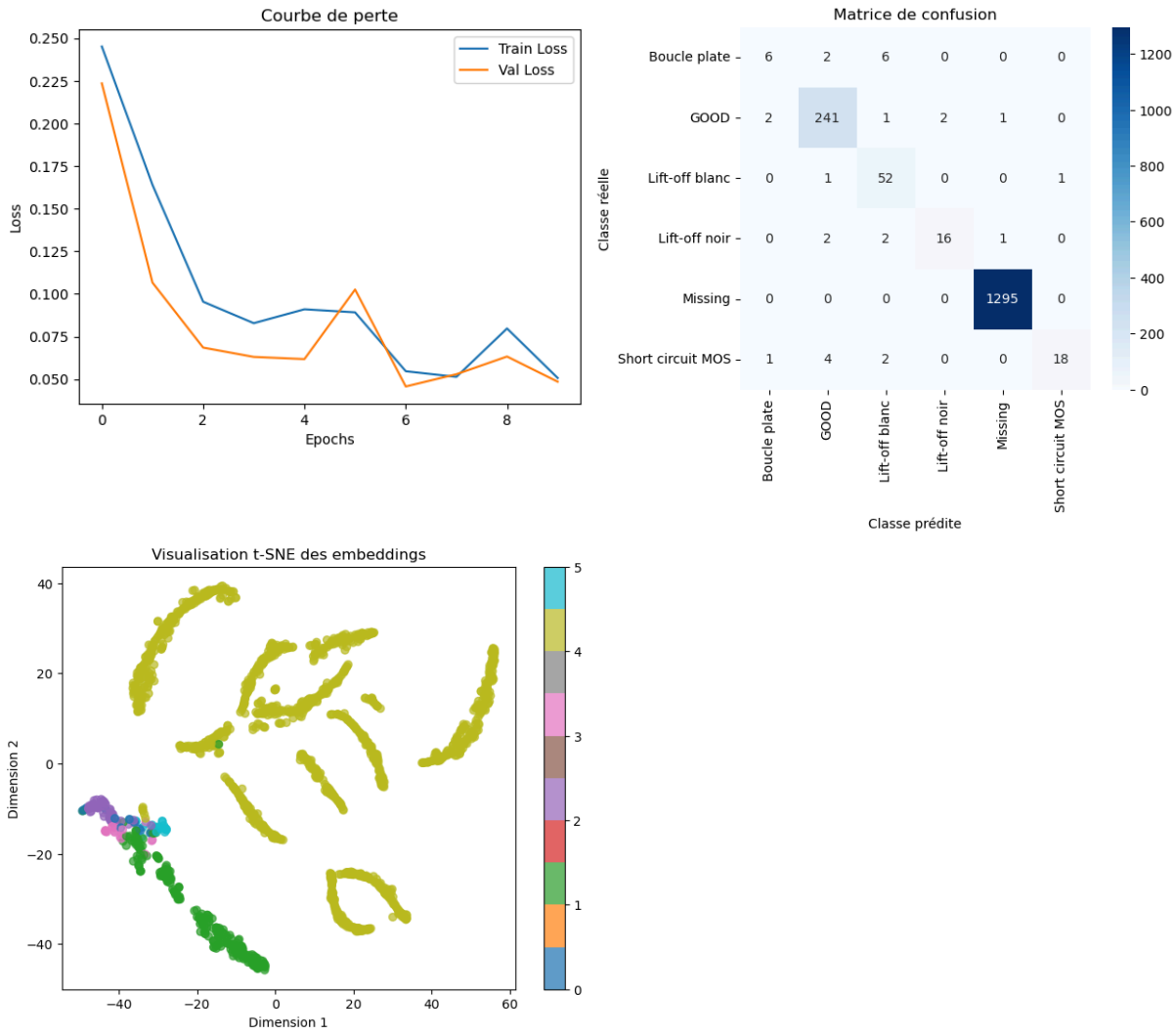

Fig. 16. – Dashboard des performance du ResNet101 avec 12 couches dégelées et dropout de 0.3

- 12 couches dégelées et plus fort dropout :

Le dégel de 12 couches avec ajout d'un dropout modéré montre une performance supérieure avec une réduction des confusions pour presque toutes les classes. La visualisation t-SNE révèle des clusters très distincts, indiquant un apprentissage plus efficace des caractéristiques discriminantes. Le score **PWA** s'améliore encore à 0.9936.

Score PWA : 0.9936

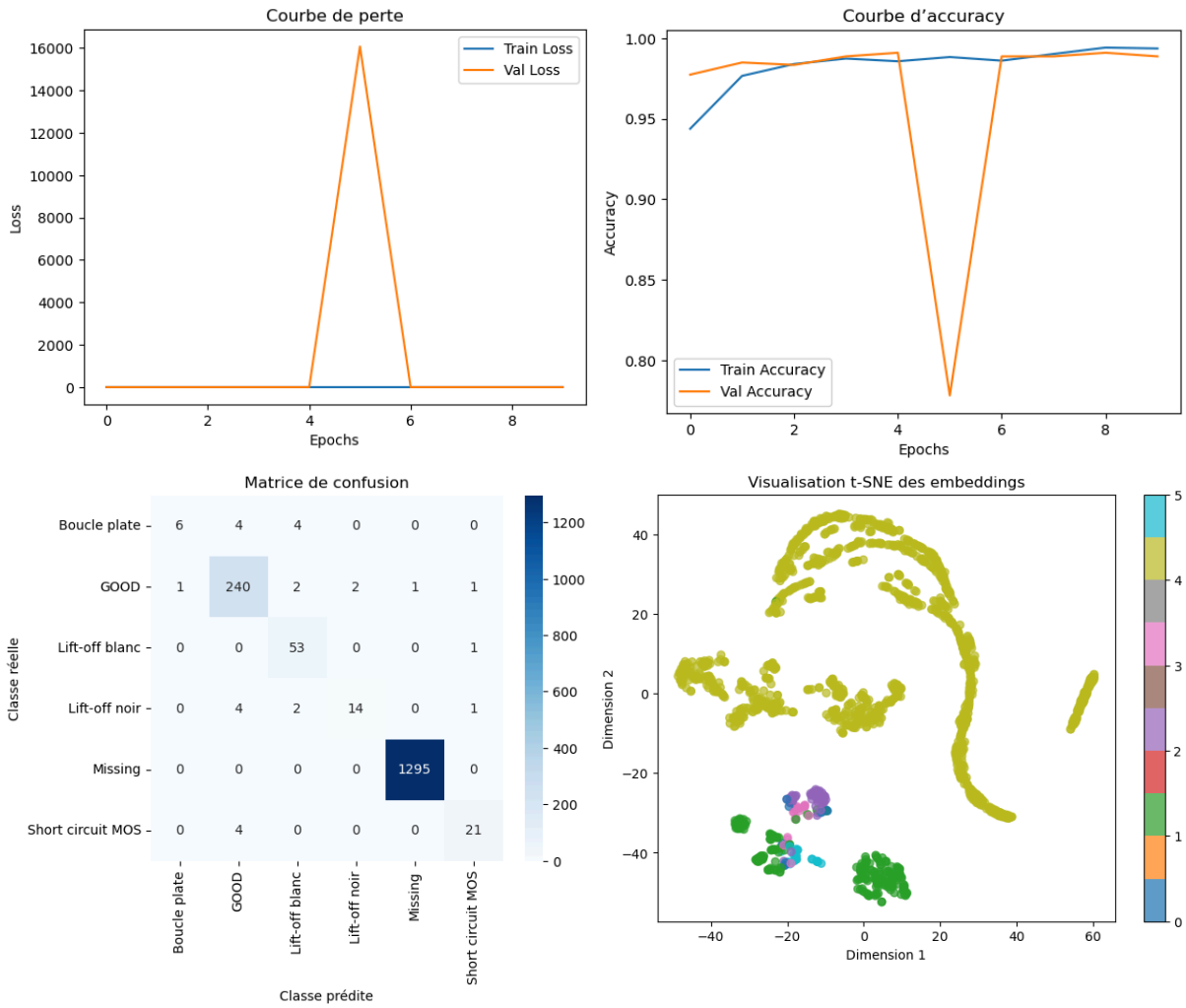

Fig. 17. – Dashboard des performance du ResNet101 avec 24 couches dégelées et dropout de 0.3

Score PWA : 0.9914

- 24 couches dégelées et plus fort dropout :

L'augmentation du nombre de couches dégelées à 24 génère une légère instabilité (pic temporaire dans la perte), suggérant un début de surapprentissage malgré l'utilisation du dropout. Les résultats restent très satisfaisants, mais ne dépassent pas ceux obtenus avec 12 couches dégelées. La visualisation t-SNE reste excellente, et le score PWA (0.9914) est très proche du résultat précédent.

III - 4 . 1 / Synthèse

Configuration	PWA	Observations	Conclusion
3 couches dégelées	0.9881	Bonne séparation initiale mais confusion sur la classe minoritaire Boucle plate.	Apprentissage rapide, mais pas suffisant pour discriminer les classes minoritaire.
6 couches dégelées	0.9915	Nette amélioration globale, réduction des erreurs notamment sur la classe Lift-off noir.	Compromis intéressant entre performance et stabilité
12 couches dégelées	0.9936	Excellente séparation des cluster, nette amélioration des résultat dans toutes les classes	Meilleur équilibre global
24 couches dégelées	0.9914	Légère instabilité, début de surapprentissage mais bon séparation des clusters	Pas d'amélioration par rapport à 12 couches, légère instabilité.

Tableau 2. – Synthèse des résultats de ResNet101

La meilleure performance pour le modèle ResNet-101 est obtenue avec 12 couches dégelées et un dropout modéré de 0.3, permettant un apprentissage optimal sans surapprentissage excessif. Au-delà de ce point, l'apprentissage supplémentaire n'apporte pas de bénéfice clair et peut même introduire une légère instabilité.

Conclusion

IV / Conclusion

Ce projet nous a permis d'explorer et de comparer plusieurs approches pour la classification automatique d'images industrielles afin d'identifier des défauts avec une attention particulière sur les classes minoritaires. Nous avons mis en place un pipeline complet, incluant prétraitement, détection préalable des anomalies, et classification par réseaux neuronaux convolutionnels (CNN).

Nous avons d'abord testé des architectures CNN simples, puis plus profondes, qui ont montré des gains de performance significatifs sur la reconnaissance des défauts minoritaires. Ensuite, l'utilisation du transfert learning avec les architectures ResNet-50 et ResNet-101 pré-entraînées sur ImageNet a considérablement amélioré les résultats, en exploitant les caractéristiques génériques apprises sur de larges jeux de données.

Nos expérimentations montrent que le choix d'une profondeur modérée (12 couches dégelées) avec un dropout adéquat offre le meilleur compromis en termes de performance et de robustesse, en réduisant les erreurs critiques tout en évitant le surapprentissage. Nous avons également mis en évidence une limite importante : l'usage d'une matrice de coûts personnalisée dans ce cas précis ne semble pas adapté car la **PWA** comporte un biais de notation que le modèle exploite.

En piste d'amélioration, nous pourrions essayer de proposer une nouvelle métrique d'évaluation pénalisant le biais évoqué précédemment.

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. “ImageNet Classification with Deep Convolutional Neural Networks”, Advances in Neural Information Processing Systems (NeurIPS), 2012.

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. “Deep Residual Learning for Image Recognition”, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770-778.